STA258 Spotify Songs Data Analysis - Group 10

Tidy Tuesday Project - Spotify

Yash Agarwal, Diaa Bakir, Angelo Gener, Steven Hua, Muhammad Iqbal

Part 1: Variable Exploration

Variable 1: Track Album Release Date

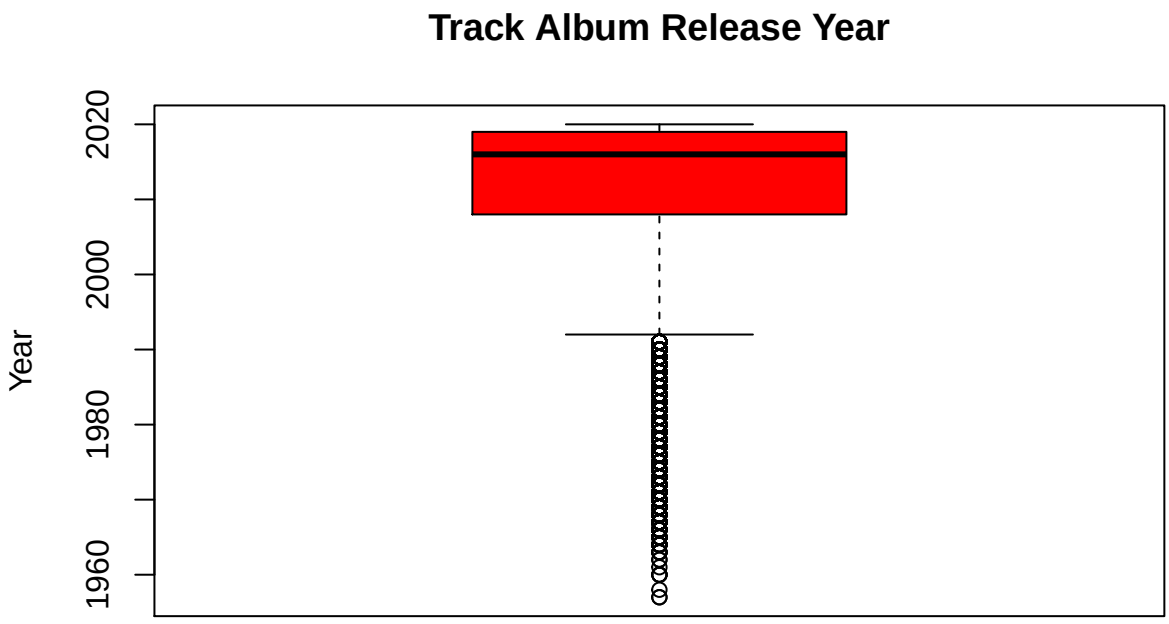
Generating the valid information, the track album variable is given as a list of characters and integers, we must transform it into only years in order to generate reliable data.

```
track_album_release_date <- spotify_songs$track_album_release_date
track_album_release_date_new <- stringr::str_extract(track_album_release_date, "^.{4}")
track_album_release_date_year <- as.numeric(track_album_release_date_new)
```

Now all the information in the list is an integer, we can use the library mosaic to determine an accurate summary

```
favstats(track_album_release_date_year)
```

```
##   min   Q1 median   Q3  max     mean       sd     n missing
##  1957 2008   2016 2019 2020 2011.137 11.41745 32833       0
```

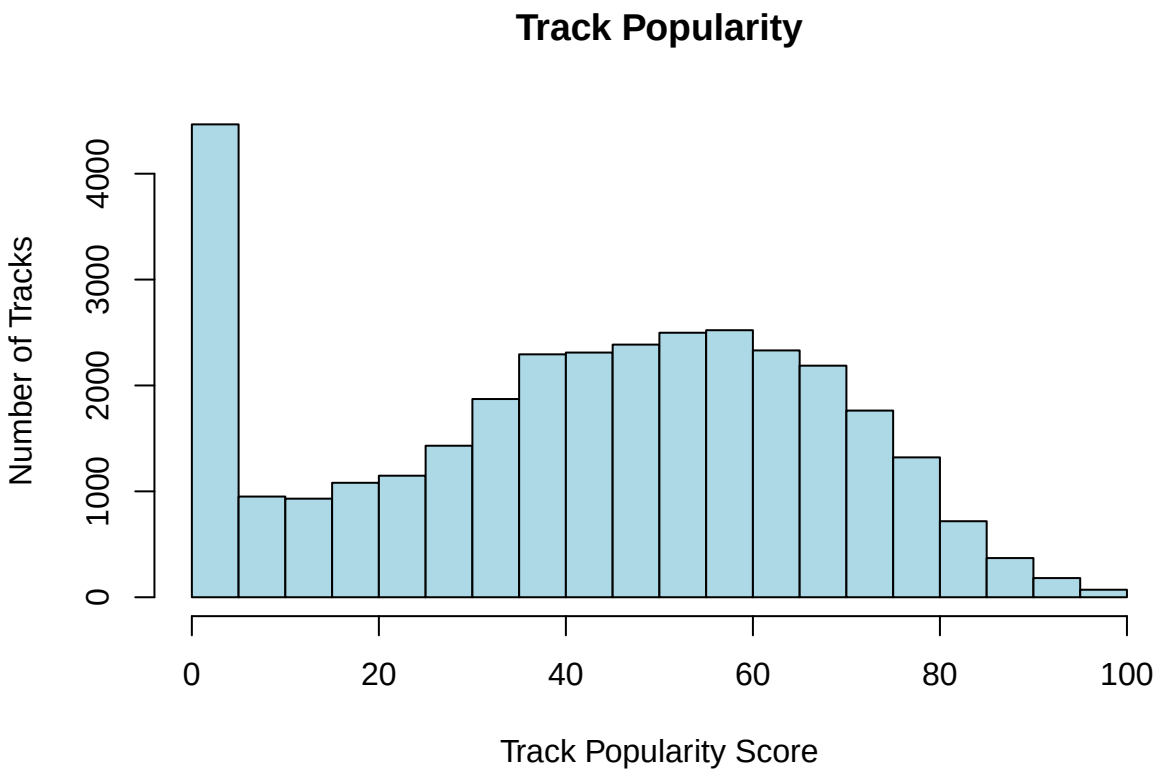


This variable is regarding the year that albums were released on. The mean is 2011.137 for this data set with a standard deviation of 11.41745. This means that the average release year for the albums in the data set is 2011 and there is a variance of roughly 11 years. The oldest release year is 1957 and the most recent album release year is 2020. The IQR is within 2008 and 2019, this is due to the median being 2016 which is the middle value. Anything after 2035.5 and before 1991.5 is considered an outlier, there is no upper whisker outliers but there is a number of lower bound outliers. This implies that the majority of the data used is recent albums and older songs are actually considered outliers according to the IQR range which is illustrated in the graph. As a result, we can imply that this is not normal, this makes sense as we know spotify is a newer app and that it is updated recently.

