Classification of Spotify Data

DATA 606 - W2023 Final Project

Kane Smith, Rodrigo Rosales Alvarez, Arhur Trim, Jordan Keelan, Scott Bennett

2023-02-17

Contents

1 Introduction

1.1 Background

Spotify is the world's most popular subscription service for audio streaming. Spotify claims 489 million users, of which 205 million users are Spotify Premium subscribers(1). Spotify has an extensive library of music tracks and gathers data about the music in order to better recommend songs to users, and makes this data available through a web API(2). By analyzing this data, we hope to gain insights into how characteristics of different songs and the artists who created them affect the popularity of the songs. We also aim to explore other trends and relationships within the data, such as whether we can predict the genre of a song based on characteristics such as loudness and musical key. This analysis could potentially be used to identify what factors make songs popular and help artists create music that will be commercially successful (if that is their goal). It may also expose other findings that would be interesting to a general audience of music consumers.

1.2 Problem & Topic Importance

The music industry is in a constant state of evolution, and the popularity of a song can play a significant role in an artist's success. With an abundance of music being produced and released, it can be challenging to predict which songs will achieve popularity. Considering this environment, there is a growing need for a more scientifically informed understanding of the factors that contribute to a song's popularity. Although virality cannot always be predicted, it can be influenced, and record labels and streaming services can benefit greatly from affiliation with viral hits. By understanding the factors that contribute to popularity and identifying them in new songs and artists, they can make informed decisions on music production and marketing strategies.

1.3 Data Source

For this project, we used a Kaggle dataset that offers consolidated data from the Spotify web APIs(3). The dataset is structured into two data tables, provided as CSV files. These two data tables are "Tracks" and "Artists"

The "Tracks" "csv" contains information for approximately 600,000 musical tracks available on Spotify. Features include "popularity" as well as a multitude of attributes to describe the character of the music itself i.e, "loudness" score and "danceability" score. The "Artists" csv contains additional data, specifically about the artist such as the list of genre's associated with that artist.

Usage of the dataset is governed by the Community Data License Agreement, which grants: "... a worldwide, non-exclusive, irrevocable (except as provided in Section 5) right to: (a) Use Data; and (b) Publish Data." (4)

1.4 Summary of Variables

One of the key variables of interest/response variables for our project is *popularity*. This is a score given to a track from 0-100, with the most popular track being given a score of 100. For some parts of the analysis, the *popularity* variable was used to classify each song as a "hit" or not. For the purposes of this project, a hit was considered a track in the upper quartile of *popularity*.

The independent variables available in the "tracks" data are:

1. **duration_ms** - the length of the track in ms
2. **explicit** - explicit lyrics
3. **artists** - artist names

4. **danceability** - score for how suited a track is for dancing, 0.0-1.0.
5. **energy** - score for how energetic a track is perceived, 0.0-1.0.
6. **key** - maps Pitch class notation (E.g. 0 = C, 1 = C sharp/D flat, 2 = D, and so on.)
7. **loudness** - decibel loudness of the track range from -60 to 0 dB
8. **mode** - modality of the track (0 is minor, 1 is major)
9. **speechiness** - score for how speech-like a track is, 0.0-1.0. Values close to one indicate something like a podcast (high speechiness).
10. **acousticness** - range of whether a track is acoustic (0.0-1.0)
11. **instrumentalness** - range of whether a track is instrumental (0.0-1.0)
12. **liveness** - range representing audience sounds in the track (0.0-1.0)
13. **valence** - represents how 'happy' a track is (0.0-1.0)
14. **tempo** - the temp of the track in beats per minute
15. **time_signature** - time signature of the track 3-7 (3 represents 3/4 time etc.)

```
artists_df <- data.frame(read.csv("../spotify_dataset/artists.csv"))
tracks_df <- data.frame(read.csv("../spotify_dataset/tracks.csv"))
```

1.5 Problem Statement

The music industry is in a constant state of evolution, and the popularity of a song can play a significant role in an artist's success. With an abundance of music being produced and released, it can be challenging to predict which songs will achieve popularity. Considering this environment, there is a growing need for a more scientifically informed understanding of the factors that contribute to a song's popularity. Although virality cannot always be predicted, it can be influenced, and record labels and streaming services can benefit greatly from affiliation with viral hits. By understanding the factors that contribute to popularity and identifying them in new songs and artists, they can make informed decisions on music production and marketing strategies.

"Data is becoming a primary way for labels and other tastemakers to find their next stars. Shav Garg is the co-founder of Indify, a company he calls a "music data platform." Music pros use his company to figure out who the next hot artists are, and they were extremely early in noticing artists like Khalid, who the company first featured in the fall of 2015, nearly a year and a half before his debut album." (SETARO, S.)

As data becomes an increasingly crucial tool in the music industry, many music producers and musicians are adopting a formulaic approach to music design. By breaking down successful or unsuccessful songs into their fundamental elements and analyzing patterns influencing listener emotions, they can leverage these findings in the creation of novel music. Understanding these song features and how they contribute to popularity can help artists stay ahead of musical trends and create more impactful music.

This project's purpose is to analyze the features of songs (as listed in the "Summary of Variables" section above) and to develop multiple statistical models to determine the relevance of these features to a song's popularity. This analysis will provide valuable insights into the music industry and inform artists, music producers, and industry professionals on how to make informed decisions to increase the popularity of their music.

2 Preliminary Analyses

2.1 Data Cleaning

The dataset was already very clean when it was downloaded, but we still did validation to ensure that everything was good before we began an exploratory analysis and modeling. We started by looking for counts of NAs and duplicate values which there were none. Next, we checked the dimensions of our data frame; there are 22 variables and 529,958 rows.

The most important part of cleaning the data was changing variables that were meant to be factors (explicit, key, mode, time_signature) as factors instead of integers. The other important was to normalize our continuous features using mean normalization to potentially improve the performance of our models and to allow us to use linear discriminant analysis.

