Creating a Test Statistic to Find if the Number of Misses in a Song from Guitar Hero is Random

Shannon Coyle, Samantha Colucci, and Brianna Cirillo

December 2020

1 INTRODUCTION

The game, Guitar Hero, collects information regarding the number of "hits" recorded by the player. This poses the question about the randomness of the misses in a song and if they are correlated to the difficulty of the part of the song.

The study investigates the randomness of different songs using varying methods. These methods looked at the number of miss streaks over the total number of misses, distances between each miss, and using the runs test. The resampling of the data sets was performed through parametric bootstrapping, permutation tests, and regular bootstrapping. Using resampling and our methods together allowed us to assess the randomness of a given song based on the results.

This report explains the methodology used in order to test the randomness of a song. The methods and resampling used throughout the project gave results for the empirical type 1 error and the power of the test. These methods were applied to different songs, therefore, returning different results. This report will also look at the limitations and future ideas for the project, and how some of our methods worked well while others did not.

2 METHODOLOGY

2.1 Method 1

The first method looks at the proportion of the number of miss streaks over the number of total misses. A streak was defined as one or more misses in a row since it was easier with the code and some samples did not have that many streaks to look at. From this method, we discovered that the proportion does not tell us anything about the location of the misses. This proved to be difficult to determine randomness of misses without knowing the exact location of them.

2.2 Method 2

The second method calculates the distances between misses in a song. The idea behind this method was that if we can see where the misses are occurring, we can see if they are random. We can seeing where the distances are occurring randomly and conclude that the misses must also be occurring randomly.

There was an attempt to use a median distance as a basis rather than the mean because the distances were varied, so the average would not tell us that much. This led to a problem because a lot of the songs had long miss streaks, which lead to a distance of 0, so the median would also become zero. With this small issue, it led to the idea of use the runs test on this method and, therefore, the distances.

2.3 Runs Test

Knowing that the runs test is a preconceived test for randomness, the idea behind using this was to use it on the distances of misses within a song, rather than just using it on the song. As mentioned previously if a conclusion can be reached that the distances are random, this would imply randomness of the misses in the song.

A run is defined as a series of increasing or decreasing values, while the number of values in each series is the length of each run. The number of runs in a data set, denoted R, is crucial because this is the number that is used to derive the test statistic. The other values involved in deriving the test statistics is the expected number of runs, denoted $\bar{R}$, the median value

of the data sample, the standard deviation of the number of runs, denoted s_R, and the number of values below and above the median.

Once the test statistic is derived, the runs test comes to a conclusion about whether or not to reject the null hypothesis. Here, the test defines:

H_0: the sequence is random

H_a: the sequence is not random

The runs test program used was found from the snpar package in RStudio. A p-value, the number of runs and an alternative hypothesis of nonrandomness is produced once the test is run. This package was specifically chosen because was created to be used on discrete data, rather than continuous. When using this method, each song was resampled and a distance vector for each sample was computed. The runs test was then used on the distances to determine if the null hypothesis of randomness should be rejected or not.

2.4 Resampling Methods

2.4.1 Parametric Bootstrap

The Parametric Bootstrap was used to generate a bootstrap using a parameterized distribution. It takes in a specified number of resamples then performs a Bernoulli distribution in order to resample the data. The Bernoulli distribution looks at the sample size then the event probability, which was set to 50

2.4.2 Permutation Resampling

The permutation resampling method is a type of randomization that was developed for data that does not conform to the assumptions needed to perform the statistical method desired. There are multiple advantages to using this type of resampling. In order to implement this, one does not need to know the distribution of the data and it can also be used on small sample sizes. A major disadvantage to using this method is the amount of computer power it takes, since as the sample size begins to increase, the number of permutations computed during the resampling increases rapidly. For this reason, when the permutation resampling is used throughout Methods 1, 2 and the Runs test, the amount of samples produced by this resampling is smaller than that produced by the other resampling techniques.

The main element of this resampling method, that differentiates it from

most other methods, is that it resamples *without* replacement. Essentially, it redistributes the original elements of the data in a different order. Though this sampling can be beneficial to use here since the distribution of our data is unknown, it also presents issues when checking for randomness. For example, if there is a detectable pattern in a specific song or sample, the permutation resampling may cause the presence of that pattern to disappear, due to the way the resampling is done. When this became an issue during the research, a different approach was taken to ensure that if the songs being analyzed do have a pattern, that pattern is not lost. This is discussed more in the Power of the Test section.

2.4.3 Nonparametric Bootstrap

Nonparametric bootstrapping is taking a sample of the same size as the data, from the data, but with replacement. This means that even if a data point is resampled, it can be resampled again. The goal of bootstrapping is to approximate the sampling distribution by simulating the data, then using this resamples data as the real data. This allows for data analysis on data that we are not willing to make assumptions about the parameters. On average, the number of times data was bootstrapped was between 200 and 5000 times.

3 EMPIRICAL TYPE 1 ERROR AND POWER SIMULATIONS

To look into the empirical type 1 error rates, random songs were created. In order to create the songs,a sample of 50 observations were taken from a binomial distribution with a probability of 0.5. The songs were resampled using each of the aforementioned methods. The distances of misses in the resampled sample, were then calculated. The runs test was then performed on these distances for each alpha from 0.001 to 0.1 that were incremented at 0.005. This therefore returned the average of the p-values that were less that 0.05 for each theoretical alpha.

For the final power simulations, different scenarios were used in order to create a song that is not random. This means that the songs created should make the null hypothesis false. Doing this will show how well the metrics identify nonrandomness.The scenarios using positive pairwise corre-

lation generated two songs, one with 200 notes and one with 600 notes. The probability model used was that the probability of missing a note was 0.2 and the correlation between any two consecutive notes was 0.3.The scenarios using blocks with different difficulties also created two songs, one with 200 notes and one with 600 notes. For each song created using this scenario, there were easy, medium, hard, and very hard sections. The probability of missing a note in an easy section is 0.01, in a medium section is 0.05, in a hard section 0.25, and in a very hard section is 0.5. The scenarios using the autoregressive model of order 1 generated six songs, three with 200 notes and three with 600 notes. The probability model used for the first type is the probability of missing a note is 0.1 and the correlation between any two notes is $0.5^{|i-j|}$. For the probability model for the second type, the probability of missing a note stayed 0.1, but the correlation between any two notes is $0.3^{|i-j|}$. For the third type, the probability of missing a note is 0.2 and the correlation between any two notes is $0.5^{|i-j|}$.

3.1 Parametric Bootstrap

When resampling with a parametric bootstrap, multiple simulations were run with a varying number of observations - it was usually 50 or 100 observations.

3.1.1 Empirical Type 1 Error

To calculate the empirical type 1 error rate, the parametric bootstrap worked better with 50 observations as opposed to 100 observations. There were also two random songs made to see how the error rate varies between songs. These random songs were generated through code in which they were both 50 notes long and had a 50% chance of being a 0 or a 1. Alpha was increased by 0.01, but error rates between 0.005 and 0.001 ended up being 0.00. The table above shows the difference between the two random songs and how the second random song displayed more of a variation of error values. The second random song showed to have more of a curve and varying error rates, whereas, the first random song had multiple occurrences of the same p-value for different alphas. Refer to figures 1 and 2 in the appendix.