Variable 2: Song Popularity

```
track_popularity <- spotify_songs$track_popularity
favstats(track_popularity)
```

```
##  min Q1 median Q3 max     mean       sd     n missing
##    0 24     45 62 100 42.47708 24.98407 32833       0
```

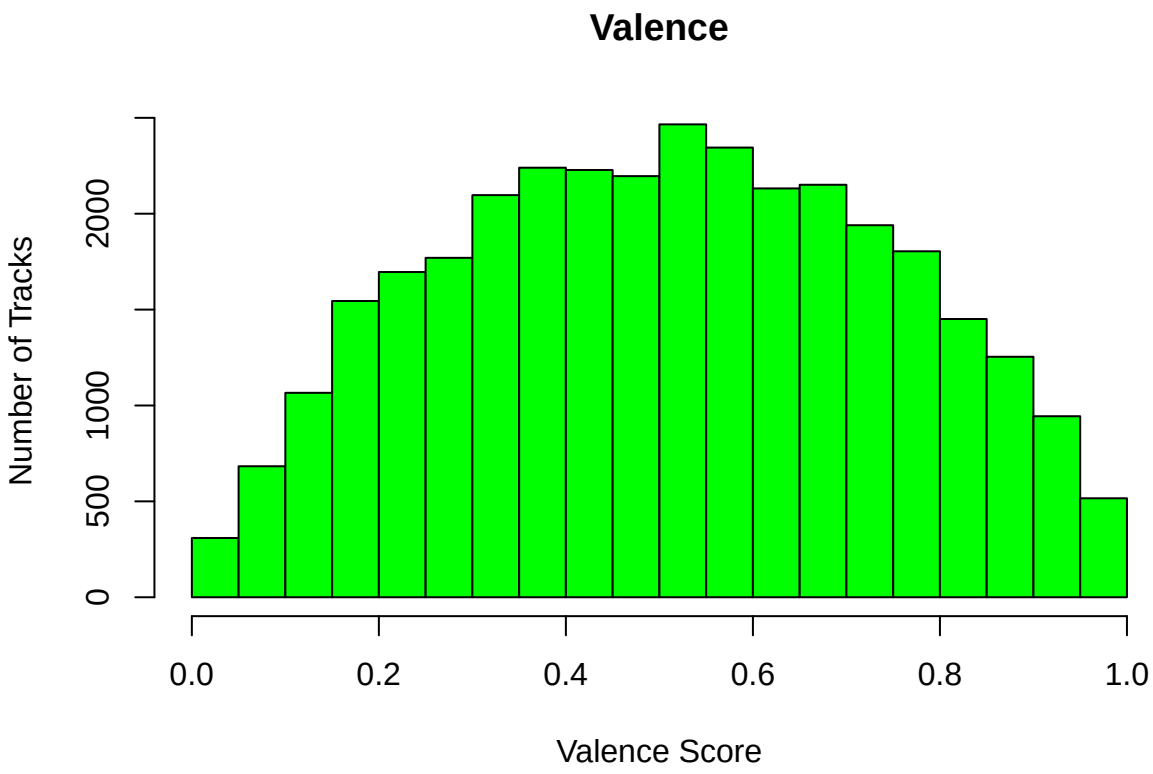


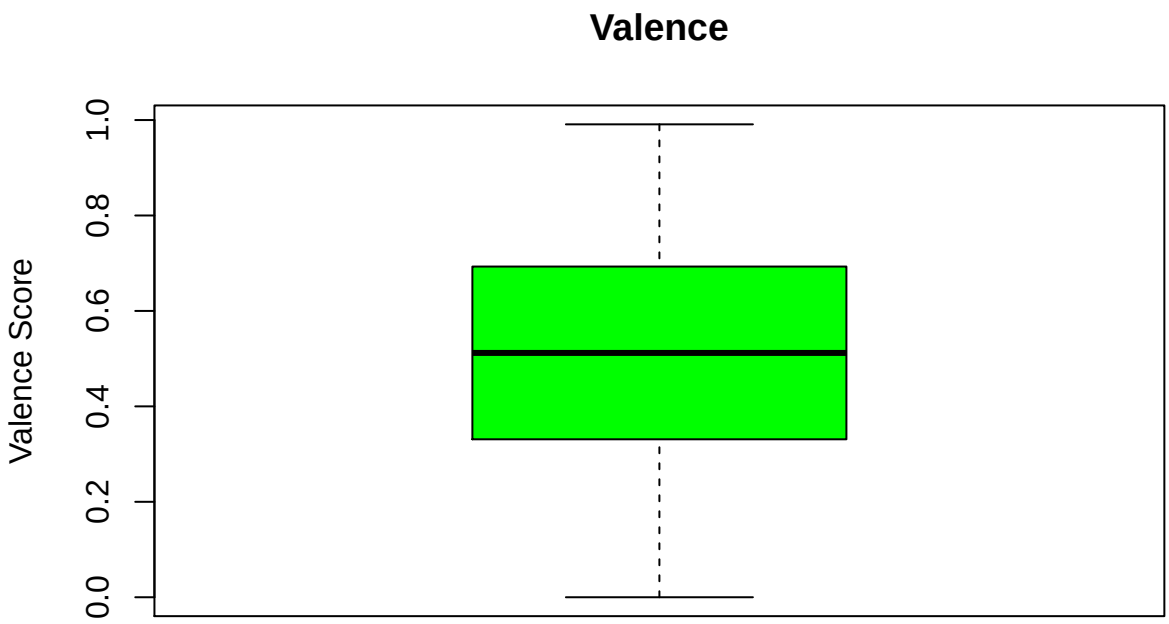
The second variable deals with track popularity, every song in the dataset has been assigned a value from 0 to 100 as an indicator of how popular it is. The average track popularity is 42.47708 while the standard deviation is 24.98407. This indicates that the dataset is about normal but with a high standard deviation value and thus a high variance. The lowest score which is also the most common score, is 0 while there is a few tracks that have received the highest score of 100. The IQR is within 24 and 62 while the median is 45 which is close to the mean value. A song cannot be considered an outlier as the IQR range is 38, the value of the outlier range 57 added or removed from the upper/lower bounds which cannot be given as it does not fall within the range provided. This means that despite having higher or lower cases, the dataset implies that all the values are crucial to making up the bell shaped curve present which is applicable to real life. Many songs cannot be superhits, usually songs are admired by the artist's fans and in rare cases listeners outside the intended audience however the amount of songs that are not popular vastly outweigh the number of incredibly popular songs.

Variable 3: Valence

```
valence <- spotify_songs$valence
favstats(valence)
```

```
##  min    Q1 median    Q3   max     mean       sd     n missing
##    0 0.331  0.512 0.693 0.991 0.510561 0.233146 32833       0
```