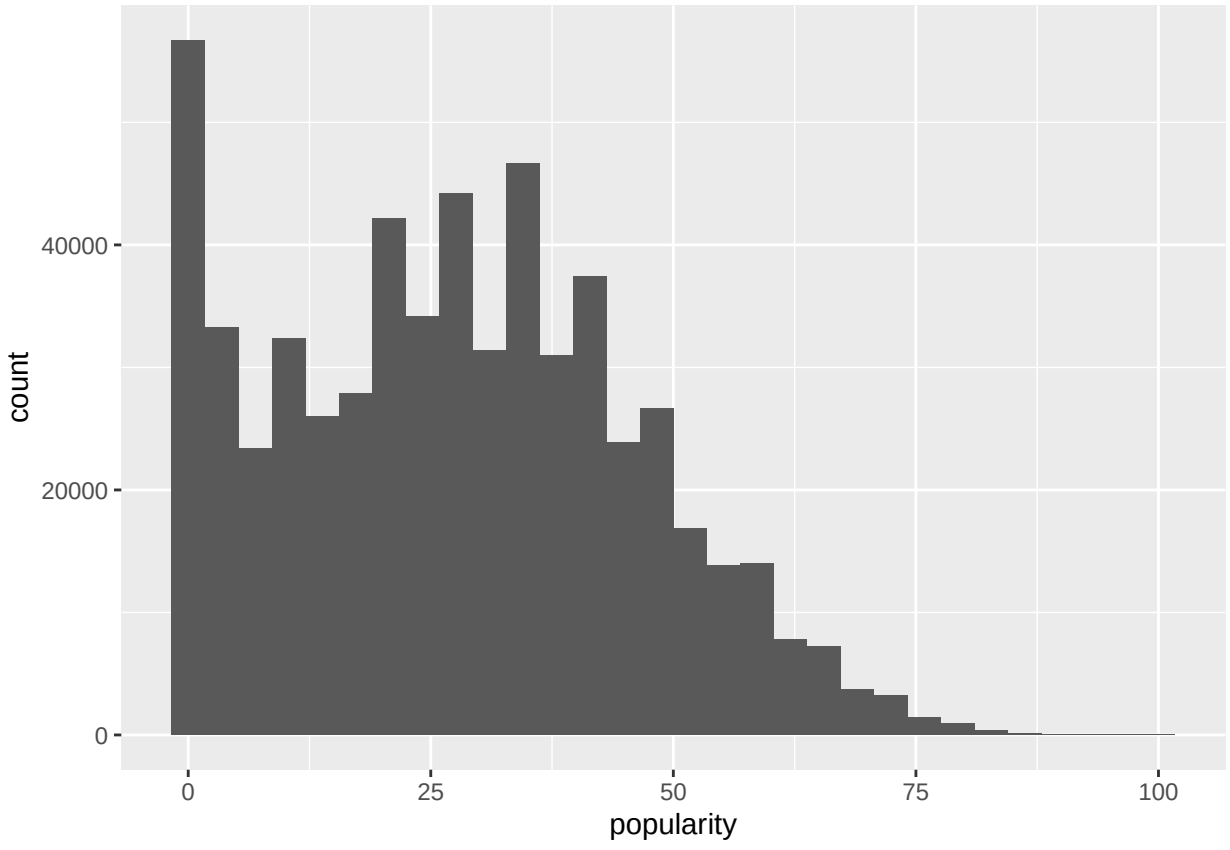
2.2 Visual Investigation

bar-plot

histogram

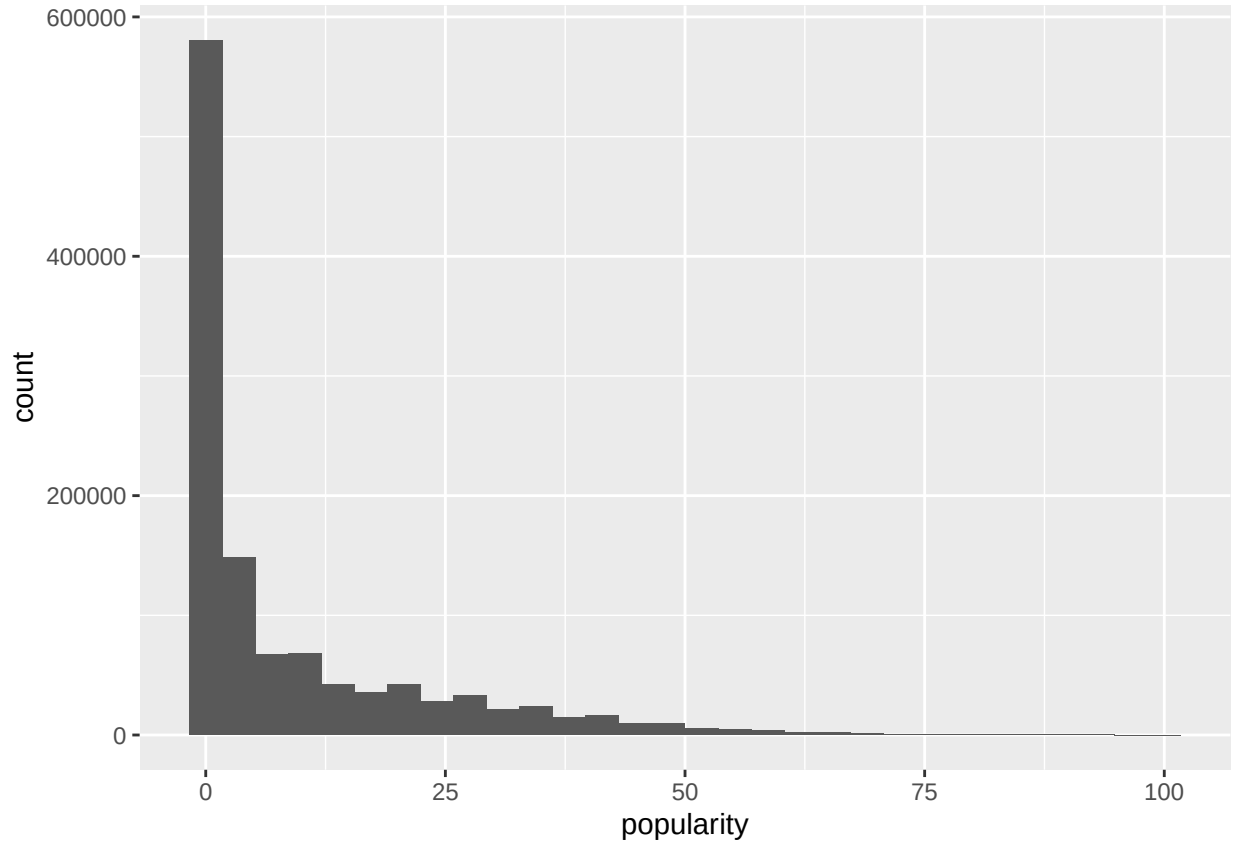
```
# head(artists_df) somthing here has characters that latex cant print
# head(tracks_df)

tracks_pop <- ggplot(data=tracks_df, aes(x=popularity), title="Full Tracks Dataset Popularity Histogram
artists_pop <- ggplot(data=artists_df, aes(x=popularity), title="Full Artists Dataset Popularity Histog

tracks_pop
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
artists_pop
```

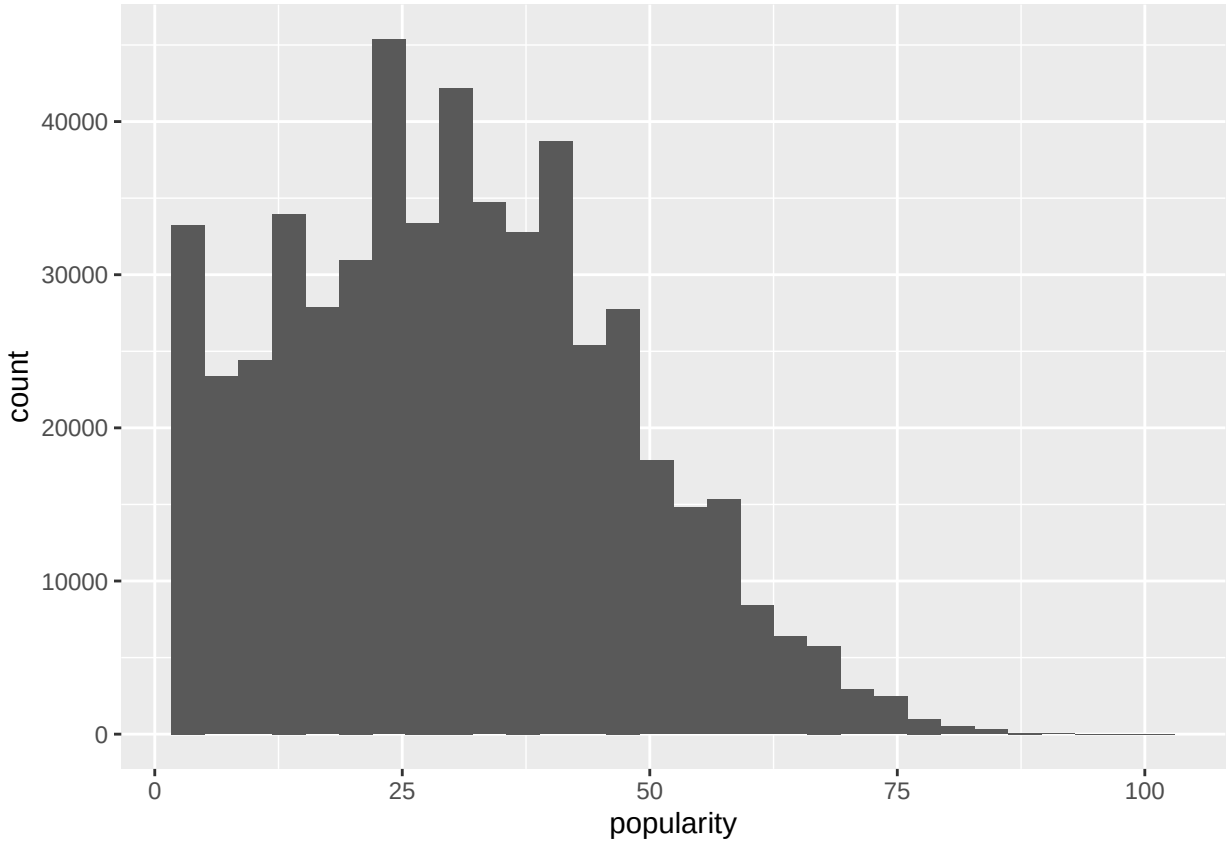
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# Filter out tracks with popularity less than 1
popular_df <- tracks_df %>% filter(popularity > 1)

# normalize Data
popular_labels <- popular_df[, c("id", "name", "artists", "id_artists", "release_date")]
popular_factors <- popular_df[, c("explicit", "key", "mode", "time_signature")]
popular_numeric <- popular_df[, c("duration_ms","popularity", "danceability", "energy", "loudness", "sp
popular_scaled <- scale(popular_numeric)
popular_factors <- lapply(popular_factors,factor)
popular_df2 <- data.frame(popular_labels, popular_factors, popular_scaled)
```

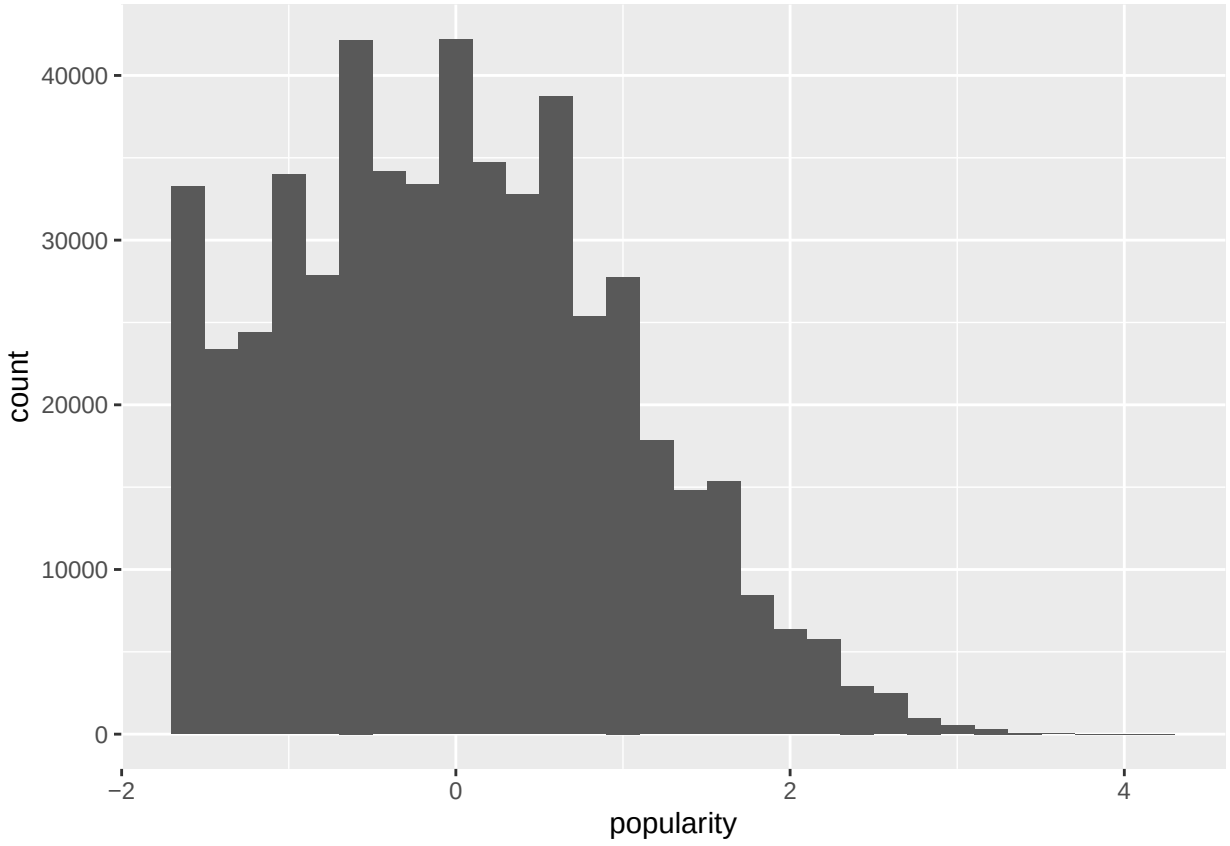
```
# Visualize filtered tracks popularity histogram
tracks_pop_filtered <- ggplot(data=popular_df, aes(x=popularity), title="Full Tracks Dataset Popularity
#artists_pop_filtered <- ggplot(data=artists_df, aes(x=popularity), title="Full Artists Dataset Popular
tracks_pop_scaled <- ggplot(data=popular_df2, aes(x=popularity), title="Full Tracks Dataset Popularity