Alpha	Empirical - Random Song 1	Empirical - Random Song 2
$\alpha = 0.001$	0.00	0.00
$\alpha = 0.005$	0.01	0.00
$\alpha = 0.006$	0.01	0.02
$\alpha = 0.007$	0.01	0.02
$\alpha = 0.008$	0.01	0.04
$\alpha = 0.009$	0.01	0.04
$\alpha = 0.01$	0.01	0.04
$\alpha = 0.02$	0.01	0.04
$\alpha = 0.03$	0.01	0.06
$\alpha = 0.04$	0.01	0.06
$\alpha = 0.05$	0.03	0.07
$\alpha = 0.06$	0.06	0.08
$\alpha = 0.07$	0.08	0.09
$\alpha = 0.08$	0.08	0.13
$\alpha = 0.09$	0.08	0.13
$\alpha = 0.1$	0.08	0.14

Table 1: The table above shows the type 1 error rates obtained with their respective alphas.

3.1.2 Final Power Simulations

When using the Parametric Bootstrap to return a power, they were generally very low p-values or 0. The power was always equal to 0 when alpha was set to 0.001 and many of the scenarios gave 0 for every value of alpha. There were three scenarios that gave actual values which were the both blocks methods and the autoregressive model of order 1 that had 200 notes per song, a probability of missing a note as p=0.1, and $\rho = 0.3$. From the table, it is evident that most of the values obtained are 0 with exceptions for three of the scenarios. Aside from the table, the graphs also show the large number of occurrences of 0 as a power. Refer to figures 3 and 4 in the appendix, to see the visualization.

Type 1 Error Rate Scenario	$\alpha = 0.001$	$\alpha = 0.005$	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.10$
Neg. Pair Correlation, n = 200	0.00	0.00	0.00	0.00	0.00
Neg. Pair Correlation, n = 600	0.00	0.00	0.00	0.00	0.00
Blocks, n = 200	0.00	0.03	0.03	0.11	0.30
Blocks, n = 600	0.00	0.00	0.01	0.02	0.08
AR(1), p=0.1, $\rho = 0.5$, n = 200	0.00	0.00	0.00	0.00	0.00
AR(1), p=0.1, $\rho = 0.5$, n = 600	0.00	0.00	0.00	0.00	0.00
AR(1), p=0.1, $\rho = 0.3$, n = 200	0.00	0.00	0.00	0.01	0.01
AR(1), p=0.1, $\rho = 0.3$, n = 600	0.00	0.00	0.00	0.00	0.00
AR(1), p=0.2, $\rho = 0.5$, n = 200	0.00	0.00	0.00	0.00	0.00
AR(1), p=0.2, $\rho = 0.5$, n = 600	0.00	0.00	0.00	0.00	0.00

Table 2: The table shows the power at each theoretical alpha for each of the scenarios above.

3.2 Permutation Resampling

3.2.1 Empirical Type 1 Error

A random song of 50 notes was created by the rbinom() function in Rstudio, which guarantees randomness. This implies that the alpha obtained from the test, the empirical alpha, should project a low rejection rate since the null hypothesis of randomness is known to be true. The resulting random song from this code was:

01111110000000010001000010111110011101001110100110

To resample the data, the permutation resampling code was used and the song was resampled 100 times. The runs test was then performed on the distances of misses in each sample. This returned the average of the p-values that were less that 0.05 for each theoretical alpha, this can be seen in the table below. The empirical alphas are mostly the same as the theoretical alphas. The empirical alphas, however, do not increase as nicely as the theoretical alphas do, due to how the test performed at each level. For the most part, these empirical alphas match the theoretical alphas. Refer to figure 5, to see the graph.

Type 1 Error	**Random Song 1**
$\alpha = 0.001$	0.00
$\alpha = 0.005$	0.00
$\alpha = 0.006$	0.00
$\alpha = 0.007$	0.00
$\alpha = 0.008$	0.00
$\alpha = 0.009$	0.01
$\alpha = 0.01$	0.01
$\alpha = 0.02$	0.03
$\alpha = 0.03$	0.05
$\alpha = 0.04$	0.06
$\alpha = 0.05$	0.09
$\alpha = 0.06$	0.09
$\alpha = 0.07$	0.10
$\alpha = 0.08$	0.11
$\alpha = 0.09$	0.11
$\alpha = 0.1$	0.11

Table 3: The table above shows the type 1 error rates obtained with their respective alpha.

3.2.2 Final Power Simulations

A similar method was used for these power simulations, however once the random song was resampled with the permutation method, 10 different methods were used in order to ensure the test was being used on songs that were nonrandom. This should give us a larger empirical alpha since the null hypothesis is known to be nonrandom. Below is a table of the outcomes of the test run with these 10 different methods. All of these power values are much lower than desired. They are consistent with previous power tests that were conducted, and all 10 simulations are mostly consistent with each other, however they are not the values that were hoped for. Refer to figure 6 and 7, to see the graph.

Type 1 Error Rate Scenario	$\alpha = 0.001$	$\alpha = 0.005$	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.10$
Neg. Pair Correlation, n = 200	0.00	0.00	0.00	0.00	0.00
Neg. Pair Correlation, n = 600	0.00	0.00	0.00	0.00	0.00
Blocks, n = 200	0.00	0.12	0.12	0.26	0.27
Blocks, n = 600	0.00	0.08	0.09	0.16	0.27
AR(1), p=0.1, $\rho = 0.5$, n = 200	0.00	0.00	0.00	0.00	0.00
AR(1), p=0.1, $\rho = 0.5$, n = 600	0.00	0.00	0.00	0.00	0.00
AR(1), p=0.1, $\rho = 0.3$, n = 200	0.01	0.01	0.01	0.01	0.01
AR(1), p=0.1, $\rho = 0.3$, n = 600	0.00	0.00	0.00	0.00	0.00
AR(1), p=0.2, $\rho = 0.5$, n = 200	0.00	0.00	0.00	0.00	0.00
AR(1), p=0.2, $\rho = 0.5$, n = 600	0.00	0.00	0.00	0.00	0.00

Table 4: The table shows the power at each theoretical alpha for each of the scenarios above.

3.3 Nonparametric Bootstrap

3.3.1 Empirical Type 1 Error

By taking 50 samples from the binomial distribution, the resulting random song was:

01000100110101111111010011001001000101010001011110

To resample the data, the nonparametric bootstrap was used and the song was resampled 100 times. The runs test was then performed on the distances of misses in the bootstrapped sample. This returned the average of the p-values that were less that 0.05 for each theoretical alpha, which can be seen in the table below.

The empirical alpha seems to very roughly, match the theoretical alphas. Although, the empirical alphas were not increasing consistently. They would get stuck at certain alphas such as 0.02 and 0.05. Therefore, the theoretical alphas are increasing at a faster rate than the empirical alphas are. Refer to figure 8 to see the plot.

3.3.2 Final Power Simulations

For the final power simulations, the majority of the scenarios gave a power of 0. This is extremely low, but overall pretty consistent with our previous

findings. The blocks scenario, when the song had 200 notes, gave errors of 0.04 for alphas equal to 0.005, 0.01, 0.05, and 0.10. When the song had 600 notes, the error rates were 0.01 for alphas equal to 0.005, 0.01, 0.05, and 0.10. This potentially means that the runs test on the distance of misses detected nonrandomness more in the block scenarios. In the autocorrelation scenario with 200 notes, where the probability of missing a note was 0.1, and the correlation between any two notes was $0.3^{|i-j|}$, when alpha was 0.05 and 0.10, the error was 0.01. This could mean that when alpha is 0.05 and 0.10 in this scenario of autocorrelation, that nonrandomness was detected more. In the autocorrelation scenario with 600 notes, where the probability of missing a note was 0.2, and the correlation between any two notes was $0.5^{|i-j|}$, when alpha was 0.005, 0.01, 0.05, and 0.10 the error was 0.01. Therefore, the detection of nonrandomness was higher in this scenario. This is shown in the table below and figures 9-12 in the appendix.