Valence
Number of Tracks
0
500
1000
2000
0.0
0.2
0.4
0.6
0.8
1.0
Valence Score

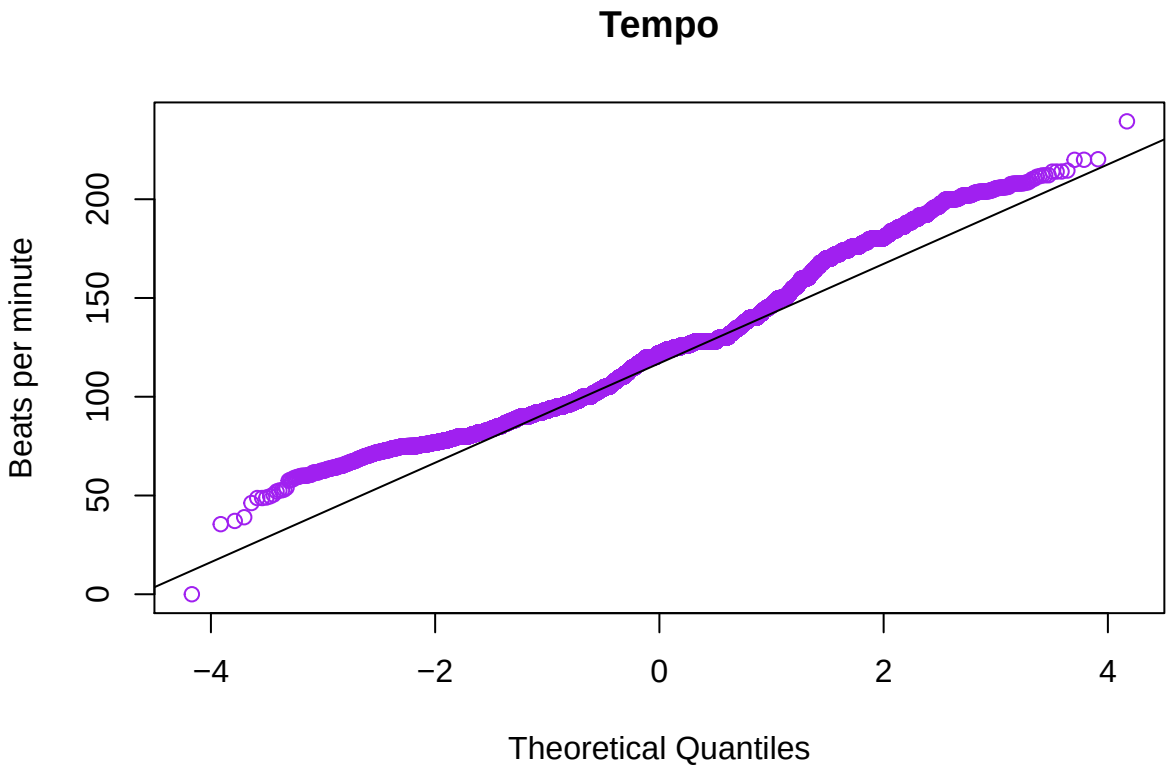


The valence variable refers to the musical mood or positivity present in a song, songs range from 0.0 to 1.0. A high valence score track is more happy in cheerful while a low valence score track is depressing and gloomy. The median for this data set is 0.510561 while there is a standard deviation of 0.233146. The highest score a track recieved was 0.991 and the lowest was 0.331. There is a stark contrast between the highest and lowest and the hypothetical highest and lowest score, there were no songs that radiated positivity and definitely no songs that was entirely negative. This matches our thinking as no one would listen to a song that is overly positive or overly negative. This is reflected by the IQR bound values and the median which are 0.69, 0.331 and 0.512 respectively. This curve is incredibly normal and this is shown through the graphs as well, the boxplot is centered in y axis and the histogram is bell curved. This is an accurate representation as music is a very powerful medium that allow for both feelings of hope and negativity in order to convey the true and complex meanings that the artists intend.

Variable 4: Tempo

```
tempo <- spotify_songs$tempo
favstats(tempo)
```

```
##  min    Q1  median      Q3    max     mean       sd     n missing
##    0 99.96 121.984 133.918 239.44 120.8811 26.90362 32833       0
```

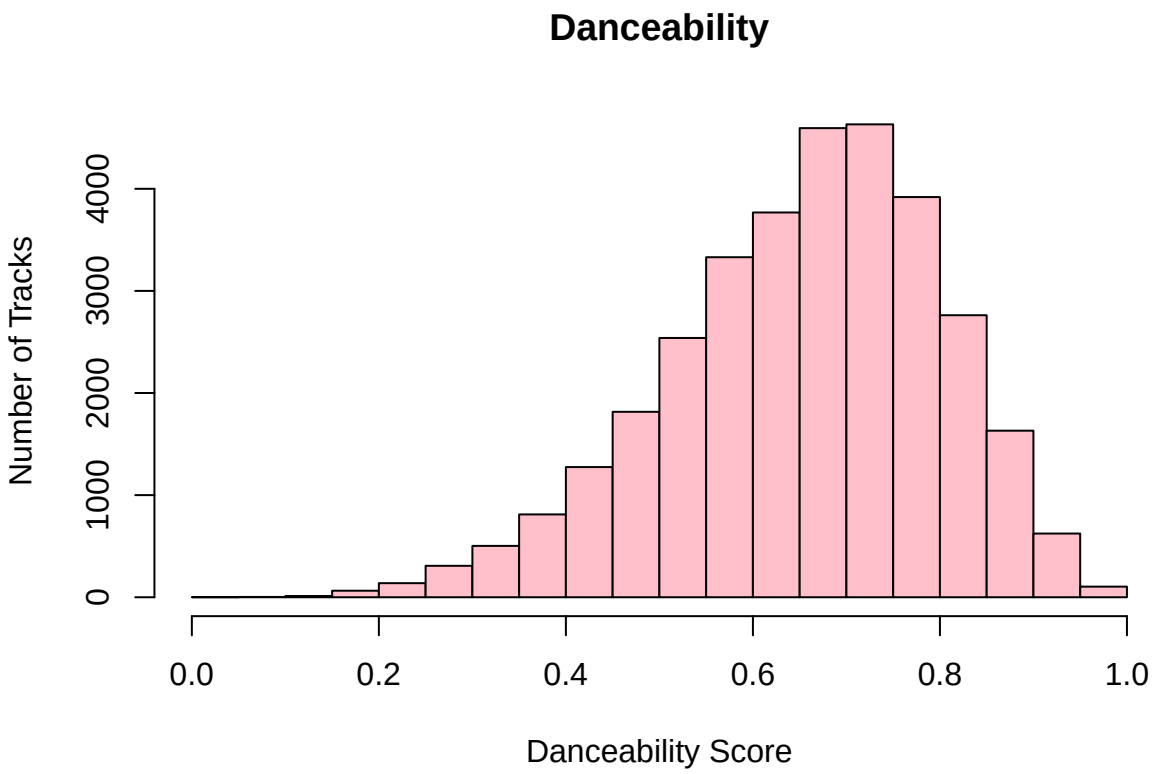


Tempo is the overall beats per minute within a track. It is the speed or pace in a song and it is calculated from the average beat duration. The data suggests the tempo mean 120.8811 and a standard deviation of 26.90362. This is quite a fast beat pace but modern music is known as being faster paced through new styles of old genres coming into the mainstream media such as EDM, hip hop, pop and rap. This data uses all of those genres and analyzes them which is a reason for the median being similar to the mean at 121.984. The lower bound is 99.96 and the upper bound is 133.918. There are a few outliers that is lower than under 49.023 and higher than 184.855. The reason for the outliers being present in this variable is because of the wide array of bpm songs that are created by different genres, for example, a reggae song would have a slower bpm than an EDM song and this is expected as it is not confined to a score value and is a freeflowing variable that depends on the artists. The data also suggests this by being very close to a normal distribution and loosely straying from it.

Variable 5: Danceability

```
danceability <- spotify_songs$danceability
favstats(danceability)
```

```
##  min    Q1 median    Q3   max      mean        sd     n missing
##    0 0.563  0.672 0.761 0.983 0.6548495 0.1450853 32833       0
```