tracks_pop_filtered
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
tracks_pop_scaled
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
#artists_pop_filtered
```

Correlation Matrix

Plots by Year and by Decade

To explore how the some of the variables in the dataset vary between different musical eras, we generated bar charts of the mean popularity, acoustic-ness, energy level, and song duration for each decade in the dataset by simplifying the 'date_published' value to a decade label. These plots show that newer music is generally more popular than older music, particularly music from decades before 1950. It also shows that the length of songs has stayed fairly steady over time, while the musical energy level has increased somewhat and the degree of acoustic instrumentation has decreased.

```
dim(tracks_df)
```

```
## [1] 586672     20
```

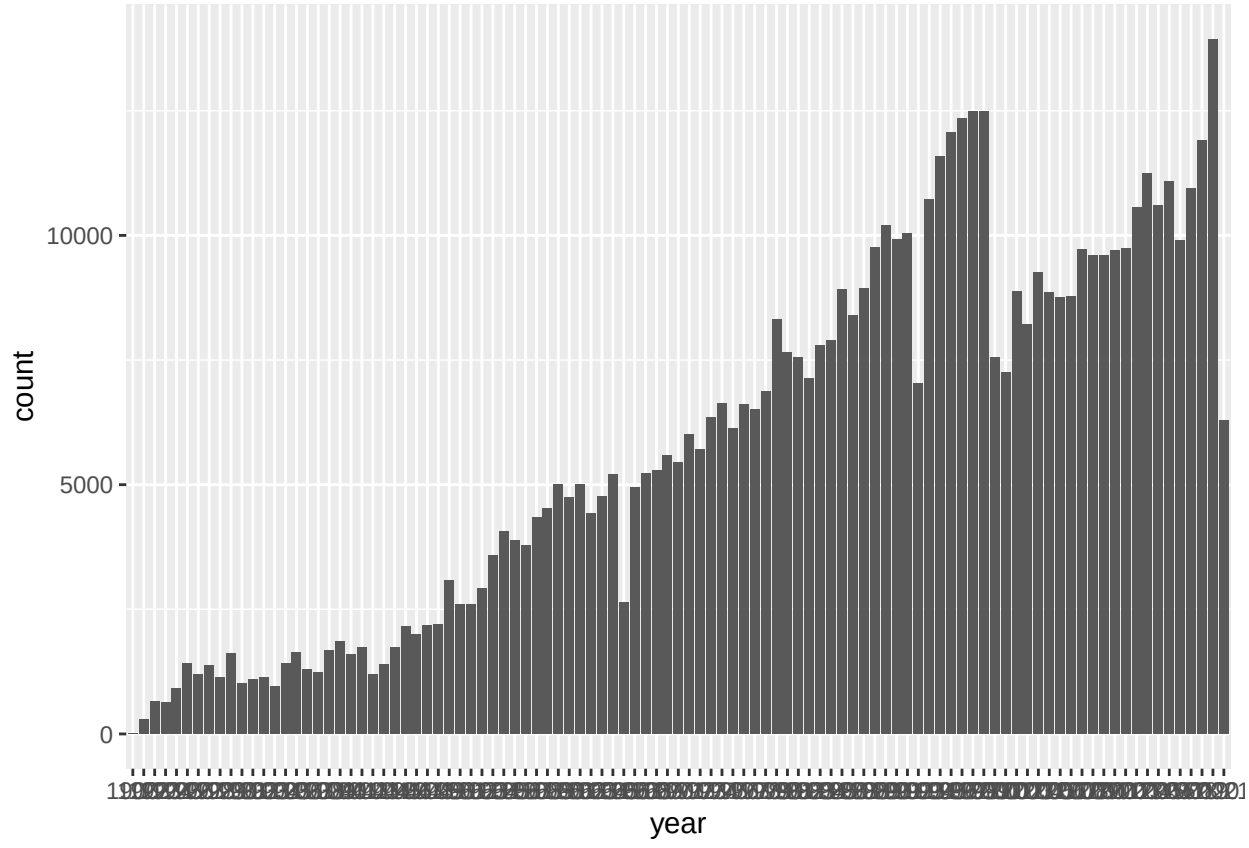
```
tracks_df$year <- eval(substr(tracks_df$release_date, 1,4))
```

```
test_str <- "1922-02-22"
test_substr <- substr(test_str, 1,4)
test_substr
```

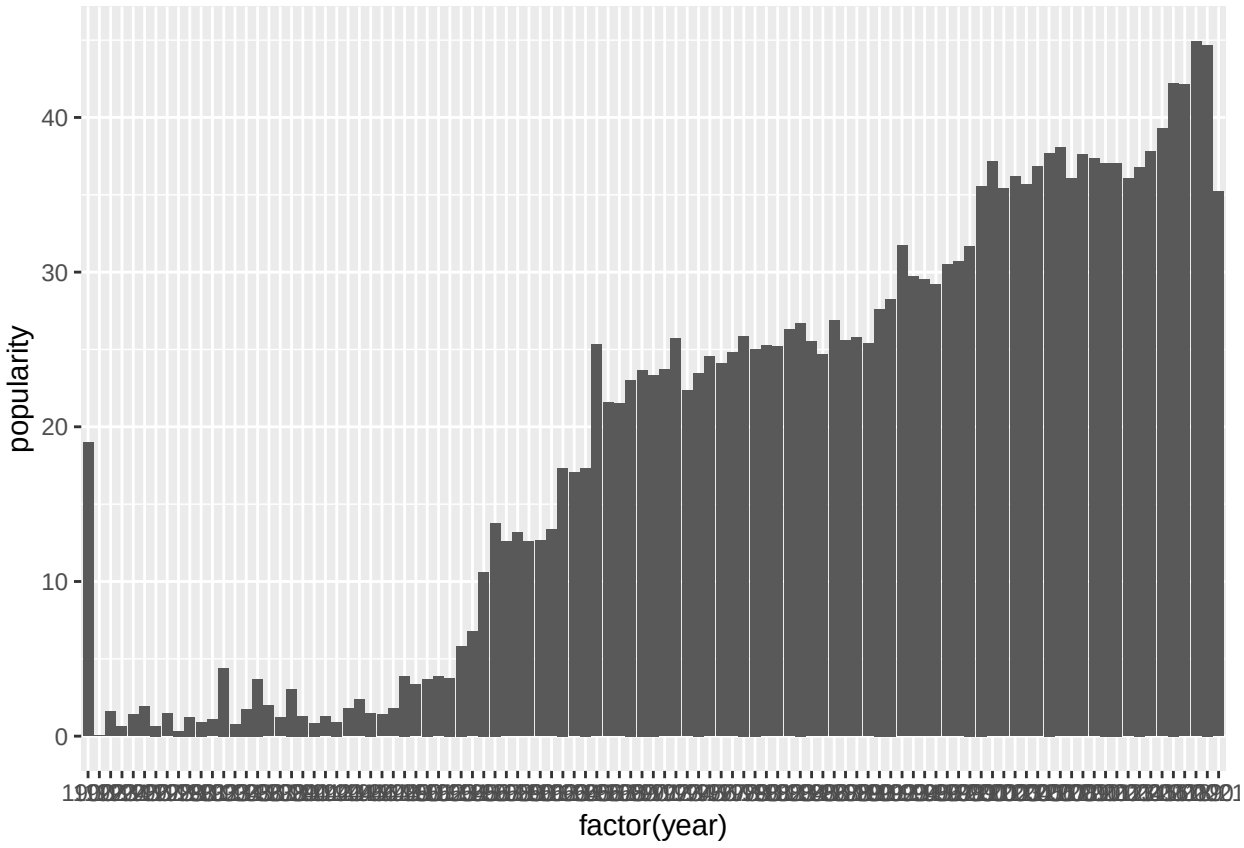
```
## [1] "1922"
```