4 APPLICATION

The hits and misses data was collected on seven songs: "Judith", "Hurts", "American Girl", "Funky", "Ring of Fire", "Watchtower", and "Wolf". For each of these songs, the methods explained above were used to determine if the hits and misses were random. Since method 2 calculates the distances between misses, which was used for the runs test calculations, method 2 was not tested on its own. Therefore, method 1 and the runs test on the distances between misses were calculated for each song using the three different resampling methods.

To obtain a p-value for each song, 100 bootstraps were performed to change the 0's and 1's to different orders and sequences. Previously, 1000 bootstraps were being done, but it was taking a significant amount of time to return a p-value. The null hypothesis being followed is that the songs are random and the alternative hypothesis is that the songs are not random. A significance level of $\alpha = 0.05$ is being used. The results we have here are based solely off of the p-values obtained from these results. Confidence intervals would be a way to improve the strength of these results, which is discussed more in the Future Research section of this paper.

4.0.1 Parametric Bootstrap

The p-value of the Method 1 test tells us to reject the null hypothesis while the Run tests p-value tells us to fail to reject the null hypothesis.

4.0.2 Permutation Resampling

Using just these p-values, we would reject the null hypothesis of randomness using Method 1, however fail to reject the null hypothesis when using the runs test on the distances. This implies that we may come to two different conclusions about the randomness of the songs depending on which method we are using.

4.0.3 Nonparametric Bootstrap

The nonparametric bootstrap was used to resample the song "Judith", in order to test for randomness using method 1 and the runs test. The resulting p-values for each of the tests can be seen in the table below. After the song bootstrapped 1000 times, the statistic for method 1 was calculated, the proportion of the number of miss streaks over the number of total misses. This resulted in a p-value of 0. The null hypothesis for method 1 is randomness, thus we would reject the null at the level 0.05. Therefore, we cannot conclude that the song, "Judith" is random.

To find the test statistic for the runs test, "Judith" was bootstrapped 100 times, the distances between misses was calculates, and the runs test was ran on the distances. This resulted in a p-value of 0.998. The null hypothesis of the runs test is that the data is normal. Since we fail to reject the null at the level 0.05, we do not have enough evidence to conclude that the data is not random.

Type 1 Error	**Random Song 1**
$\alpha = 0.001$	0.00
$\alpha = 0.005$	0.00
$\alpha = 0.006$	0.00
$\alpha = 0.007$	0.00
$\alpha = 0.008$	0.02
$\alpha = 0.009$	0.02
$\alpha = 0.01$	0.02
$\alpha = 0.02$	0.03
$\alpha = 0.03$	0.03
$\alpha = 0.04$	0.05
$\alpha = 0.05$	0.05
$\alpha = 0.06$	0.05
$\alpha = 0.07$	0.05
$\alpha = 0.08$	0.06
$\alpha = 0.09$	0.07
$\alpha = 0.1$	0.08

Table 5: A function was created to loop through p-values from 0.001 to 0.1 that incremented by 0.005, in order to generate the Type 1 Error Rates. The table above shows the rates obtained with their respective alpha.

Type 1 Error Rate Scenario	$\alpha = 0.001$	$\alpha = 0.005$	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.10$
Neg. Pair Correlation, n = 200	0.00	0.00	0.00	0.00	0.00
Neg. Pair Correlation, n = 600	0.00	0.00	0.00	0.00	0.00
Blocks, n = 200	0.00	0.04	0.04	0.04	0.04
Blocks, n = 600	0.00	0.01	0.01	0.01	0.01
AR(1), p=0.1, $\rho = 0.5$, n = 200	0.00	0.00	0.00	0.00	0.00
AR(1), p=0.1, $\rho = 0.5$, n = 600	0.00	0.00	0.00	0.00	0.00
AR(1), p=0.1, $\rho = 0.3$, n = 200	0.00	0.00	0.00	0.01	0.01
AR(1), p=0.1, $\rho = 0.3$, n = 600	0.00	0.00	0.00	0.00	0.00
AR(1), p=0.2, $\rho = 0.5$, n = 200	0.00	0.00	0.00	0.00	0.00
AR(1), p=0.2, $\rho = 0.5$, n = 600	0.00	0.01	0.01	0.01	0.01

Table 6: The table shows the power at each theoretical alpha for each of the scenarios above.

Method Type	P-Value
Method 1	0
Runs Test	0.512

Table 7: The table shows the resulting p-values of the method 1 and runs test on the song, "Judith".

Method Type	P-Value
Method 1	0
Runs Test	0.847

Table 8: The table shows the resulting p-values of the method 1 and runs test on the song, "Judith".

Method Type	P-Value
Method 1	0
Runs Test	0.998

Table 9: The table shows the resulting p-values of the method 1 and runs test on the song, "Judith".

4.1 Song 2: "Hurts"

4.1.1 Parametric Bootstrap

The p-value of the Method 1 test tells us to reject the null hypothesis while the Run tests p-value tells us to fail to reject the null hypothesis.

Method Type	P-Value
Method 1	0
Runs Test	0.512

Table 10: The table shows the resulting p-values of the method 1 and runs test on the song, "Hurts".

4.1.2 Permutation Resampling

According to the p-value of the Method 1 test, the null hypothesis would be rejected, meaning there is enough evidence to say that the songs are not random. However, according to the p-value obtained from the Runs Test, we would fail to reject the null hypothesis of randomness. Similarly to the test done on the song "Judith" with permutation resampling, these are contrasting results.

Method Type	P-Value
Method 1	0
Runs Test	0.249

Table 11: The table shows the resulting p-values of the method 1 and runs test on the song, "Hurts".

4.1.3 Nonparametric Bootstrap

For the song, "Hurts", the nonparametric bootstrap was used to resample the song in order to test for randomness using method 1 and the runs test. The resulting p-values for each of the tests can be seen in the table below. The song was bootstrapped 1000 times and the test statistic for method 1 was calculated. The resulting p-value was 0.186. Thus we would fail to reject

the null at the level 0.05. Therefore, we can conclude that the song, "Hurts" is random.

For the runs test, "Hurts" was bootstrapped 100 times. The runs test was ran on the distances from the bootstrapped sample, which resulted in a p-value of 0.459. Therefore, we fail to reject the null at the level 0.05. We do not have enough evidence to conclude that the song, "Hurts", is not random.

Method Type	P-Value
Method 1	0.186
Runs Test	0.459

Table 12: The table shows the resulting p-values of the method 1 and runs test on the song, "Hurts".

4.2 Song 3: "American Girl"

4.2.1 Parametric Bootstrap

The p-value of the Method 1 test tells us to fail to reject the null hypothesis and the Run tests p-value tells us to fail to reject the null hypothesis as well.

Method Type	P-Value
Method 1	0.16
Runs Test	0.211

Table 13: The table shows the resulting p-values of the method 1 and runs test on the song, "American Girl".

4.2.2 Permutation Resampling

These p-values alone give us contrasting results once again. The Method 1 test p-value implies that the null hypothesis of randomness would be rejected, while the Runs Test p-value says that we should fail to reject the null hypothesis.