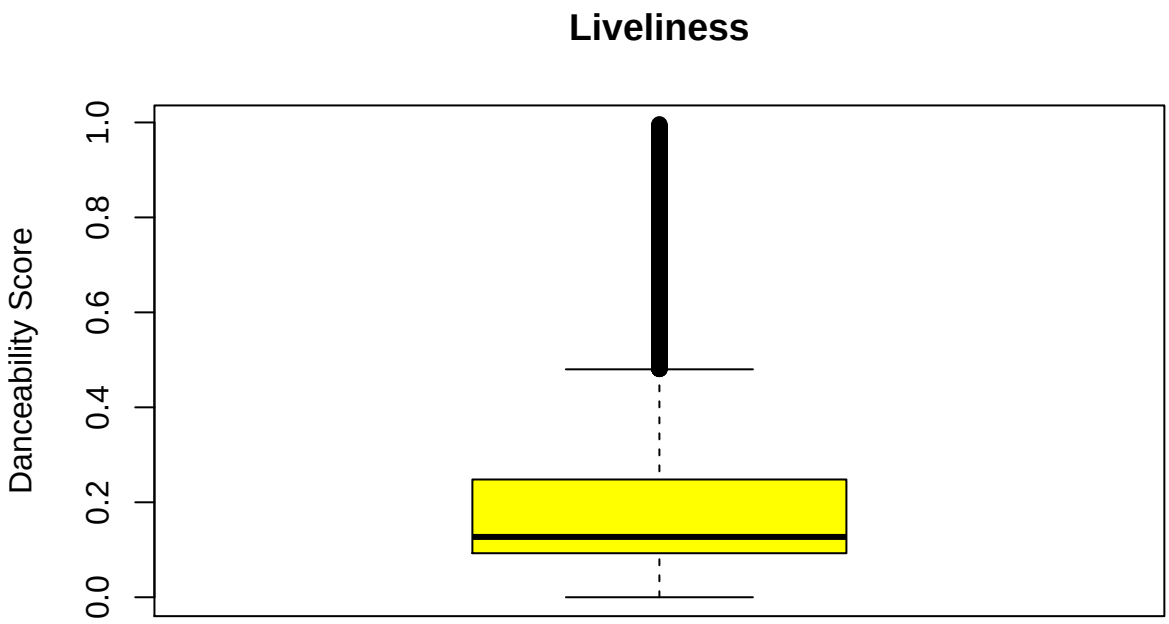


Danceability is a rating from 0.0 to 1.0 on how danceable a song is. A high score indicates that people are more likely to dance to it whereas songs with a low score means that there is a low chance someone will dance to it. The mean for this variable is 0.6548495 with a standard deviation of 0.1450853. The lowest score a song recieved was 0 and the highest was 0.983. The IQR ranges from 0.563 to 0.761 with a median of 0.672. There can be no upper outliers as it would need a score of 1.058 which is not possible, however, lower outliers exist and a track with a score of 0.267 or lower is considered an outlier. This matches our real life expectation as most songs are enjoyable and encourages a positive mood change, however there are certain songs such as purely instrumental songs that do not make you want to dance. Additionally, most recent songs are often catchy and thus are more danceable which explains the slight skewness in the graph.

Variable 6: Liveliness

```
liveliness <- spotify_songs$liveness
favstats(liveliness)
```

```
##  min     Q1 median    Q3   max      mean        sd     n missing
##    0 0.0927  0.127 0.248 0.996 0.1901762 0.1543173 32833       0
```

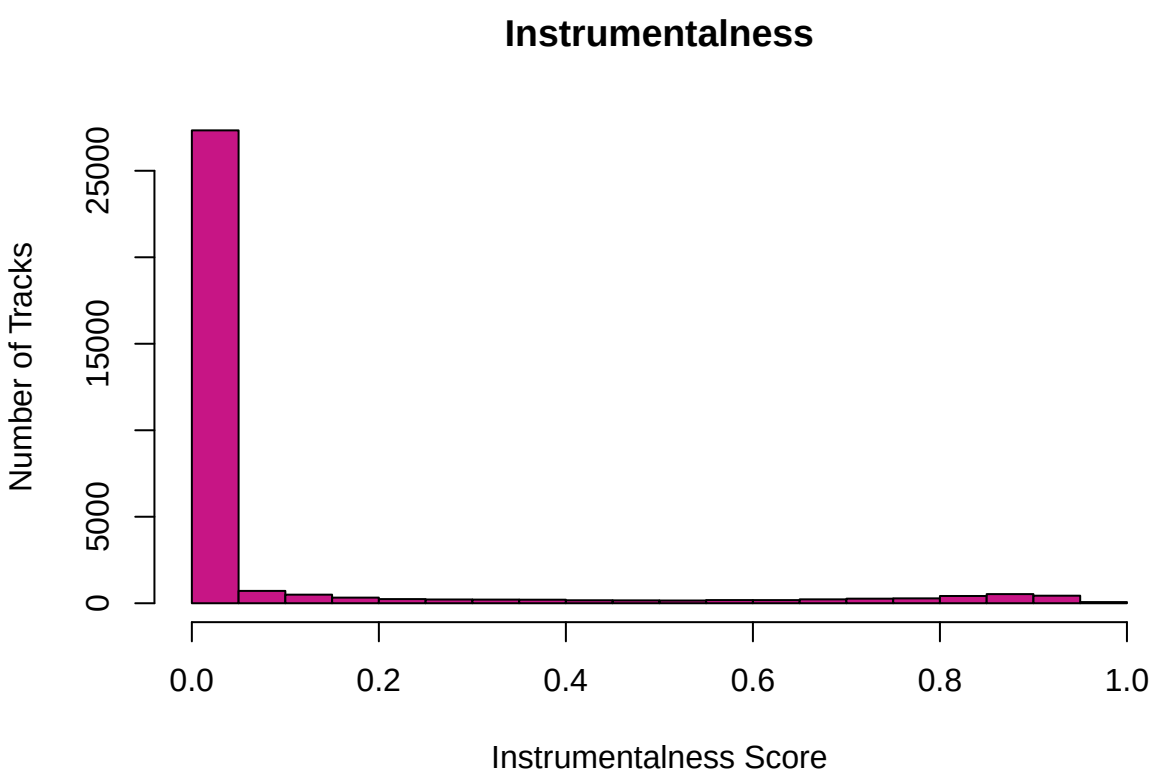


Liveliness is an interesting category that looks at the presence of an audience within a track. A higher liveliness value indicates a higher probability of a crowd and the song being live and a low score indicates that the song was not performed live, a song with a 0.8 score or higher indicates an extremely high likelihood that it was performed live. The mean for this variable is 0.1901762 while there is a standard deviation of 0.1543173. The lowest score a track received was a 0 while the highest was 0.996 so the range varies drastically. The Interquartile Range spans from 0.0927 to 0.248 with a median of 0.127, this is expected as the dataset is extremely right skewed. This indicates most of the songs were not performed live but were performed in a studio, usually this is standard practice since most songs are sung and altered in studios prior to release and then sung live in person during tours or concerts. The boxplot also indicates this as there are many upper bound outliers with scores higher than a 0.48095. The boxplot graphic also matches this as the median is quite low within the IQR range which also implies the dataset is right skewed.

Variable 7: Instrumentalness

```
instrumentalness <- spotify_songs$instrumentalness
favstats(instrumentalness)
```

```
##  min Q1   median      Q3   max       mean        sd     n missing
##    0  0 1.61e-05 0.00483 0.994 0.08474716 0.2242301 32833       0
```