```
tracks_df$decade <- eval(substr(tracks_df$release_date, 1,3))
```

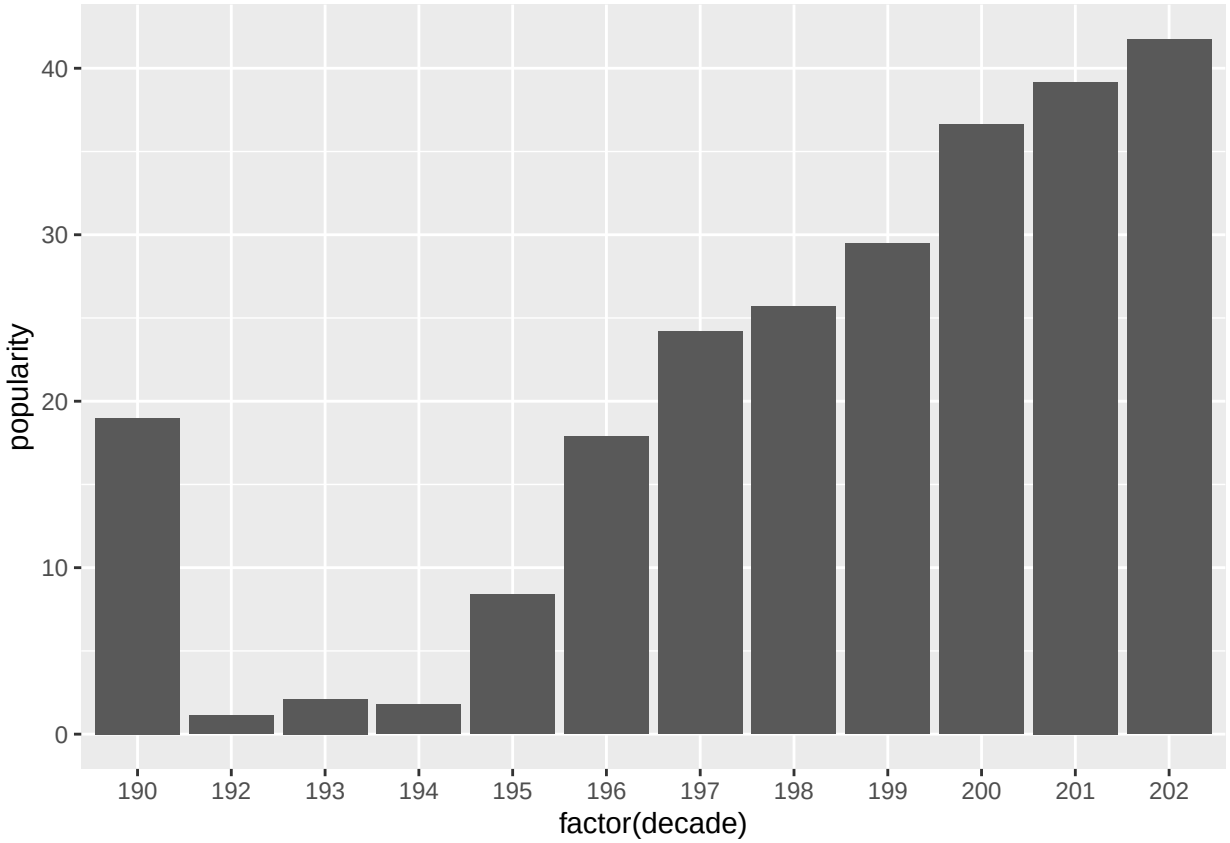
```
ggplot(data = tracks_df, mapping = aes(x = year)) + geom_bar()
```




```
ggplot(tracks_df, aes(x = factor(year), y = popularity)) +
  geom_bar(stat = "summary", fun = "mean")
```

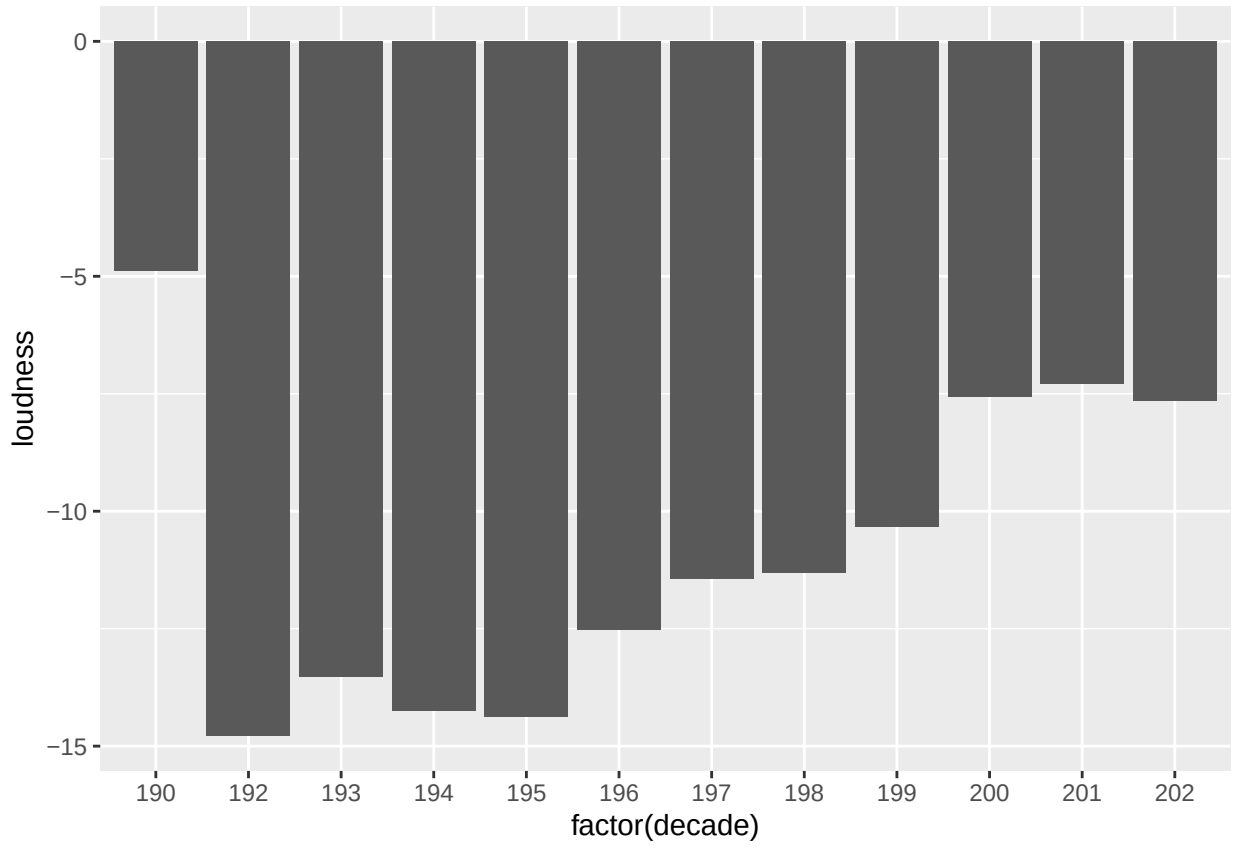



```
ggplot(tracks_df, aes(x = factor(decade), y = popularity)) +
  geom_bar(stat = "summary", fun = "mean")
```

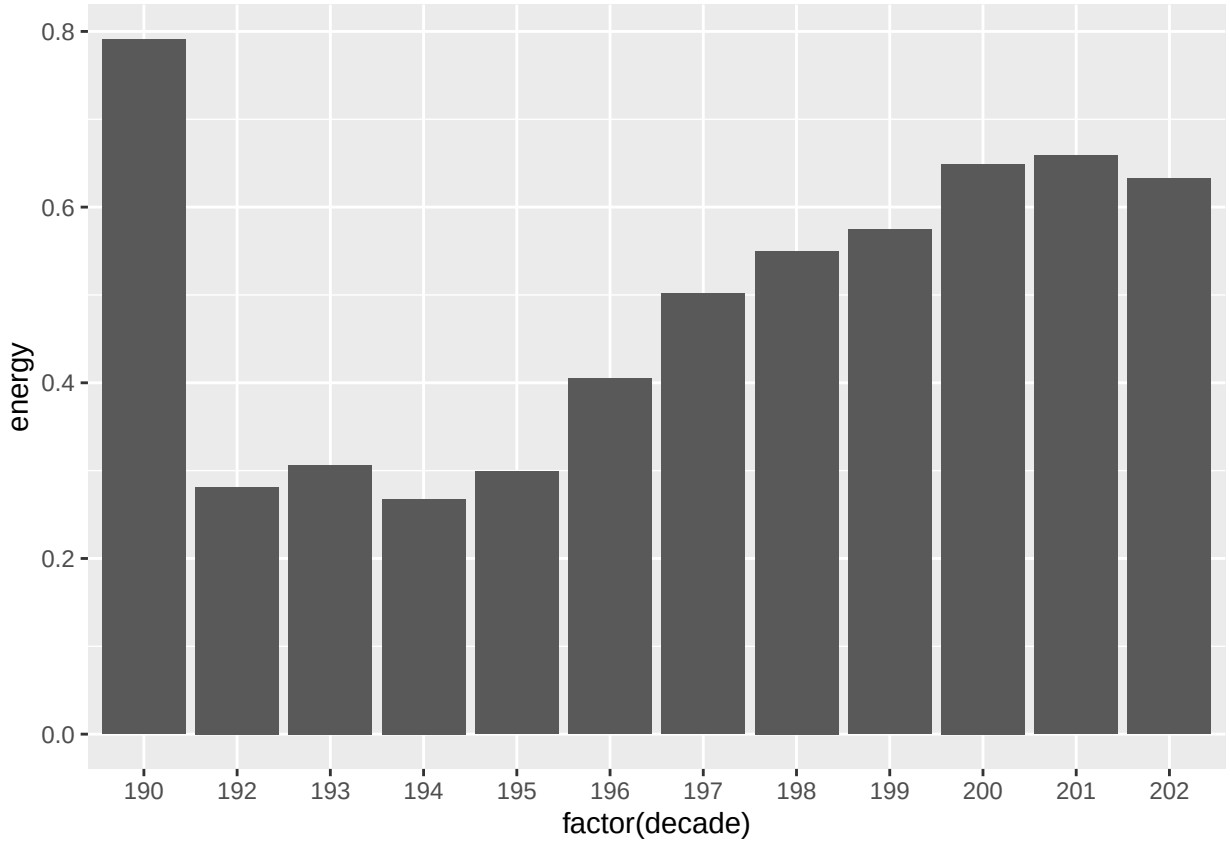