Method Type	P-Value
Method 1	0
Runs Test	0.414

Table 14: The table shows the resulting p-values of the method 1 and runs test on the song, "American Girl".

4.2.3 Nonparametric Bootstrap

In order to test for randomness using method 1 and the runs test, "American Girl" was bootstrapped 1000 times. This resulted in p-values for each of the tests, which can be seen in the table below. "American Girl" was bootstrapped 1000 times and the test statistic for method 1 was calculated. The resulting p-value was 0. Hence, we would reject the null at the level 0.05. Therefore, we cannot conclude that the song, "American Girl" is random.

"American Girl" was bootstrapped 100 times for the runs test. The bootstrapped sample was used to calculate the test statistic for the run's test, which resulted in a p-value of 0.841. Thus, we fail to reject the null at the level 0.05. We do not have enough evidence to conclude that the song, "American Girl", is not random.

Method Type	P-Value
Method 1	0
Runs Test	0.841

Table 15: The table shows the resulting p-values of the method 1 and runs test on the song, "American Girl".

4.3 Song 4: "Funky"

4.3.1 Parametric Bootstrap

The p-value of the Method 1 test tells us to reject the null hypothesis while the Run tests p-value tells us to fail to reject the null hypothesis.

Method Type	P-Value
Method 1	0
Runs Test	0.08

Table 16: The table shows the resulting p-values of the method 1 and runs test on the song, "Funky".

4.3.2 Permutation Resampling

These p-values are very similar to those of the tests on "American Girl" using permutation resampling. Rejecting the null hypothesis for the Method 1 test and failing to reject the null hypothesis for the Runs test leaves us with conflicting results since both tests have a null hypothesis of randomness.

Method Type	P-Value
Method 1	0
Runs Test	0.442

Table 17: The table shows the resulting p-values of the method 1 and runs test on the song, "Funky".

4.3.3 Nonparametric Bootstrap

The resulting in p-values for method 1 and the runs test, can be seen in the table below. "Funky" was bootstrapped 1000 times and the test statistic for method 1 was calculated, for which the p-value was 0. Hence, we would reject the null at the level 0.05. Therefore, we cannot conclude that the song, "Funky" is random.

The test statistic for the runs test for "Funky" was calculated after the song was bootstrapped 100 times. The p-value of the test is 0.656. We fail to reject the null at the level 0.05. We do not have enough evidence to conclude that the song, "Funky", is not random.

Method Type	P-Value
Method 1	0
Runs Test	0.656

Table 18: The table shows the resulting p-values of the method 1 and runs test on the song, "Funky".

4.4 Song 5: "Ring of Fire"

4.4.1 Parametric Bootstrap

The p-value of the Method 1 test tells us to reject the null hypothesis while the Run tests p-value tells us to fail to reject the null hypothesis.

Method Type	P-Value
Method 1	2.204 $[e^{-16}]$
Runs Test	0.44

Table 19: The table shows the resulting p-values of the method 1 and runs test on the song, "Ring of Fire".

4.4.2 Permutation Resampling

Similarly to the other results of the hypothesis tests with permutation resampling, we have differing results here. The p-value of the Method 1 test tells us to reject the null hypothesis while the Run tests p-value tells us to fail to reject the null hypothesis.

Method Type	P-Value
Method 1	2.204 $[e^{-16}]$
Runs Test	0.614

Table 20: The table shows the resulting p-values of the method 1 and runs test on the song, "Ring of Fire".

4.4.3 Nonparametric Bootstrap

The p-values for method 1 and the runs test, can be seen in the table below. "Ring of Fire" was bootstrapped 1000 times and the test statistic for method 1 was calculated. The resulting p-value was approximately 0. We would reject the null at the level 0.05. Thus, we cannot conclude that the song, "Ring of Fire" is random.

"Ring of Fire" was bootstrapped 100 times and the test statistic for the runs test was calculated. The p-value of the test is 0.317. We fail to reject the null at the level 0.05. We do not have enough evidence to conclude that the song, "Ring of Fire", is not random.

Method Type	P-Value
Method 1	0
Runs Test	0.317

Table 21: The table shows the resulting p-values of the method 1 and runs test on the song, "Ring of Fire".

4.5 Song 6: "Watchtower"

4.5.1 Parametric Bootstrap

The p-value of the Method 1 test tells us to reject the null hypothesis while the Run tests p-value tells us to fail to reject the null hypothesis.

Method Type	P-Value
Method 1	2.204 $[e^{-16}]$
Runs Test	0.66

Table 22: The table shows the resulting p-values of the method 1 and runs test on the song, "Watchtower".

4.5.2 Permutation Resampling

Here, the Method 1 test says to reject the null hypothesis of randomness and the Runs test says to fail to reject the null hypothesis.

Method Type	P-Value
Method 1	2.204 $[e^{-16}]$
Runs Test	0.577

Table 23: The table shows the resulting p-values of the method 1 and runs test on the song, "Watchtower".

4.5.3 Nonparametric Bootstrap

The p-values for method 1 and the runs test, can be seen in the table below. "Watchtower" was bootstrapped 1000 times and the test statistic for method 1 was calculated. The resulting p-value was approximately 0. We would reject the null at the level 0.05. Hence, we cannot conclude that the song, "Watchtower" is random.

The test statistic for the runs test was calculated after "Watchtower" was bootstrapped 100 times. The p-value of the test is 0.727. We fail to reject the null at the level 0.05. We do not have enough evidence to conclude that the song, "Watchtower", is not random.

Method Type	P-Value
Method 1	0
Runs Test	0.727

Table 24: The table shows the resulting p-values of the method 1 and runs test on the song, "Watchtower".

4.6 Song 7: "Wolf"

4.6.1 Parametric Bootstrap

The Method 1 test says to reject the null hypothesis of randomness and the Runs test says to fail to reject the null hypothesis.

4.6.2 Permutation Resampling

Similar to all of the other results in this section for permutation resampling, these p-values are telling us to do two different things. The p-value of the

Method Type	P-Value
Method 1	2.204 $[e^{-16}]$
Runs Test	0.58

Table 25: The table shows the resulting p-values of the method 1 and runs test on the song, "Wolf".

Method 1 test is telling us to reject the null hypothesis while the p-value of the Runs test is telling us to fail to rejcet the null hypothesis. Since these results are all differing, we cannot make a definite conclusion on whether there is enough evidence to conclude that these songs are random.

Method Type	P-Value
Method 1	0.06
Runs Test	0.265

Table 26: The table shows the resulting p-values of the method 1 and runs test on the song, "Wolf".

4.6.3 Nonparametric Bootstrap

The p-values for method 1 and the runs test, can be seen in the table below. The test statistic for method 1 was calculated after "Wolf" was bootstrapped 1000 times. The resulting p-value was 0.02. We would reject the null at the level 0.05. Hence, we cannot conclude that the song, "Wolf" is random.

The test statistic for the runs test was calculated after "Wolf" was bootstrapped 100 times. The p-value resulting from the test is 0.727. We fail to reject the null at the level 0.05. We do not have enough evidence to conclude that the song, "Wolf", is not random.

5 DISCUSSION/LIMITATIONS/IDEAS FOR FURTHER RESEARCH

Method Type	P-Value
Method 1	0.02
Runs Test	0.727

Table 27: The table shows the resulting p-values of the method 1 and runs test on the song, "Wolf".