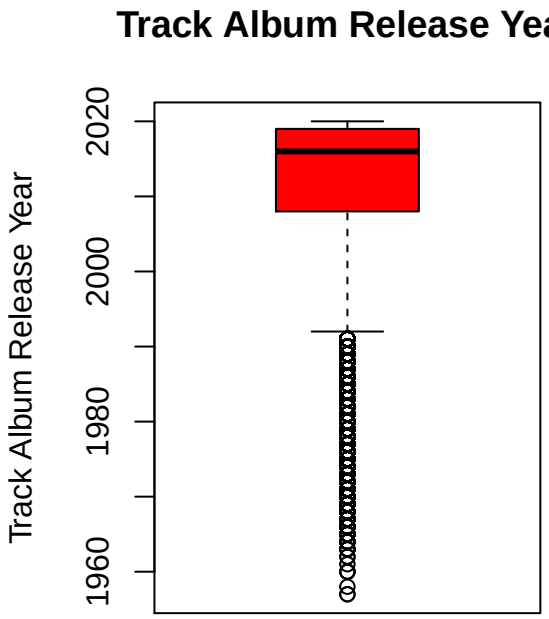


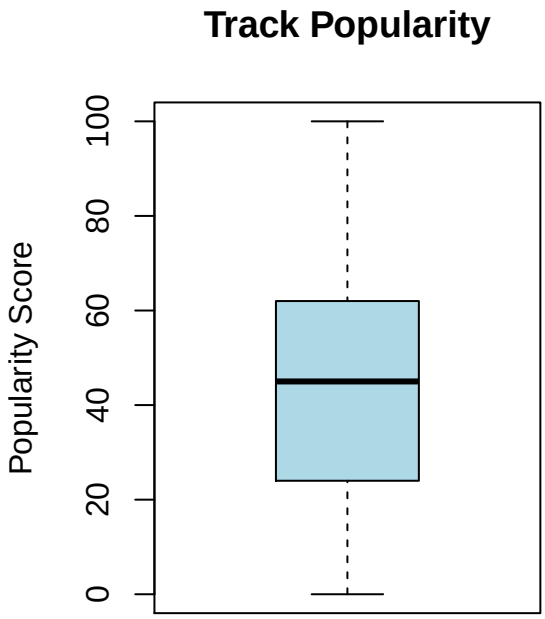
Instrumentalness is a score that reflects the number of vocals, it only counts words that are rapped or spoken and does not count adlibs such as "ooh" and "ah". The higher the score, the higher the probability that the song contains no words or vocals. For songs with a score of over 0.5, are tracks that are meant to represent instrumental tracks. 0.08474716 was the mean for this data and the standard deviation was 0.2242301. The highest and lowest scores were 0 and 0.994 respectively which means that the range of the dataset is fairly spread out. Something interesting that occurs is that the lower bound for the IQR starts at 0 and the upper bound ends at 0.00483 which is an extremely low range. This is expected but a lot of the data leaning so heavily against instrumentalness implies that most songs contain a high about of vocals as most people do not like to listen to instrumental music often. Anything above a score of 0.012075 is considered highly instrumental and an outlier as it is unnatural. For songs in the dataset to have such a low instrumental score indicates that the data is heavily right skewed as it favors more vocal tracks. This can be within the graph, although there is a high number of instrumental tracks it is because of the sheer size of the data set containing 32833 songs. The vast majority of tracks range very close to if not a 0 whereas the the number of tracks that have a high instrumental score is considerably less with them all having under an estimated 1000 tracks at each score.

Part 2: Exploring relationships amongst chosen variables

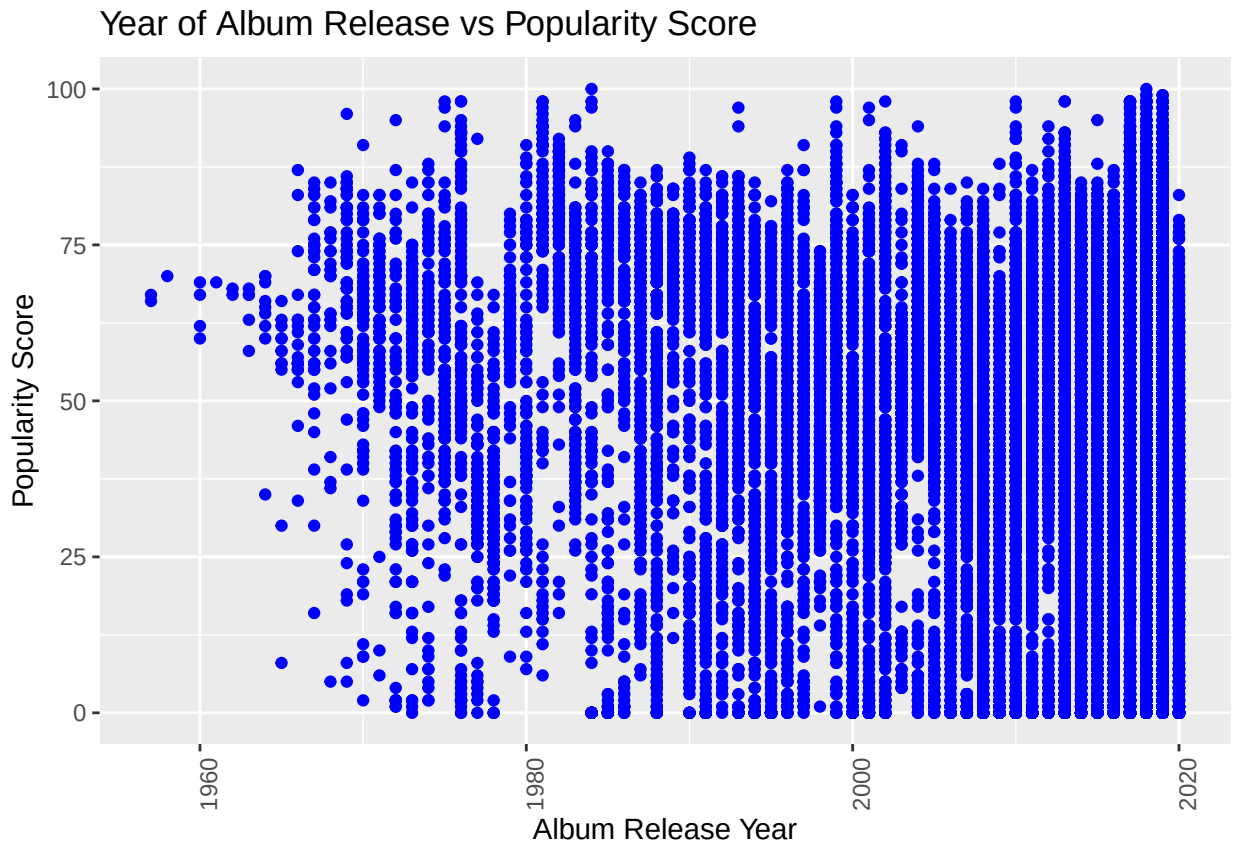
Year of Album Release vs Popularity Score

```
par(mfrow=c(1,2))
boxplot(track_album_release_date_year, main = "Track Album Release Year", ylab = "Track Album Release Y
boxplot(track_popularity, main = "Track Popularity", ylab = "Popularity Score", col="light blue")
```