```
#+ guide_axis()
```

```
ggplot(tracks_df, aes(x = factor(decade), y = loudness)) +
  geom_bar(stat = "summary", fun = "mean")
```




```
ggplot(tracks_df, aes(x = factor(decade), y = energy)) +
  geom_bar(stat = "summary", fun = "mean")
```




```
# tracks_190 <- filter(tracks_df, filter = tracks_df$decade == "190") this line did not knit
```

```
# dim(tracks_190)
```

```
tracks_df[478628,]
```

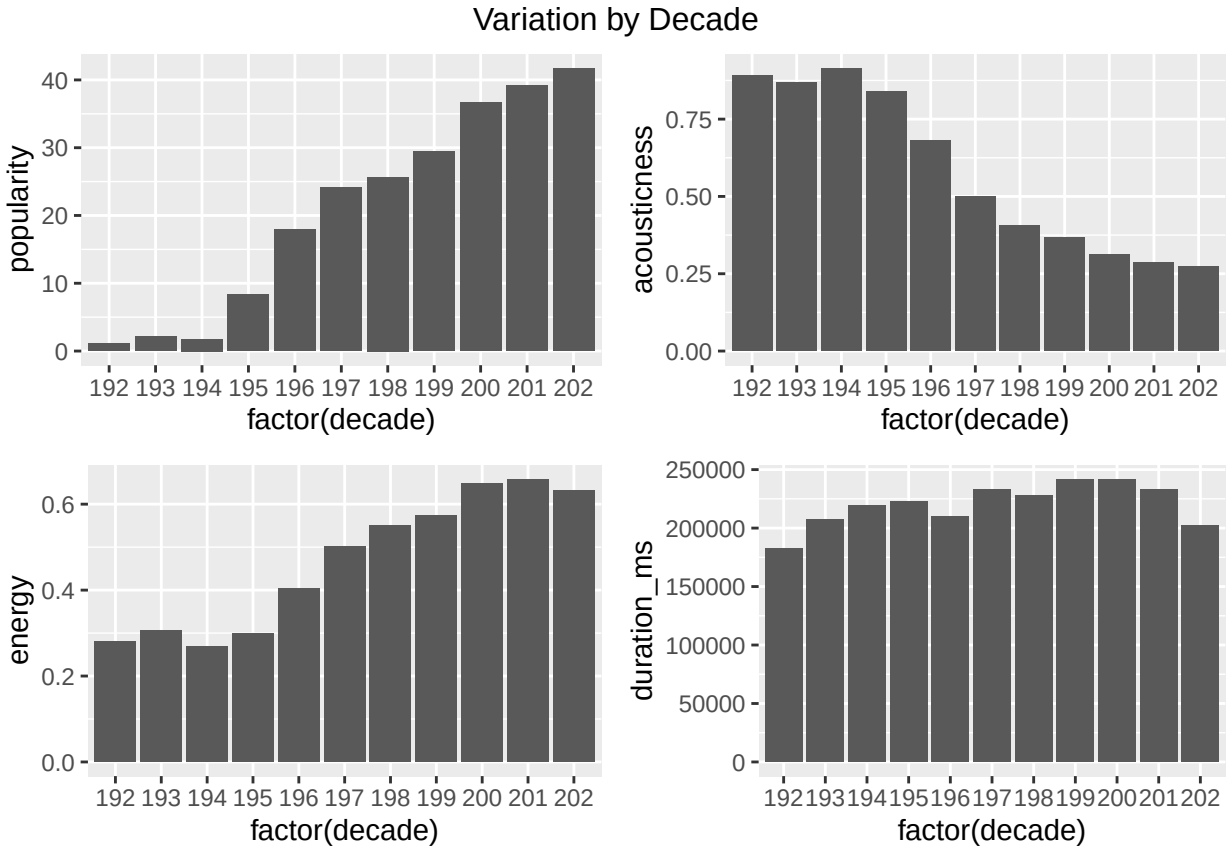
```
##                            id                        name popularity duration_ms
## 478628 74CSJTE5QQp1e4bHzm3wti Maldita sea la primera vez         19      233920
##        explicit                   artists                id_artists
## 478628        0 ['Los Pincheira del Sur'] ['1BnQrx8p0bHBpidjIGq26z']
##        release_date danceability energy key loudness mode speechiness
## 478628   1900-01-01        0.659  0.791   2   -4.895    1      0.0295
##        acousticness instrumentalness liveness valence tempo time_signature year
## 478628        0.139       0.00000163    0.161   0.956   142              4 1900
##        decade
## 478628    190
```

```
tracks_clean <- tracks_df[-478628,]
```

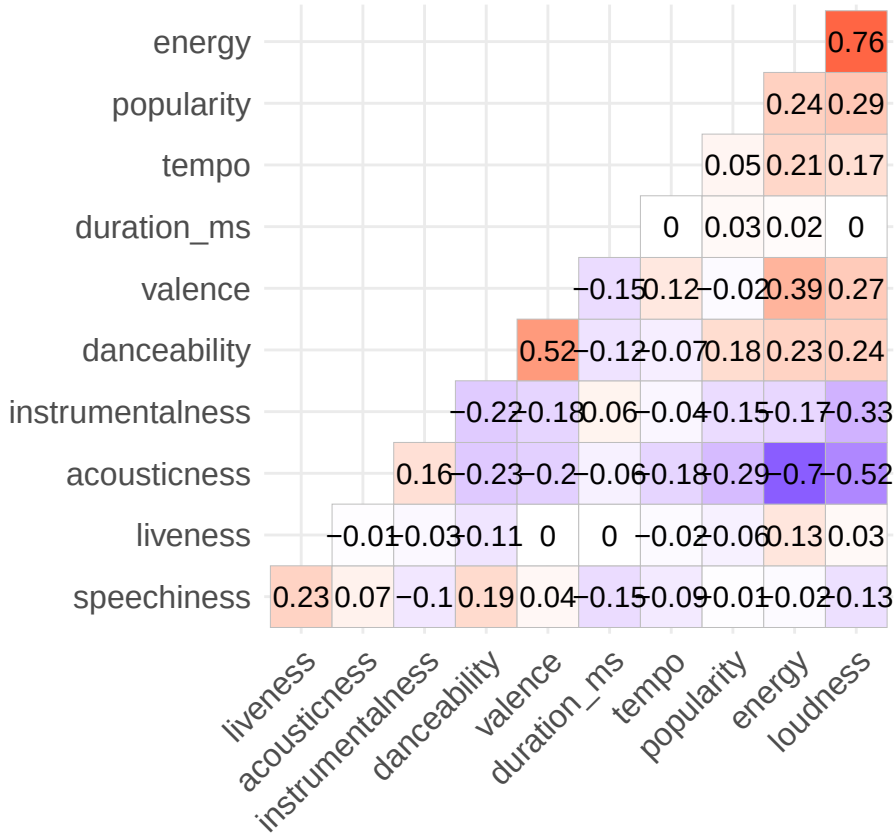
```
plt2 <- ggplot(tracks_clean, aes(x = factor(decade), y = acousticness)) +
  geom_bar(stat = "summary", fun = "mean")

plt3 <- ggplot(tracks_clean, aes(x = factor(decade), y = energy)) +
  geom_bar(stat = "summary", fun = "mean")
```

```
plt1 <- ggplot(tracks_clean, aes(x = factor(decade), y = popularity)) +
  geom_bar(stat = "summary", fun = "mean")

plt4 <- ggplot(tracks_clean, aes(x = factor(decade), y = duration_ms)) +
  geom_bar(stat = "summary", fun = "mean")
```