5.1 Discussion

Overall, the tests conducted on this data were mostly inconclusive. Many of the p-values obtained led us to little to no conclusions. This however, is beneficial in a way because we made progress towards how our methods need to be modified and how they could possibly be used in the future. Due to the following limitations we were unable to obtain as many results as we had hoped, however in the Future Research section below we discuss what couldbe done to help move past these limitations and hopefully provide further results.

Limitations One of the main limitations that we ran into durig this research was the extent of what we could find out about the runs test. Since the runs test that was implemented was taken from an Rstudio package, it did not give all of the information desired about the runs test statistic, how the number of runs was calculated, and more. Though we were able to figure most of these things out, we could not derive exactly how the runs test statistic was calculated since there are different methods to compute this, depending on which form of the runs test being used. Knowing how this test statistic is computed and being able to view this in addition ot the p-value of the test would give more flexibility in what we could do with our results.

5.2 Ideas for Future Research

To continue this research, some additional things can be done. Looking into splitting the song into sections before using these tests could help determine randomness in a different way. This may also give more information on how the level of difficulty varies throughout the song, since it would be visible which specific parts of the song are random and which are not. Multiple plays of the same song may also help since there would be more data to work with and comparisons could be made between the resamplings of the

different plays.

In addition, a new method to look into would be the Serial Method. This method counts the number of "01"s together in a song. From this method, it can be concluded that the larger number of 01s meant that there is a larger number of hits turned into misses. The possible idea once the method works as anticipated is to compare the number of 01s for each individual row of a bootstrap to the total number of 01s in the entire bootstrap.

In addition to p-values, our results may have been stronger if confidence intervals were included. This was difficult to do because of the limitations of our knowledge on the runs test statistic. However, in the future, creating confidence intervals for the tests once the runs test statistic is obtained would give the ability to come to better conclusions of the tests.