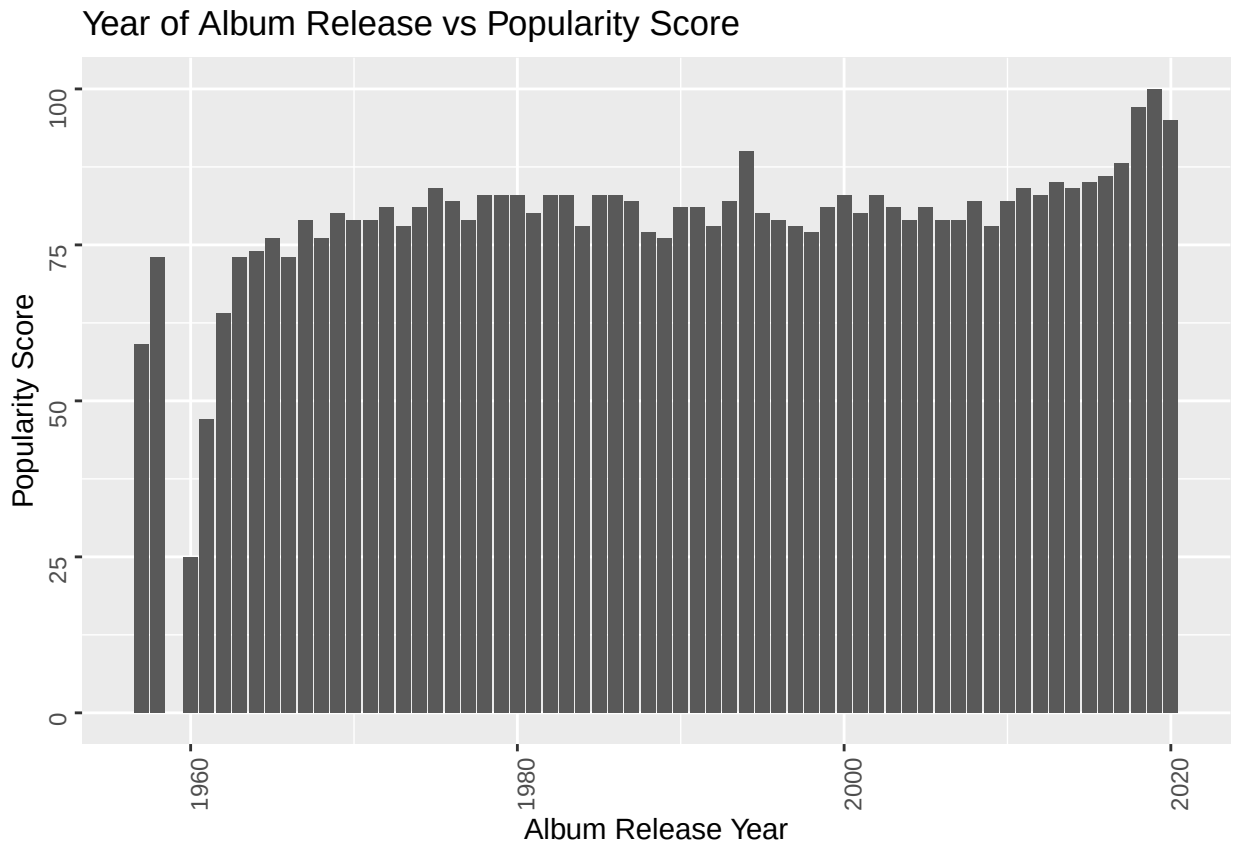





```
track_album_release_date_year_sorted <- sort(track_album_release_date_year)
ggplot(spotify_songs, aes(x=track_album_release_date_year_sorted,
                          track_popularity), y=track_popularity) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  geom_point(col = "blue")+
  ggtitle("Year of Album Release vs Popularity Score")+
  xlab("Album Release Year")+
  ylab("Popularity Score")
```




```
ggplot(spotify_songs, aes(y=track_popularity,
                          x=track_album_release_date_year)) +
  geom_bar(position="dodge", stat="identity") +
  theme(axis.text = element_text(angle=90, hjust=1)) +
  ggtitle("Year of Album Release vs Popularity Score") +
  xlab("Album Release Year") + ylab("Popularity Score")
```




```
favstats(track_album_release_date_year)
```

```
##   min   Q1 median   Q3  max     mean       sd     n missing
##  1957 2008   2016 2019 2020 2011.137 11.41745 32833       0
```

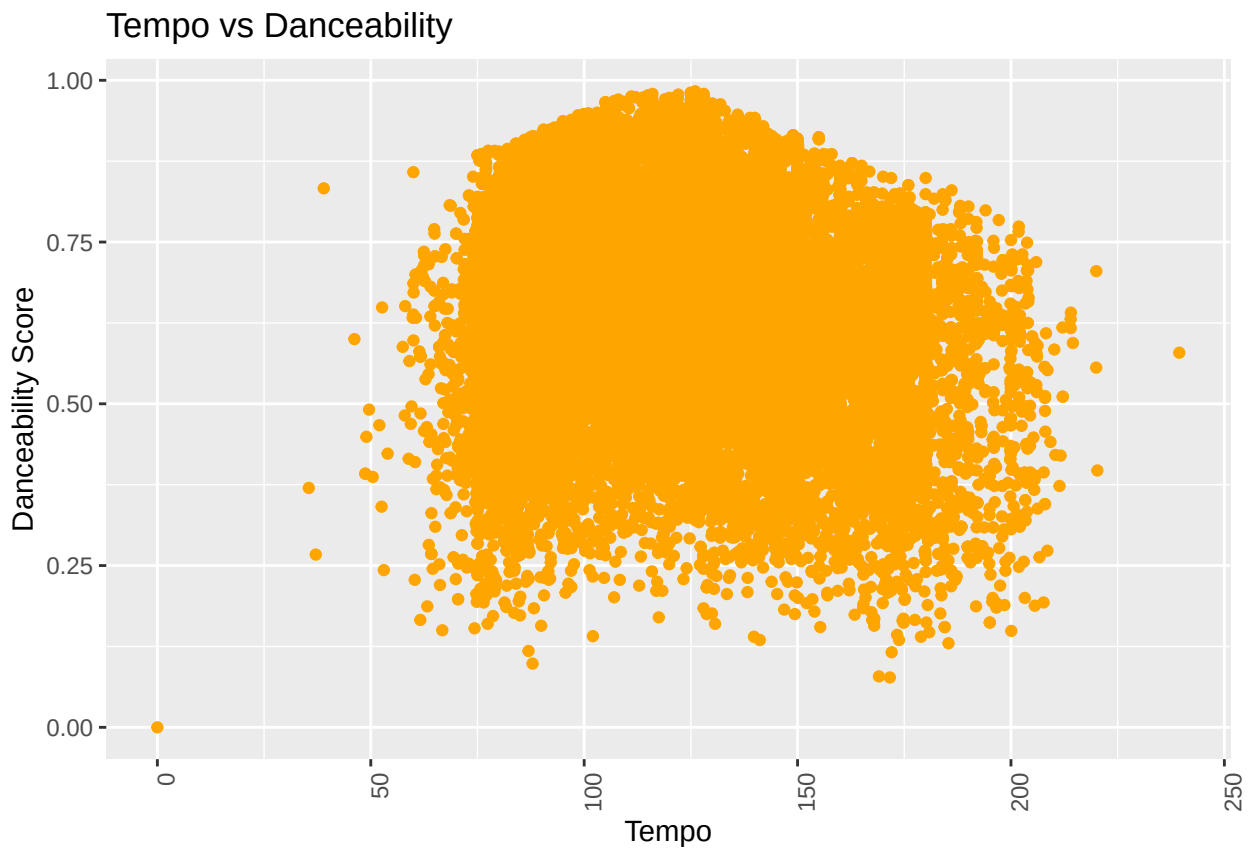
```
favstats(track_popularity)
```

```
##  min Q1 median Q3 max     mean       sd     n missing
##    0 24     45 62 100 42.47708 24.98407 32833       0
```

There appears to be a strong positive relationship between the track album release year and the popularity score of those tracks. As many songs have been released in recent years, it did correlate to current songs becoming more popular. The chance of the songs becoming popular as time progresses is linear, popularity scores for tracks do increase as the years progress. The covariance between the two variables is 17.32757, it suggests that there is a strong positive relationship present. As the years increase, there are more and more albums that receive higher popularity scores. The histogram suggests that any release before 1991.5 is an outlier, it seems as if more scores have recieved a wide variety of scores as indicated by the mean for track popularity. However it seems as if the peaks and concentrations are higher and more dense within the recent years, this is also demonstrated by the bar graph. This relationship suggests that there is a greater likely hood of an album that it will have a song that is popular. An important factor to consider is that spotify itself is fairly new and has features that encourage its users to listen to newer songs through features such as "New Releases", "Made Just For You" & "Discover Weekly" which might influence the a track's popularity score.