```
grid.arrange(plt1, plt2, plt3, plt4, top = "Variation by Decade", ncol = 2)
```



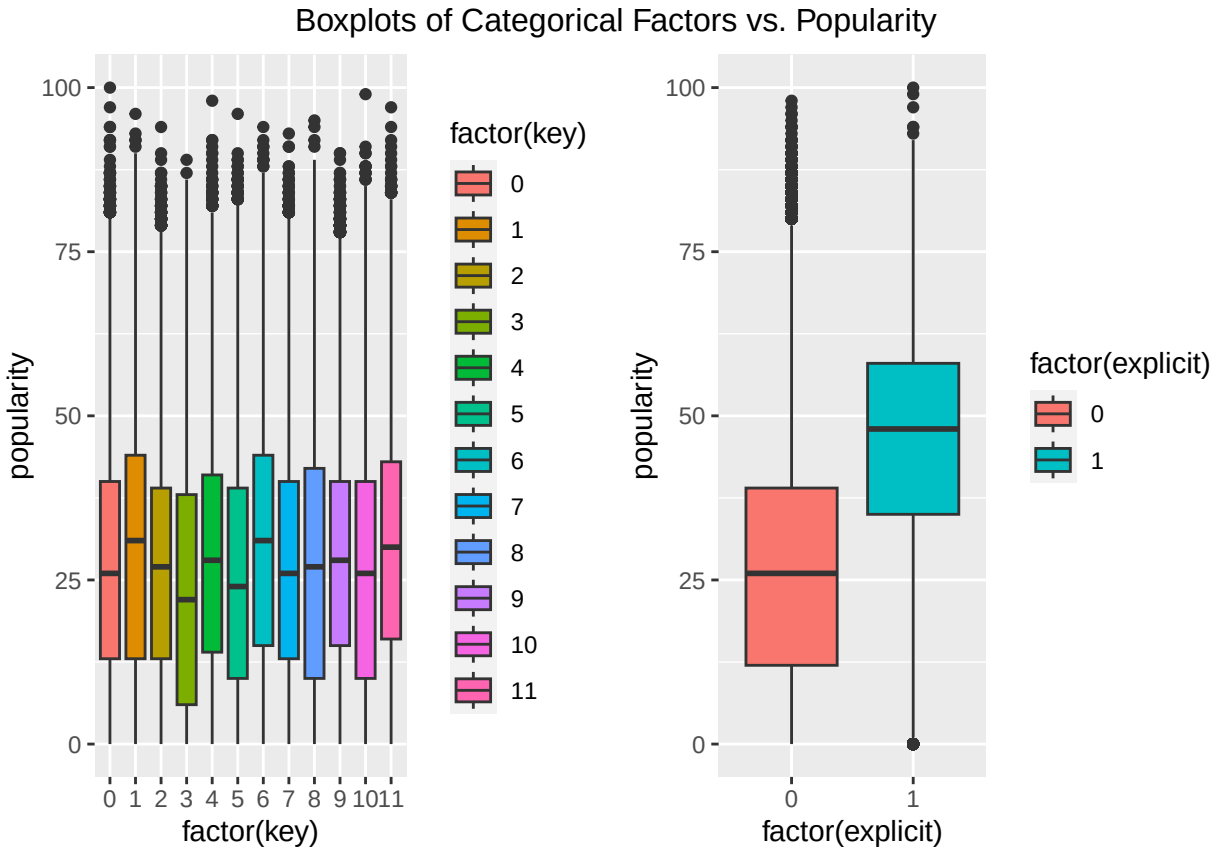

```
# Calculate the correlation matrix
cor_matrix <- cor(popular_numeric)

# plot the correlation
ggcorrplot(cor_matrix,
           hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           show.legend = FALSE
           )
```



Popularity and Hit Boxplots

To begin investigating the idea that *popularity* could be related to features which describe the character of the music, it would be useful to do some preliminary visual exploration of those features. Two of the available "factor" variables (*key* and *explicit*) can be analyzed using boxplots:

```
box1 <- ggplot(data = tracks_df, aes(x=factor(key), y=popularity, fill=factor(key))) + geom_boxplot()
box2 <- ggplot(data = tracks_df, aes(x=factor(explicit), y=popularity, fill=factor(explicit))) + geom_bo

grid.arrange(box1, box2, ncol = 2, top="Boxplots of Categorical Factors vs. Popularity")
```

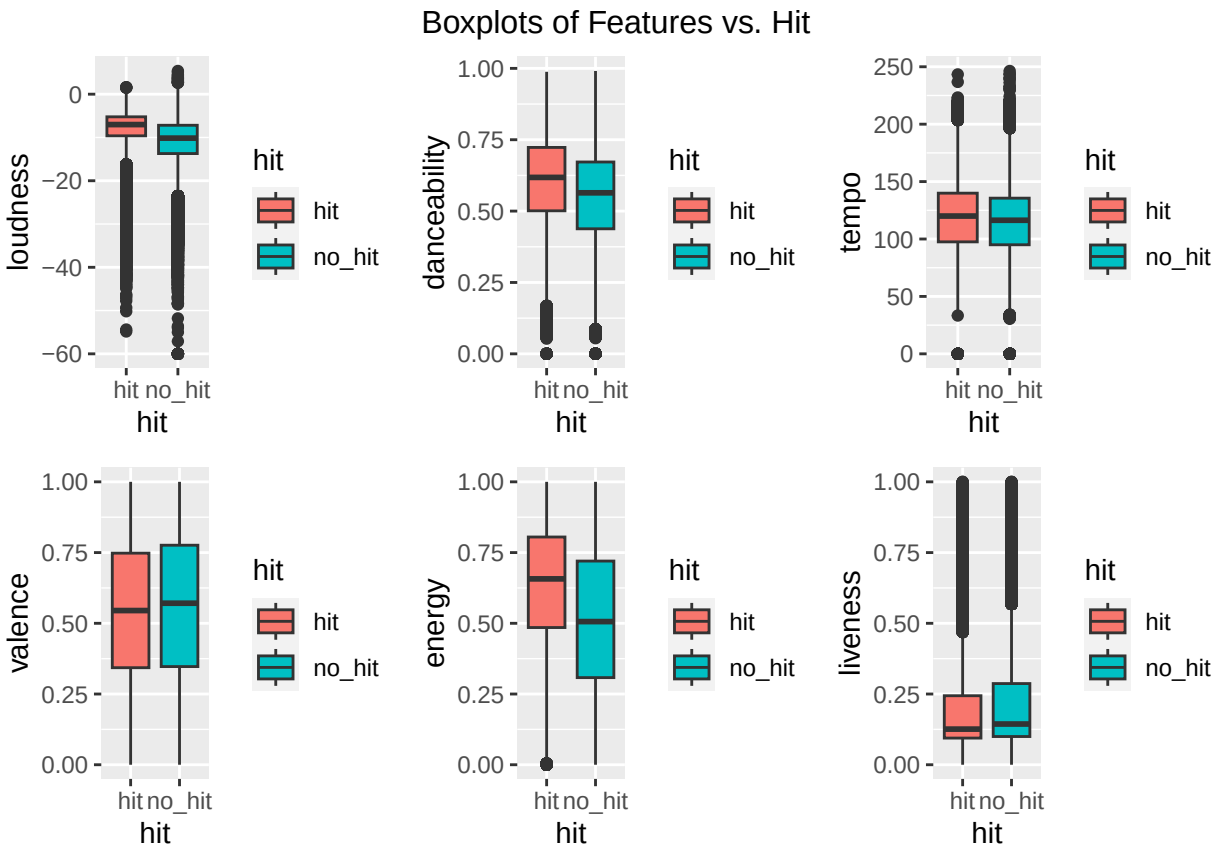


It can be seen that the interquartile range of "popularity" appears to be different for songs flagged as explicit. For the song "key", there may be differences but they are not as obvious in this visual.

In addition to predicting the numerical value of popularity, another opportunity is to predict if a song is a "hit" or not (i.e., predict if the song's popularity will be in the top quartile). By framing in the problem this binary way, we will have an opportunity to investigate more potential models (such as logistiic regression). Additionally, some people might find more meaning in a clear simple, classification such as "hit" and "no hit".

To begin investigating "hit" and "no hit" visually, we first classified a tracks in the top quartile of popularity as "hit", and other songs and "no hit". From here, we produced boxplots of "hit" or "no_hit" for different features such as *loudness* and *danceability*:

```
quant_pop <- tracks_df %>%
              mutate(quartile = factor(ntile(popularity, 4)),
                     hit = case_when(quartile == 4 ~"hit", TRUE~"no_hit"))

box1 <- ggplot(data = quant_pop, aes(x=hit, y=loudness, fill=hit)) + geom_boxplot()
box2 <- ggplot(data = quant_pop, aes(x=hit, y=danceability, fill=hit)) + geom_boxplot()
box3 <- ggplot(data = quant_pop, aes(x=hit, y=tempo, fill=hit)) + geom_boxplot()
box4 <- ggplot(data = quant_pop, aes(x=hit, y=valence, fill=hit)) + geom_boxplot()
box5 <- ggplot(data = quant_pop, aes(x=hit, y=energy, fill=hit)) + geom_boxplot()
box6 <- ggplot(data = quant_pop, aes(x=hit, y=liveness, fill=hit)) + geom_boxplot()


grid.arrange(box1, box2, box3, box4, box5, box6, ncol = 3, top="Boxplots of Features vs. Hit")
```



At least visually it appears that some of these features could be useful in predicting which tracks are a hit. For example, the mean of "loudness" and "energy" appear to be higher for "hit" than "no_hit". On the other hand, some of these features don't seem to have such an obvious relationship in this visual; the means and interquartile ranges for the tempo plot do not appear to be so different between "hit" and "no hit".

Sampling

To build some of our models, we decided to use random sampling to devide the data into training and testing sets.

```
set.seed(1)

N <- nrow(popular_df2)
n <- N * 0.8

idx = sample(1:N, size=n, replace = FALSE)
train = popular_df2[idx,]
test = popular_df2[-idx,]
```

3 Classification of Target Variable - Hit or Not?

As we have stated, the dependent variable that we are using to assess if a song will be classified as a hit is "popularity," a score from 0 to 100 that ranks how popular an artist is relative to other artists on the

platform. To use this variable in classification problems, we decided to code it, "1" means that the song is a hit, "0" indicates that the song is not a hit; the threshold used to make the division was the 75th percentile in the popularity column, everything below the 75th percentile is considered "not hit", the rest is considered a "hit".

```
popular_df2 <- popular_df2 %>% mutate(popularity_coded = ifelse(popularity >= quantile(popular_df2$popul

# Convert the outcome variable to a factor
popular_df2$popularity_coded <- as.factor(popular_df2$popularity_coded)

# control object for k-fold cross validation
ctrl <- trainControl(method = "cv", number = 10)
```

3.1 LDA

```
#fit a regression model and use k-fold CV to evaluate performance
lda_model <- train(popularity_coded~duration_ms+explicit+danceability+energy+key+loudness+mode+speechin

print(lda_model)
```

```
## Linear Discriminant Analysis
##
## 529958 samples
##     14 predictor
##      2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 476962, 476962, 476963, 476963, 476963, 476962, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.7572    0.1721
```

3.1.1 LDA Assumptions

Multivariate Normality - mulri.norm Test

H_0 (Null hypothesis): The variables follow a multivariate normal distribution.

H_A (Alternative hypothesis): The variables do not follow a multivariate normal distribution.

We will use an alpha value of 0.05.