6 APPENDIX

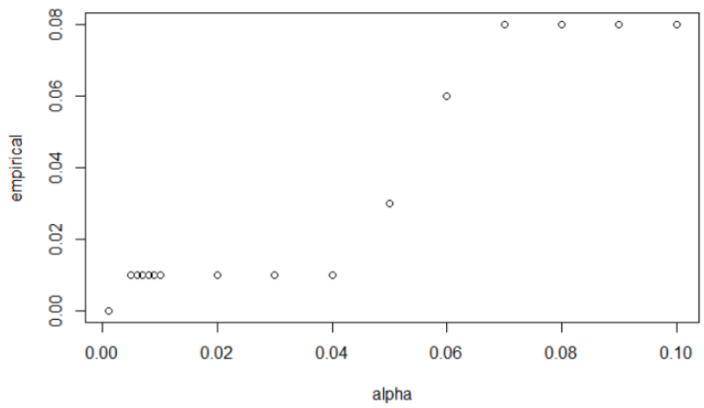


Figure 1: Graph of Type 1 error rates for Random Song 1 with 50 observations

7 PERSONAL REFLECTIONS

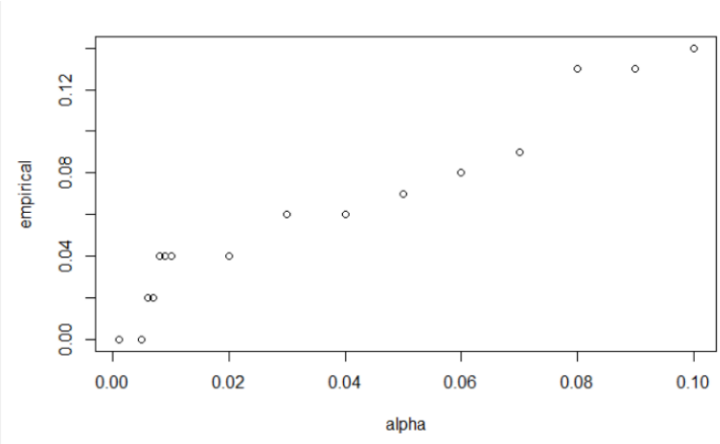


Figure 2: Graph of Type 1 error rates for Random Song 2 with 50 observations

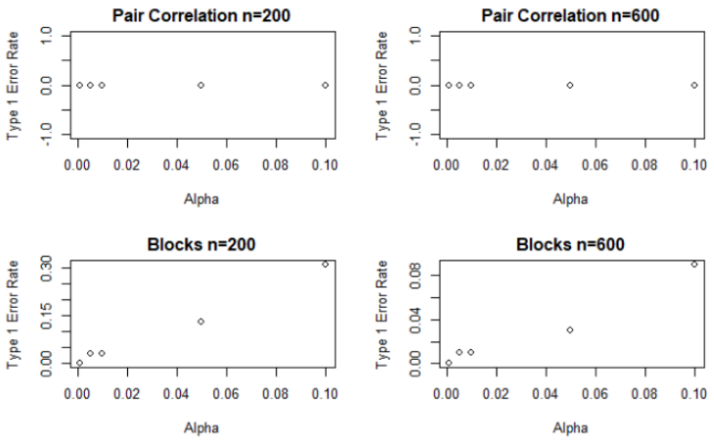


Figure 3: Graph of first four scenarios to calculate the power of a test

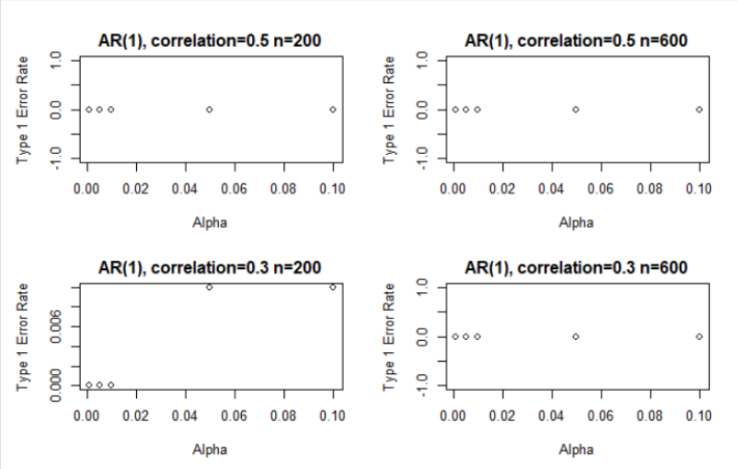


Figure 4: Graph of last four scenarios to calculate the power of a test

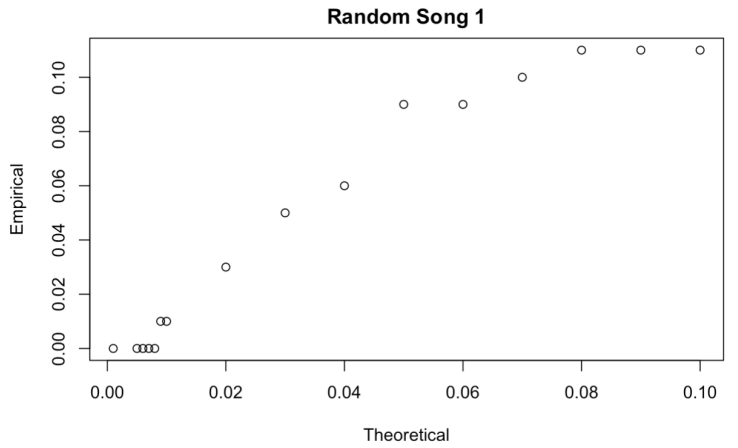


Figure 5: Graph of Type 1 error rates with 50 observations

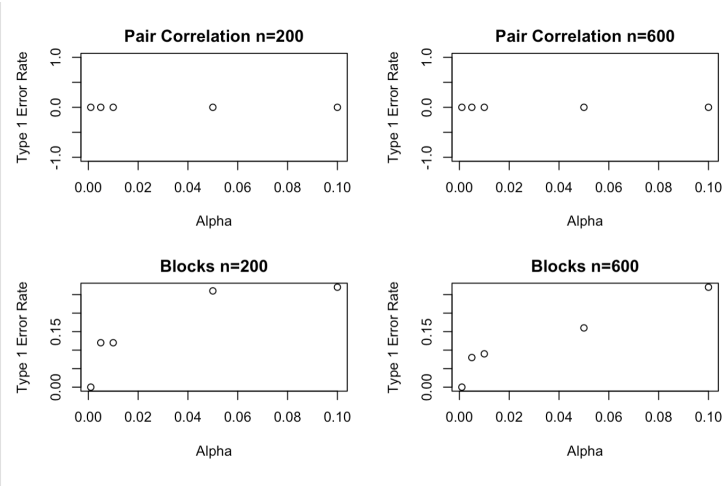


Figure 6: Graph of first four scenarios to calculate the power of a test

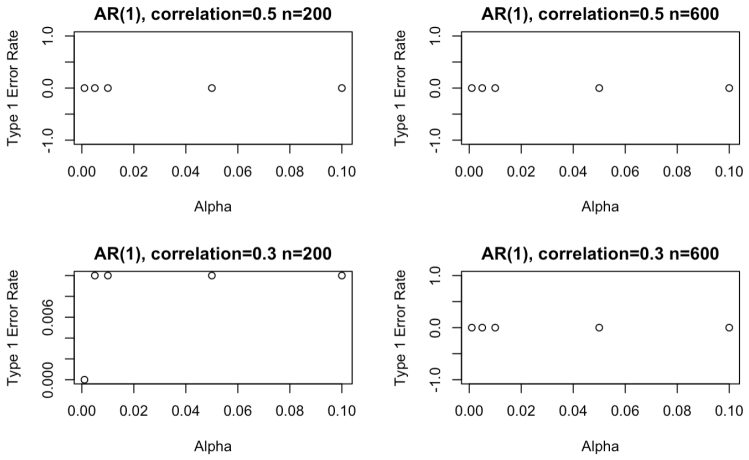


Figure 7: Graph of last four scenarios to calculate the power of a test

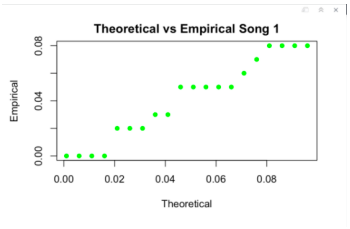


Figure 8: Graph of Type 1 error rates for Random Song 1 with 50 observations

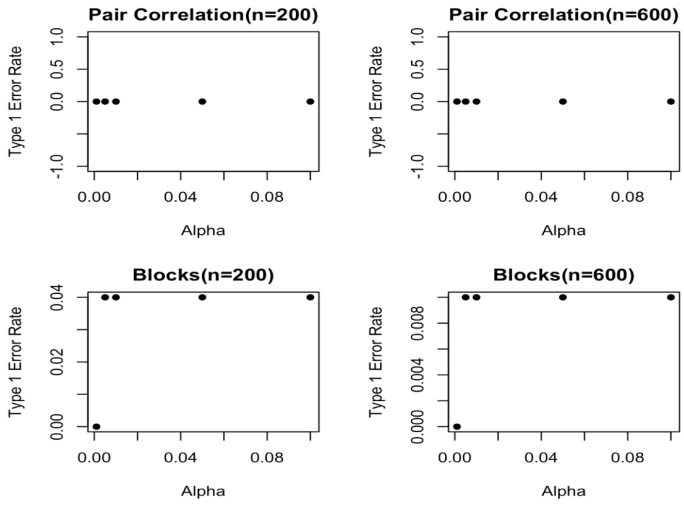


Figure 9: Graph of first four scenarios to calculate the power of a test

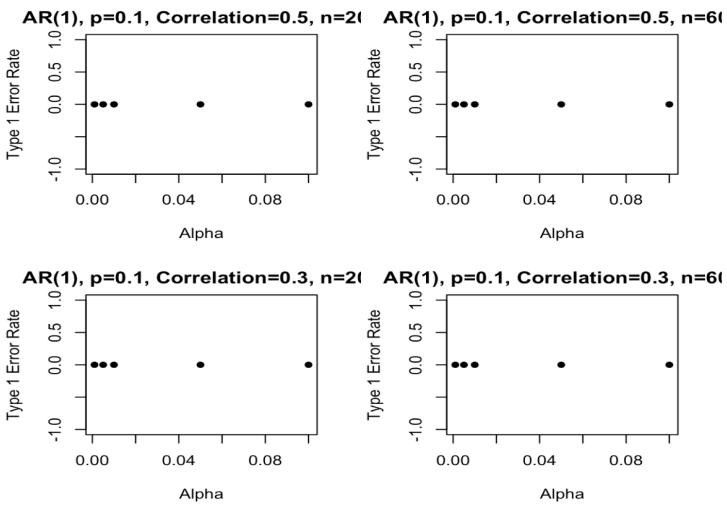


Figure 10: Graph of second four scenarios to calculate the power of a test

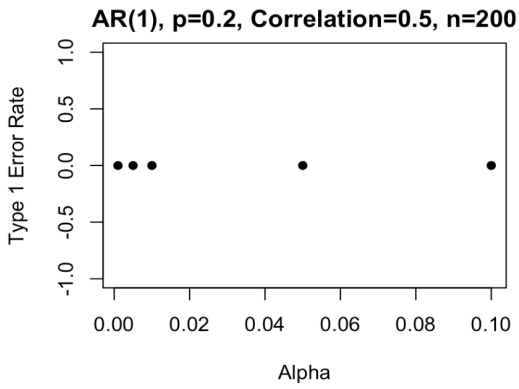


Figure 11: Graph of second to last scenario to calculate the power of a test

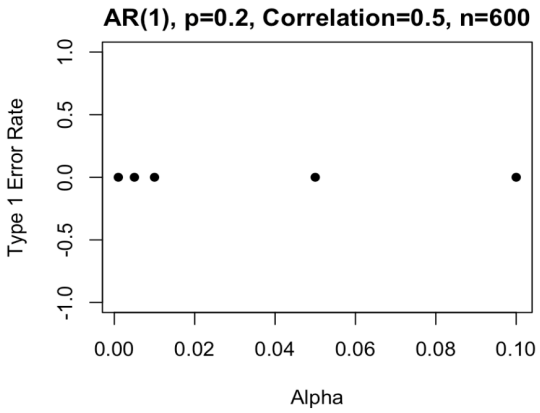


Figure 12: Graph of the last scenario to calculate the power of a test

```
## IMPLEMENTING A PARAMETRIC BOOTSTRAP
# with a Bernoulli distribution
```{r}
parametricBootstrap <- function(v, n) {
  # Initializes a matrix for the sampled data
  sampled <- matrix(nrow = n, ncol = length(v))