Relationship between Tempo & Danceability

```
ggplot(spotify_songs, aes(x=tempo,
                          danceability), y=danceability) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  geom_point(col = "orange")+
  ggtitle("Tempo vs Danceability")+
  xlab("Tempo")+
  ylab("Danceability Score")
```




```
favstats(tempo)
```

```
##  min    Q1  median      Q3    max     mean       sd     n missing
##    0 99.96 121.984 133.918 239.44 120.8811 26.90362 32833       0
```

```
favstats(danceability)
```

```
##  min    Q1 median    Q3   max      mean        sd     n missing
##    0 0.563  0.672 0.761 0.983 0.6548495 0.1450853 32833       0
```

```
cov(tempo, danceability)
```

```
## [1] -0.7185403
```

As the covariance is -0.7185403, it is evident that the relationship between the tempo and the danceability score have a moderate negative correlation. Additionally, the graph also indicates that there is a negative relationship as when tempo increases to higher values, the danceability scores decrease. In theory, this makes sense as songs with a very high tempo, generally falling into music genres such as hyper-pop and dubstep, are not commonly seen aS songs that individuals in the general population dance to due to its overwhelming fast pace.Furthermore,the graph indicates that songs with a tempo that is closer to the mean of 120 bpm, and median of 121 bpm, have the highest danceability scores which generally fall into music genres such as hip-hop and pop. These music genres seem to be ones that are danced to especially with the emergence of dance trends in popular social media platforms such as Tikok and instagram in today's culture. This relationship suggests that generally as the tempo increases, the danceability of a song will decrease.

Relationship between Tempo and Valence

```
ggplot(spotify_songs, aes(x=valence,
                          tempo), y=tempo) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))+
  geom_point(col = "aquamarine")+
  ggtitle("Tempo vs Valence")+
  xlab("Valence Score")+
  ylab("Tempo")
```


```
favstats(tempo)
```

```
##  min    Q1  median      Q3    max     mean       sd     n missing
```

```
##    0 99.96 121.984 133.918 239.44 120.8811 26.90362 32833       0
```

```
favstats(valence)
```

```
##  min    Q1 median    Q3   max     mean       sd     n missing
##    0 0.331  0.512 0.693 0.991 0.510561 0.233146 32833       0
```

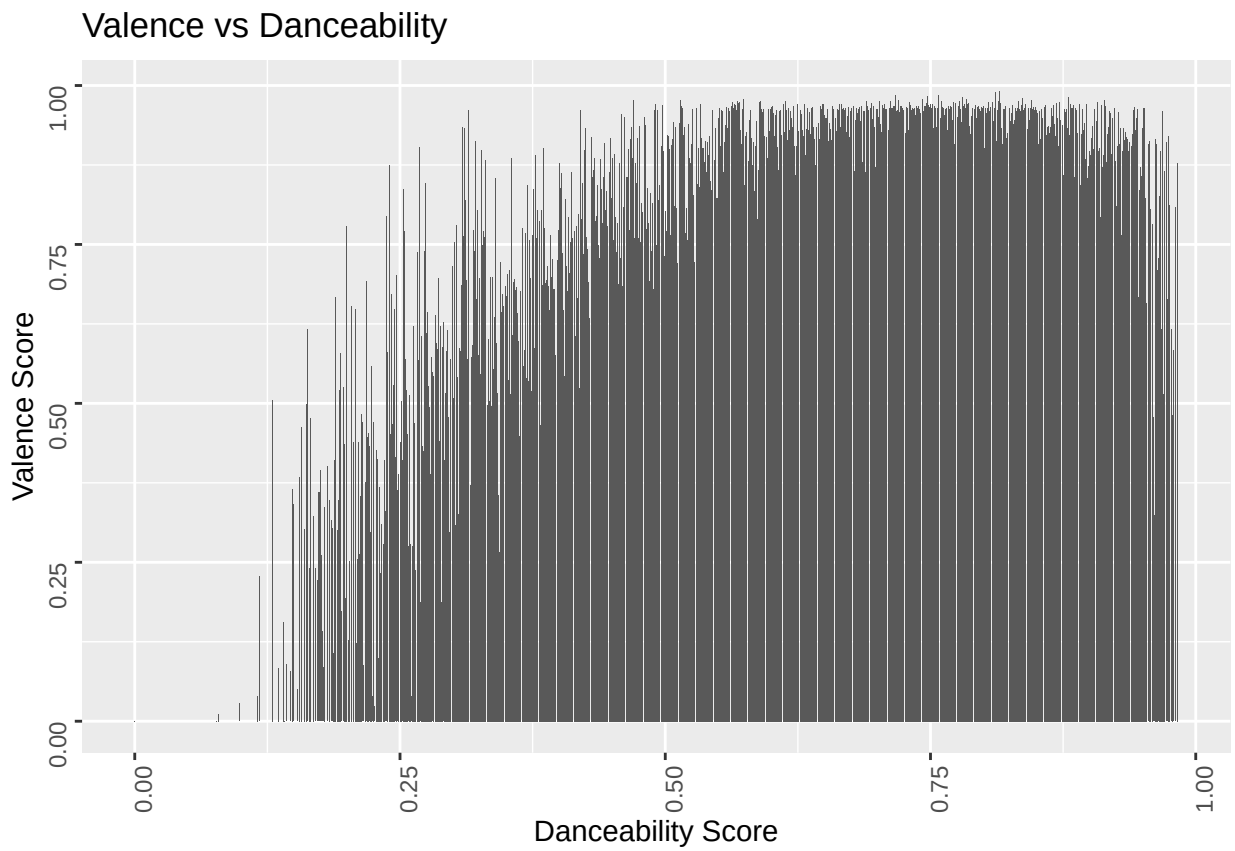
```
cov(tempo, valence)
```

```
## [1] -0.1614042
```

As the covariance is -0.1614042, it is evident that the relationship between the tempo and valence score have a weak correlation or no correlation. Additionally, the graph also indicates that there seems to be a uniform distribution between the two variables. As the valence score represents the musical mood or positivity present in a song, the data can indicate that the tempo of a song does not generally reflect the musical mood or positivity that a song produces.

Relationship between Valence & Danceability

```
ggplot(spotify_songs, aes(y=valence, x=danceability)) +
  geom_bar(position="dodge", stat="identity") +
  theme(axis.text = element_text(angle=90, hjust=1)) +
  ggtitle("Valence vs Danceability") + xlab("Danceability Score") +
  ylab("Valence Score")
```



```
favstats(valence)
```

```
##  min    Q1 median    Q3   max     mean       sd     n missing
##    0 0.331  0.512 0.693 0.991 0.510561 0.233146 32833       0
```

```
favstats(danceability)
```

```
##  min    Q1 median    Q3   max      mean        sd     n missing
##    0 0.563  0.672 0.761 0.983 0.6548495 0.1450853 32833       0
```

```
cov(valence, danceability)
```

```
## [1] 0.0111803
```

We can see from the covariance score of 0.0111803, that the relationship between the valence score and the danceability score have a positive correlation. The means for the two variables are 0.51 and 0.65 respectively and the sd consists of roughly 0.233 and 0.15 so they both can fall into the same mean range. Additionally, the graph also indicates that there is a positive relationship as higher danceability scores indicate higher valence scores. This allows us to conclude that the relationship between these two variables is left skewed. This indicates that if a song is more positive and has a high valence value, it correlates to the danceability score that it has. This matches our hypothesis as we believed that positive songs will make make people want to dance to the track more than their wish to dance to a negative song. The positive correlation is expected and the bar graph shows high concentration for scores when the danceability and valence score are over there mean.