  # Populates the matrix
  for (j in 1:length(v)) {
    for (i in 1:n) {
      # Performs a Bernoulli distribution to do the resampling of 0s and 1s
      sampled[i, j] <- (sample(rbern(length(v), .50)))[j]
    }
  }

  return(sampled)
}
```
```

Figure 13: Code to perform a Parametric Resampling on the data given

```r
# function of a song (vector, v) and number of samples to compute (n)
perm.resample <- function(v,n) {

  # creating an empty matrix of number of samples (rows) by the number of notes (cols)
  shuffled <- matrix(nrow=n, ncol=length(v))

  # this loop fills each row with a resampling (done without replacment) of the given song
  for (j in 1:length(v))
    for (i in 1:n) {
    shuffled[i,j] <- (sample(v, size = length(v), replace = FALSE))[j]
    }

  # returns matrix where each row is a sample song
  return(shuffled)
}
```

Figure 14: Code to perform Permutation Resampling on a vector of a song

```
bootstrap <- function(x, nboot, n){
  bootstraps <- c()

  for (i in 1:nboot) {
  bsample <- sample(x, n, replace = TRUE)
   bootstraps <- c(median(bsample), bootstraps)
}

boot.sa <- matrix(sample(x, size = n * nboot,
replace = TRUE), nboot, n)
}
```

Figure 15: Code to perform Nonparametric Bootstrap Resampling on a vector of a song

```r
numHits <- function(v) {
  # set number of hits to 0
  count.hit <- 0

  # setting up parameters
  hits <- c()
  j  <- 1
  distance <- c()

  # this loop creates a count of the number of hits between two misses
  for(i in 1:length(v)) {

    # if a hit, count increases
    if(v[i] == 0) {
      count.hit <- count.hit + 1
      hits[i] <- count.hit
    }
    else
    {
      # if a miss, count is marked as 0
      count.hit <- 0
      hits[i] <- count.hit
    }

    # this loop computes the distance of the count vector
    for (j in 1:length(v)-1) {
      if (isTRUE(hits[i] == 0 & hits[i+j] == 0)) {
        distance[i] <- hits[i+j-1]
        j <- j+1
        } else {
          j<- j+1
        }
      }
  }
  # removes any distances that may be NA
  distance <- distance[!is.na(distance)]

  # removes distances that are 0
  distance <- distance[distance!=0]

  # this is for songs that start with a hit, considers the distances after the first note
  if (v[1] == 0) {
    distance <- distance[2:length(song.a)]
    distance <- distance[!is.na(distance)]
  } else {
    distance <- distance
  }
  return(distance)
}
```

Figure 16: Code to compute the distances on a given vector of a song

```r
# function to compute distances on each row of resampled matrix, m
samples.numHits <- function(m) {

  # empty list
  numHits.vals <- vector(mode = "list", length = nrow(m))

  # this loop computes the distances function on each row and stores them in a list
    for (i in 1:nrow(m)) {
    numHits.vals[[i]] <- numHits(m[i,])
    }
  return(numHits.vals)
}
```

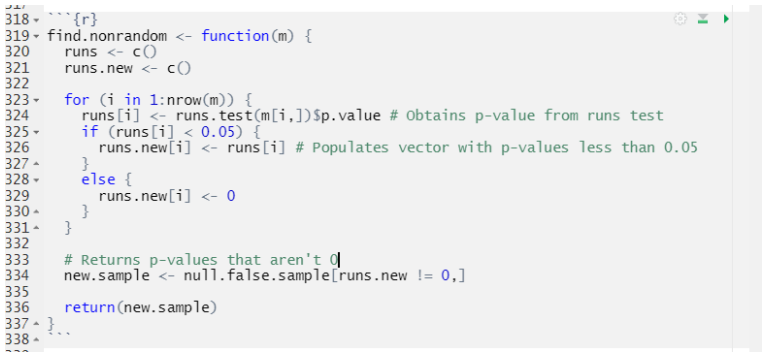
Figure 17: Code to compute the distances on a given matrix of songs (all samples)

```r
# function to compute runs.test on distances
# snpar is package with the runs test function
library(snpar)

# takes in the list of distances
runs.test.numHits <- function(list) {
  # reduces list to a vector
  distances <- Reduce(c,list)

  # computes runs test on the vector
  test <- runs.test(distances)

  # pvalue of the test
  pval <- test$p.value

  # statistic of the runs test -- gives the number of runs
  stat <- test$statistic

  # returns pvalue and number of runs
  return(c(pval, stat))
}
```

Figure 18: Code to compute the runs test on all samples

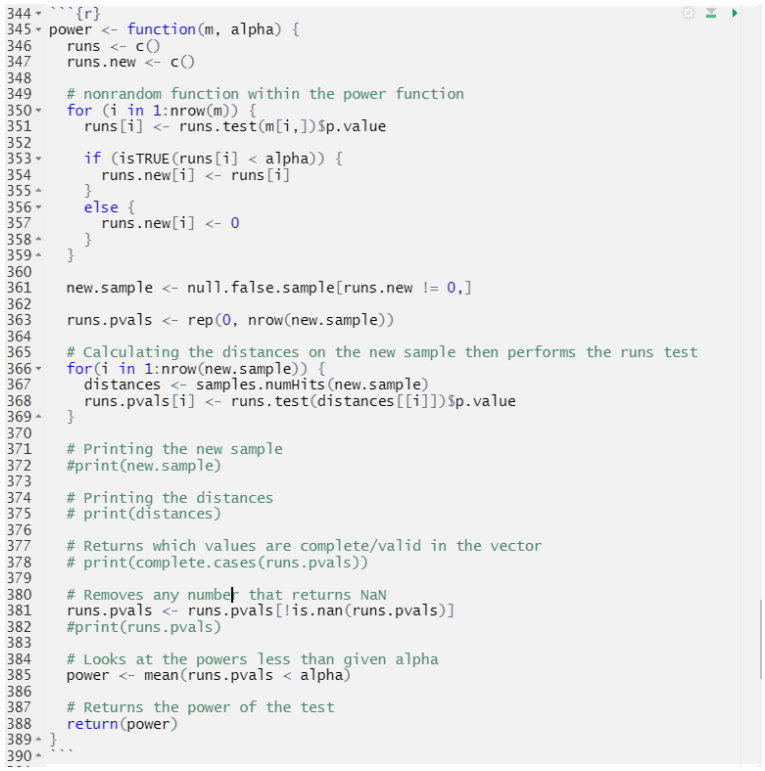
```r
# Getting the Type 1 Error of the bootstrap
error.runs <- function(song, alpha) {

  runs_pvals <- rep(0, nrow(song))

  for (i in 1:100) {
    # Compute the distances on each row
    distances <- samples.numHits(song)

    # Populates the vector with the p-values obtained by the runs test on the distances
    runs_pvals[i] <- runs.test(distances[[i]])$p.value
  }

  rej.rate <- mean(runs_pvals < alpha)

  return(rej.rate)
}
```

Figure 19: Code to get the Type 1 error of the inputted song



```r
find.nonrandom <- function(m) {
  runs <- c()
  runs.new <- c()

  for (i in 1:nrow(m)) {
    runs[i] <- runs.test(m[i,])$p.value # Obtains p-value from runs test
    if (runs[i] < 0.05) {
      runs.new[i] <- runs[i] # Populates vector with p-values less than 0.05
    }
    else {
      runs.new[i] <- 0
    }
  }

  # Returns p-values that aren't 0
  new.sample <- null.false.sample[runs.new != 0,]

  return(new.sample)
}
```

Figure 20: Code finds the songs that are not random within the matrix



```r
power <- function(m, alpha) {
  runs <- c()
  runs.new <- c()

  # nonrandom function within the power function
  for (i in 1:nrow(m)) {
    runs[i] <- runs.test(m[i,])$p.value

    if (isTRUE(runs[i] < alpha)) {
      runs.new[i] <- runs[i]
    }
    else {
      runs.new[i] <- 0
    }
  }

  new.sample <- null.false.sample[runs.new != 0,]

  runs.pvals <- rep(0, nrow(new.sample))

  # Calculating the distances on the new sample then performs the runs test
  for(i in 1:nrow(new.sample)) {
    distances <- samples.numHits(new.sample)
    runs.pvals[i] <- runs.test(distances[[i]])$p.value
  }

  # Printing the new sample
  #print(new.sample)

  # Printing the distances
  # print(distances)

  # Returns which values are complete/valid in the vector
  # print(complete.cases(runs.pvals))

  # Removes any number that returns NaN
  runs.pvals <- runs.pvals[!is.nan(runs.pvals)]
  #print(runs.pvals)

  # Looks at the powers less than given alpha
  power <- mean(runs.pvals < alpha)

  # Returns the power of the test
  return(power)
}
```

Figure 21: Calculates the power of the test being performed on the song

7.1 Brianna Cirillo

This course has truly taught me a lot about R. I have coded things in this class that I could have never imagined doing in my life. I have learned a lot about the way that I should code as well. I always coded the same way that I do math, I sit down and just push through it. I have learned that it is more of a process. You need to start and when you get stuck give yourself a break and come back to it with a fresh mind. I have seen that research is hard work and there is a lot that goes into it. There is so much background that needs to be done before getting to any actual statistic or computing. I have also realized that getting it right doesn't really exist, and as someone great has said plenty of time, "celebrate the little victories". It is most definitely a process. I was very surprised at how much I knew about statistics and R. I definitely downplay my knowledge and this course showed me that I know more than I think. The impact it has made on my future is that I probably will not go in the research direction. This class gave me a lot of anxiety about if I was on the right path or doing the right thing, which goes back to me doubting my skills and knowledge.

This course taught me that I definitely appreciate either complete guidance or complete control. I like to be able to either just complete an assignment or figure out what I need to do and do it the way that most makes sense in my head. I have realized that when I am extremely frustrated I tend to walk away and come back to it later. But, the stress about not knowing how to fix it makes me think about it constantly until I figure it out. I definitely doubt myself more than I should, and when I am extremely stressed I doubt myself more. I think I work well with others, I love that we can throw ideas off of each other. Since this class was pretty diverse in major and skill level, I think we all brought something different to the table. The group part of this class was so great, because it allowed us to grow together.

To make this course better next time, I think we should go over the things we were reading about as a class. Most of the time, I felt like I understood but not completely. This definitely showed when we started doing the project and it was time to apply everything. I also think it may be beneficial to do the readings and then work on that part of the project. I felt a little confused on some topics when we started doing them, just because it had been a while since we learned about it. I also think it would be beneficial if the layout was similar to consulting. If we had one assignment due every work that we got to work on for one class period, and then present the next class time.

Overall, this class was an amazing experience and truly taught me a lot. It was a lot of work, but so worth it in the end. I definitely feel like I have grown as a person and a statistician.

7.2 Samantha Colucci

This course was extremely beneficial for me in many different aspects. Prior to this semester I had little background in advanced statistics, only having taken ProbStats II and having some practice from Consulting last semester. However, this class I learned a lot about the methodology behind statistical methods including how they are formed and how to form one on our own. I learned that when researching new methods, I work best with examples that are well explained and examples that I can recreate on my own. I also learned that it is necessary to learn a new topic one step at a time, rather than all at once, and to start with simpler examples. Though we ran into a lot of issues, I learned that this was okay and that it was actually a part of the process, regardless of how frustrating it can be. I was surprised with how much I actually enjoyed this process of trial and error because of how frustrating it was to me. However, once I accepted that I might not be able to figure everything out all at once, it became a more enjoyable process that almost became like a game. Due to this new found interest I will definitely want to experience more of this and will hopefully be able to do this more in my future. This course taught me a lot about how I learn and what works best for me when learning topics from readings and code. I feel that with a good combination of the two, I am able to grasp a new topic fairly well, and I can fully grasp it with some examples. A major thing I learned about myself is that I have a need to figure things out, and when I can't I usually won't stop working at it until I can. This happened many times during this research when I was trying to figure out how to write a new code or how to modify our method. I also learned that I enjoy working with other people because I like bouncing ideas off of others and hearing their ideas to see if it changes mine in any way. My group was a great to have when doing this project because everyone was helpful and was able to contribute a lot to the research. I also feel that me and my group both had a growth of self-confidence when dealing with this research, because as the semester continued we, most of the time, had a better and better grasp on what we needed to do. To improve the course, I would suggest discussing in detail the readings and assignments due in the beginning of the semester. I feel

that if we had more of a discussion and went over the topics as a group the ideas would have stuck in my head better. I also feel that for some of the assignments for the research we needed more time to complete them and ask questions about what needed to be done. Other than that, I think this course was formatted really well and I loved how it was setup compared to other classes. This has definitely been one of my favorite classes that I have taken at Monmouth and I am grateful to have had this experience.

7.3 Shannon Coyle

From this course, I've definitely gained more knowledge about R Studio and how to perform functions, equations, and other features along with new statistics skills. In previous courses, I understood the math behind confidence intervals and p-values, but this course truly helped me understand the application and importance of those two components. It was different learning statistics to apply it to actual data as opposed to just doing the math for homework and tests to get a grade. When researching, I learned that ideas are not going to work 100% of the time and that is okay, trial-and-error is an integral part of research that everyone has to go through. I was also able to learn more from my peers than from the professor because we have a different understanding of topics than a professor does. I can confidently say that I have a better understanding of statistics after getting explanations from my peers instead of being taught on a whiteboard by someone who has somewhat mastered the topic. Taking this class has made me more interested in the data science aspect of computer science, whereas, before this course I was on the fence about the crossover between coding and mathematics. For my future, this course has given me another coding language that I can say that I understand as well as showing me the importance of data and computing.

I definitely learned a lot about myself by taking this course - both good and bad attributes. My patience could use some work when it comes to coding, when things would not work the way I wanted to it made me feel like a failure and I would give up. I did learn that not even the best coders can get these things right the first time. I also realized that I can work in groups and with whatever I do in my future, I will have to work with others in the technological world. It was helpful to have Bri and Sam pick me up when I was down and help me understand any of the information that I was struggling to grasp. They truly helped me with my stress and frustrations, but aside from that, I would take breaks if I found myself getting overly

frustrated and remind myself that we were trying to do something that most people will not even attempt in their lifetime. I was definitely surprised about how I was able to pick up some of the math concepts and then try to figure them out in R. In the beginning of this class, I was convinced I should not be in it because I felt dumber than everyone else. After this semester, I've realized that if I should not have been in the class, Dr. B would not have asked me to take it. I've seen a lot of personal growth this semester, and I am really glad I was able to take this class.

For the future of this course, I would have mini lessons about the homeworks we did in the beginning of the semester. I was struggling to fully understand the material and I think having small lessons led by Dr. B would have helped me fully get the topic at hand. I think the diaries are really important to the course, so I would keep those as well as presenting the homeworks to everyone else in the class. I think that giving both assignments for the week at the same time would be beneficial if the days between classes aren't that spread out. One of the hardest parts was finding time to do the second assignment for the week when we only had one full day to complete it. Other than that, I truly loved this course and I am so glad I was able to take it.