Relationship between Liveliness & Instrumentalness

```
ggplot(spotify_songs, aes(x=liveliness, instrumentalness), y=instrumentalness) +  theme(axis.text.x = el
  geom_point(col = "blue")+
  ggtitle("Liveliness Score vs Instrumentalness Score")+
  xlab("Liveliness Score")+
  ylab("Instrumentalness Score")
```


```
favstats(liveliness)
```

```
##  min     Q1 median    Q3   max      mean        sd     n missing
##    0 0.0927  0.127 0.248 0.996 0.1901762 0.1543173 32833       0
```

```
favstats(instrumentalness)
```

```
##  min Q1   median      Q3   max       mean        sd     n missing
##    0  0 1.61e-05 0.00483 0.994 0.08474716 0.2242301 32833       0
```

```
cov(liveliness, instrumentalness)
```

```
## [1] -0.0001905579
```

As we can see from the correlation value it is a negative relation value exists with a score of -0.0001905579. There is a negative correlation, whenever songs possess a low score of liveliness, they are typically not very instrumental and largely vocal. This shows a negative correlation as there is an extremely high number of cases with a low liveliness score with a low instrumentalness score. Additionally, there are cases with a low liveliness score but with varying instrumentalness scores which shows that tracks are almost always written without the presence of an audience. Interestingly, there is many discrete cases of tracks with not only high instrumental scores but also high liveliness scores. This is an indication of tracks typically having extremely low instrumental and liveliness scores or having high probability of being performed live and containing minimal vocals. This revelation lines up with what was expected in our hypothesis, lower scores of instrumentalness indicates that the track has a high chance of possessing a low liveliness score. The total graph is right skewed as the majority of the dataset has its information on the left side of the visual with minimal amounts of scattered data across the rest of the range.

Part 3: Confidence Intervals Estimations

One Population Mean - Tempo

Tempo is used in order to calculate the population mean with a 95% confidence interval. This is done so as it is a numerical value and it makes more sense in order to calculate either the mean or variance for this variable. In this case, the mean was chosen. This was acquired by getting the mean, standard deviation and the number of data entries in the tempo variable. Standard Error was calculated by dividing the standard deviation by the square root of n. A Z score was calculated by using the 95% confidence interval statistics. Margin of Error was calculated by multiplying the z score by standard error. Finally the bounds were calculated by adding and subtracting the margin of error from the mean.

```
tempo_mean <- mean(tempo)
tempo_sd <- sd(tempo)
tempo_n <- length(tempo)
tempo_SE <- tempo_sd/sqrt(tempo_n)
tempo_zscore <- qnorm(0.975, 0, 1)

tempo_MOE <- tempo_zscore * tempo_SE

tempo_upper_bound <- tempo_mean + tempo_MOE
tempo_lower_bound <- tempo_mean - tempo_MOE

tempo_CI_Interval <- data.frame(tempo_lower_bound, tempo_upper_bound)
tempo_CI_Interval
```

```
##   tempo_lower_bound tempo_upper_bound
## 1          120.5901          121.1721
```

```
tempo_test <- mean(spotify_songs$tempo)
tempo_test
```

```
## [1] 120.8811
```

We are 95% confident that the true mean is in between the lower bound 120.5901 and the upper bound 121.1721. We know this is true by by examining the mean itself which is 120.8811. Thus we can conclude that this confidence interval was accurate and yielded the right result.

One Population Variance - Track Popularity

In order to calculate the population variance through a 95% confidence interval, you need the the values of n and the standard deviation of the dataset. From this, you can obtain the values of the degrees of freedom, s^2, and the upper and lower confidence levels. By using a chi squared distribution, you can obtain the lower and upper chi score. After calculating that the upper and lower bounds can be calculated, those bounds must then be square rooted in order to find the confidence interval of the standard deviation.

```
popularity_n <- length(track_popularity)
popularity_df <- popularity_n - 1
popularity_s = sd(track_popularity)
popularity_s_squared = popularity_s^2
popularity_upper_conf_level <- 0.975
```

```r
popularity_lower_conf_level <- 0.025

# chi squares
popularity_lower_chi_score = qchisq(popularity_lower_conf_level,
                                    popularity_df, lower.tail = FALSE)
popularity_upper_chi_score = qchisq(popularity_upper_conf_level,
                                    popularity_df, lower.tail = FALSE)

popularity_lower_bound = ((popularity_n - 1)*popularity_s_squared)/
  popularity_lower_chi_score
popularity_upper_bound = ((popularity_n - 1)*popularity_s_squared)/
  popularity_upper_chi_score

popularity_standard_lower_bound = sqrt(popularity_lower_bound)
popularity_standard_upper_bound = sqrt(popularity_upper_bound)

popularity_CI = data.frame(popularity_standard_lower_bound,
                           popularity_upper_bound)
popularity_CI
```

```
##   popularity_standard_lower_bound popularity_upper_bound
## 1                        24.79444               633.8638
```

```r
popularity_test <- mean(spotify_songs$track_popularity)
popularity_test
```

```
## [1] 42.47708
```

We can be 95% confident that the true standard deviation of the popularity score falls between the lower bound 24.79444 and the upper bound 633.8638. We can confirm that this is true because the mean is 42.47708.

Difference Between Two Means - Instrumentalness & Liveliness

For this confidence interval, let instrumentalness be group 1 and let liveliness be group 2. In order to arrive to the conclusion, we must get the variables for n, sd and mean for both groups. After this the z score must be got through the percent confidence interval. The formula to get the upper bounds and lower bounds is then used in order to generate the mean difference confidence interval bounds.

```r
instrumental_n = length(instrumentalness)
instrumental_sd = sd(instrumentalness)
instrumental_mean = mean(instrumentalness)

liveliness_n = length(liveliness)
liveliness_sd = sd(liveliness)
liveliness_mean = mean(liveliness)

z_score <- qnorm(0.975, 0, 1)

mean_diff = instrumental_mean - liveliness_mean
pooled_variance_numerator_term1 = instrumental_sd^2 * instrumental_n-1
```

```
pooled_variance_numerator_term2 = liveliness_n^2 * liveliness_n-1
pooled_variance = pooled_variance_numerator_term1+
  pooled_variance_numerator_term2/(instrumental_n+liveliness_n)


upper_bound = mean_diff + (z_score * sqrt(1/instrumental_n+1/liveliness_n))
lower_bound = mean_diff - (z_score * sqrt(1/instrumental_n+1/liveliness_n))

mean_difference_interval = data.frame(lower_bound, upper_bound)
mean_difference_interval
```

```
##   lower_bound upper_bound
## 1  -0.1207261 -0.09013198
```

```
mean_test <- instrumental_mean - liveliness_mean
mean_test
```

```
## [1] -0.105429
```

We can conclude there is a 95% possibility that the difference between the means is between the lower bound -0.1207261 and -0.09013198. This is the case if group 1 is instrumentalness and group 2 is liveliness. This is true since the calculated difference between the mean falls in the range as it is 0.105429.

Ratio of the two population variances - Valence & Danceability

Valence is group 1, danceability is group 2. We calculated the values of n, sd and the standard deviation squared. After that we used the values of the confidence intervals and n to generate the values required for the formula. The upper bound and lower bound were then calculated.

```
valence_n = length(valence)
valence_sd = sd(valence)
valence_s_squared = (valence_sd^2)

danceability_n = length(danceability)
danceability_sd = sd(danceability)
danceability_s_squared = (danceability_sd^2)

denom1 = qf(0.025, valence_n-1, danceability_n-1)
denom2 = qf(0.975, valence_n-1, danceability_n-1)

upper_bound = valence_s_squared/(danceability_s_squared*denom1)
lower_bound = valence_s_squared/(danceability_s_squared*denom2)

CI = data.frame(lower_bound, upper_bound)
CI
```

```
##   lower_bound upper_bound
## 1    2.527047    2.638788
```

```
test = valence_sd - danceability_sd
test
```

```
## [1] 0.08806065
```

We can conclude there is a 95% possibility that the difference in variance is between 2.527047 and 2.638788. This is only the case if valence is group 1 and danceability is group 2.