```
# Multivariate Normality
N <- nrow(popular_scaled)
idx = sample(1:N, size=1000, replace = FALSE)
popular_sample = popular_scaled[idx,]
mult.norm(popular_sample)$mult.test
```

```
##          Beta-hat   kappa p-val
## Skewness    225.4 37560.7     0
## Kurtosis    472.0   307.6     0
```

Equality of Variance - Levene Test

H_0 (Null hypothesis): Sample variances are equal.

H_A (Alternative hypothesis): Samples variances are not equal.

We will use an alpha value of 0.05.

```
# Equality of Variance
popularity_var_test <- data.frame(popular_factors, popular_df2["popularity"])
N <- nrow(popularity_var_test)
idx = sample(1:N, size=1000, replace = FALSE)
popular_sample2 = popularity_var_test[idx,]
leveneTest(popularity~., data = popular_sample2)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group  86    1.02   0.44
##       913
```

3.2 Logistic Regression

```
#fit a regression model and use k-fold CV to evaluate performance
lr_model <- train(popularity_coded ~ duration_ms+explicit+danceability+energy+key+loudness+mode+speechi

summary(lr_model)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.378  -0.782  -0.560   0.764   3.973
##
## Coefficients:
##                  Estimate Std. Error z value             Pr(>|z|)
## (Intercept)       1.55382    0.14363   10.82 < 0.0000000000000002 ***
## duration_ms       0.01789    0.00387    4.63     0.00000368242839 ***
## explicit1         1.26024    0.01499   84.07 < 0.0000000000000002 ***
## danceability      0.37619    0.00446   84.33 < 0.0000000000000002 ***
## energy           -0.18598    0.00720  -25.83 < 0.0000000000000002 ***
## key1              0.25819    0.01563   16.52 < 0.0000000000000002 ***
## key2             -0.04236    0.01395   -3.04              0.00240 **
## key3              0.12211    0.02106    5.80     0.00000000668318 ***
## key4              0.04076    0.01515    2.69              0.00712 **
## key5              0.05113    0.01502    3.40              0.00066 ***
## key6              0.26415    0.01700   15.54 < 0.0000000000000002 ***
## key7             -0.00958    0.01353   -0.71              0.47901
## key8              0.23179    0.01692   13.70 < 0.0000000000000002 ***
## key9              0.00189    0.01388    0.14              0.89159
## key10             0.12010    0.01658    7.24     0.00000000000044 ***
## key11             0.17803    0.01585   11.23 < 0.0000000000000002 ***
```

```
## loudness          0.68378    0.00680  100.62 < 0.0000000000000002 ***
## mode1             0.02835    0.00733    3.87              0.00011 ***
## speechiness      -0.21075    0.00544  -38.72 < 0.0000000000000002 ***
## acousticness     -0.26449    0.00503  -52.63 < 0.0000000000000002 ***
## instrumentalness -0.05438    0.00427  -12.73 < 0.0000000000000002 ***
## liveness         -0.07829    0.00380  -20.61 < 0.0000000000000002 ***
## valence          -0.39463    0.00443  -89.09 < 0.0000000000000002 ***
## tempo             0.07500    0.00366   20.51 < 0.0000000000000002 ***
## time_signature1  -2.99683    0.14944  -20.05 < 0.0000000000000002 ***
## time_signature3  -3.04856    0.14376  -21.21 < 0.0000000000000002 ***
## time_signature4  -2.93093    0.14340  -20.44 < 0.0000000000000002 ***
## time_signature5  -2.80910    0.14647  -19.18 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 608015  on 529957  degrees of freedom
## Residual deviance: 536462  on 529930  degrees of freedom
## AIC: 536518
##
## Number of Fisher Scoring iterations: 5
```

3.2.1 Logistic Regression Assumptions

Absence of Multicolinearity - Variance Inflation Factor

```
## VIF
model_fit<-lm(popularity~duration_ms+factor(explicit)+ danceability+energy+factor(key)+loudness+factor(

vif(model_fit)
```

```
##                          GVIF Df GVIF^(1/(2*Df))
## duration_ms             1.063  1           1.031
## factor(explicit)        1.080  1           1.039
## danceability            1.727  1           1.314
## energy                  4.322  1           2.079
## factor(key)             1.121 11           1.005
## loudness                2.916  1           1.708
## factor(mode)            1.080  1           1.039
## speechiness             1.342  1           1.159
## acousticness            2.142  1           1.464
## instrumentalness        1.218  1           1.104
## liveness                1.138  1           1.067
## valence                 1.724  1           1.313
## tempo                   1.110  1           1.054
## factor(time_signature)  1.241  4           1.027
```

Lack of Influential Outliers - Cook's Distance

```
# Influential Outliers
popular_df[cooks.distance(model_fit)>1,]
```

```
##  [1] id               name             popularity       duration_ms
##  [5] explicit         artists          id_artists       release_date
##  [9] danceability     energy           key              loudness
## [13] mode             speechiness      acousticness     instrumentalness
## [17] liveness         valence          tempo            time_signature
## <0 rows> (or 0-length row.names)
```

3.3 Classification Tree

To compare to the previous classification models, a classification tree was also built with the same predictor varaibles. As with the other models, training was done with 80% of data, and testing with the remaining 20%.

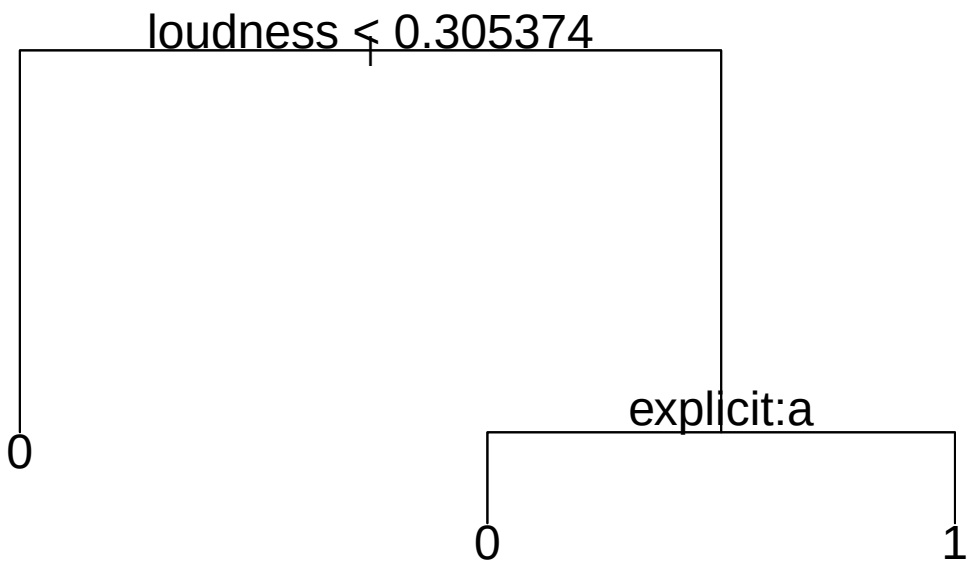
```
set.seed(1)

N <- nrow(popular_df2)
n <- N * 0.8

idx = sample(1:N, size=n, replace = FALSE)
train = popular_df2[idx,]
test = popular_df2[-idx,]
```

The classification tree was build as shown in the code below:

```
tree.class <- tree(popularity_coded ~ duration_ms+explicit+danceability+energy+key+loudness+mode+speechi
summary(tree.class)
```

```
##
## Classification tree:
## tree(formula = popularity_coded ~ duration_ms + explicit + danceability +
##     energy + key + loudness + mode + speechiness + acousticness +
##     instrumentalness + liveness + valence + tempo + time_signature,
##     data = train)
## Variables actually used in tree construction:
## [1] "loudness" "explicit"
## Number of terminal nodes:  3
## Residual mean deviance:  1.07 = 454000 / 424000
## Misclassification error rate: 0.247 = 104805 / 423966
```

```
plot(tree.class)
text(tree.class, cex=1.5, col="black")
```



As shown in the plot, the tree produced with default setting was very simple. It contained only 3 terminal nodes and only used "loudness" and "explicit" to perform the classification. The misclassification rate was determined with the 20% test data as shown below:

```
tree.pred <- predict(tree.class, test, type = "class")

tab <- table(tree.pred, test$popularity_coded)
tab
```

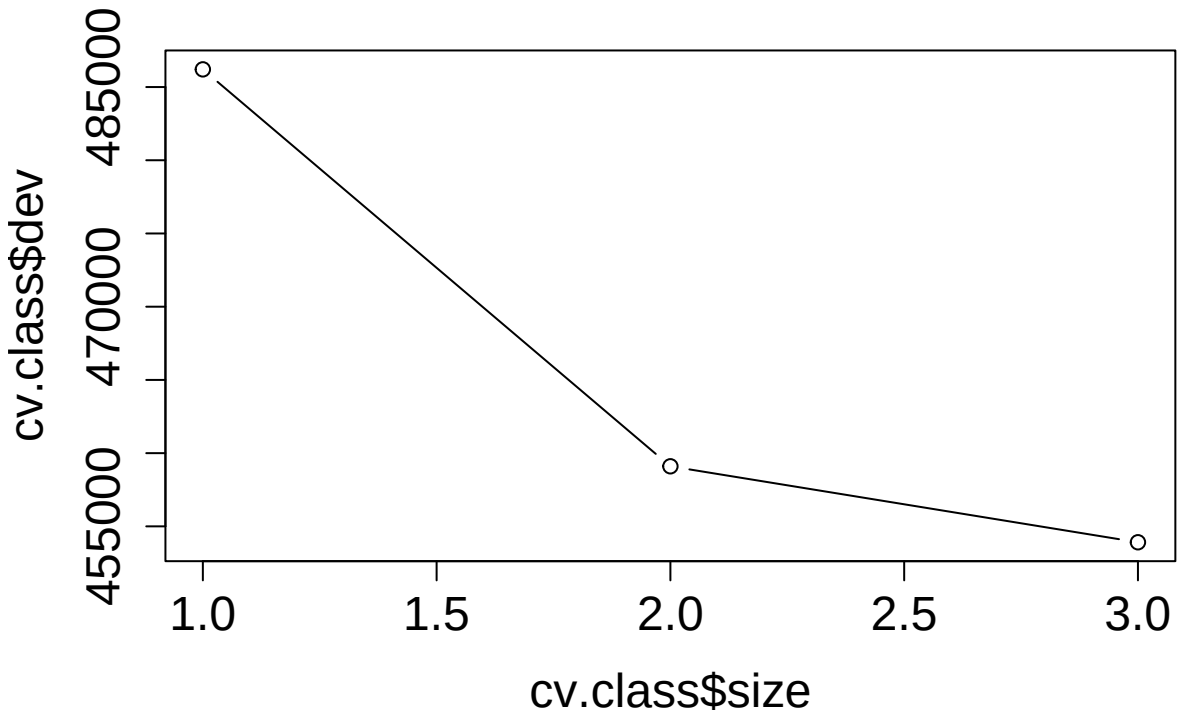
```
##
## tree.pred     0     1
##         0 77109 25174
##         1  1165  2544
```

```
mis = 1 - sum(diag(tab)) / sum(tab)
mis
```

```
## [1] 0.2485
```

The misclassification rate was found to be 0.285. Since the tree was already very small, it was not expected that pruning would be necessary. Still, we confirmed this using cross-validation to choose tree complexity. The chart shown below confirms the suspicion that 3 terminal nodes is optimal for this case; reducing terminal nodes would reduce model performance.

```
cv.class = cv.tree(tree.class)
plot(cv.class$size, cv.class$dev, type='b', cex.main=1.5, cex.lab=1.5, cex.axis=1.5)
```



Now that the tree complexity has been determined, we used 10 k fold cross-validation to determine the mean misclassification. To perform this cross-validation, 10 folds were produced which were stratified on "hit" or "no hit" i.e, "popularity_coded".

```
set.seed(10)
strat_folds <- createFolds(factor(popular_df2$popularity_coded), k=10)

for (i in 1:10){

idx <- strat_folds[[i]]
fold <- popular_df2[idx,]
# print(table(fold$popularity_coded))

}


mis_tree <- function(idx){
  Train <- popular_df2[-idx,]
  Test <- popular_df2[idx,]
  tree.class <- tree(popularity_coded ~ duration_ms+explicit+danceability+energy+key+loudness+mode+spee
  tree_hat <- predict(tree.class, Test, type = "class")

  tab <- table(tree_hat, Test$popularity_coded)
```

```
  print(tab)

  misclass = 1 - sum(diag(tab)) / sum(tab)
  print(misclass)
  return(misclass)

}

misclass_tree = lapply(strat_folds, mis_tree)
```

```
##
## tree_hat     0     1
##        0 38532 12501
##        1   655  1308
## [1] 0.2482
##
## tree_hat     0     1
##        0 38574 12557
##        1   613  1251
## [1] 0.2485
##
## tree_hat     0     1
##        0 38573 12478
##        1   614  1331
## [1] 0.247
##
## tree_hat     0     1
##        0 38563 12486
##        1   624  1323
## [1] 0.2474
##
## tree_hat     0     1
##        0 38599 12458
##        1   588  1350
## [1] 0.2462
##
## tree_hat     0     1
##        0 38597 12475
##        1   590  1334
## [1] 0.2465
##
## tree_hat     0     1
##        0 38602 12459
##        1   585  1350
## [1] 0.2461
##
## tree_hat     0     1
##        0 38551 12540
##        1   636  1268
## [1] 0.2486
##
## tree_hat     0     1
##        0 38599 12528
```

```
##        1   589  1281
## [1] 0.2475
##
## tree_hat     0     1
##        0 38583 12518
##        1   604  1291
## [1] 0.2476
```

```
"The mean misclassification:"
```

```
## [1] "The mean misclassification:"
```

```
mean(as.numeric(misclass_tree))
```

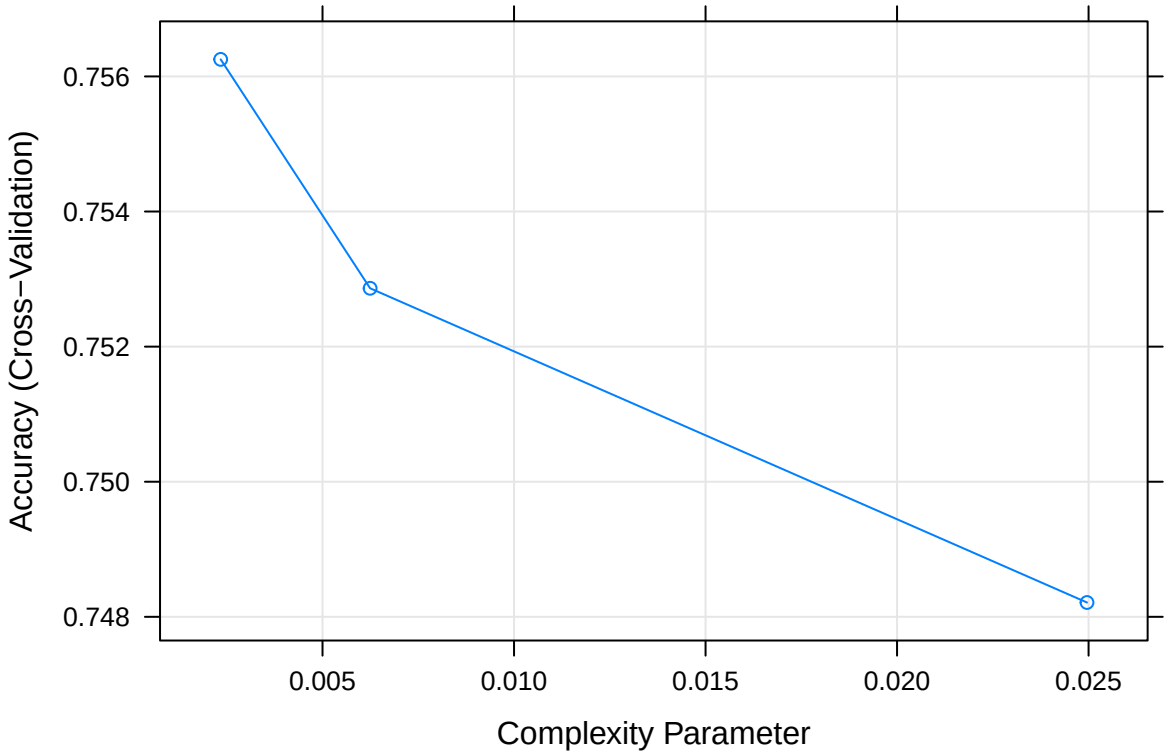
```
## [1] 0.2474
```

The results of the 10 k-fold cross validation echoed the results of the training set. The mean misclassification was 0.2474. This is a higher misclassification than the the logisitic regression model, and therefore the classification tree is not recommended.

```
tree_model_class <- train(popularity_coded ~ duration_ms+explicit+danceability+energy+key+loudness+mode+

print(tree_model_class)
```

```
## CART
##
## 529958 samples
##     14 predictor
##      2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 476962, 476962, 476962, 476961, 476963, 476963, ...
## Resampling results across tuning parameters:
##
##   cp        Accuracy  Kappa
##   0.002342  0.7563    0.16421
##   0.006242  0.7529    0.11532
##   0.024959  0.7482    0.07455
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.002342.
```

```
plot(tree_model_class)
```



3.3.1 Classification Tree Assumptions

For tree methods, we do not have any assumptions to test besides our observations being independent of each other, which they are.

4.4 Regression Tree

Is a supervised learning statistical model where the target variable can take continuous values. We use the model to predict the popularity score for each of the songs, using all other variables as predictors, first we divide the data into training and testing using simple random sampling, we used 80% of the data as training and 20% as testing.

A regression Tree of 6 terminal nodes was created and using the variables loudness, acousticness, explicit and duration_ms as predictors.

```
popularity_tree_fit <- tree(popularity ~ duration_ms+explicit+danceability+energy+key+loudness+mode+spe

summary(popularity_tree_fit)
```

```
##
## Regression tree:
## tree(formula = popularity ~ duration_ms + explicit + danceability +
##     energy + key + loudness + mode + speechiness + acousticness +
##     instrumentalness + liveness + valence + tempo + time_signature,
```

```
##     data = train)
## Variables actually used in tree construction:
## [1] "loudness"     "acousticness" "explicit"     "duration_ms"
## Number of terminal nodes:  6
## Residual mean deviance:  0.852 = 361000 / 424000
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -2.730  -0.694  -0.043   0.000   0.609   4.180
```

The RMSE obtained for our Regression Tree is 0.9239. RMSE is in the same units as our target value, in this case we have a very big RMSE if we compared to 0.00000000000000005539.

```
popularity_tree_predict <- predict(popularity_tree_fit, test)
print('RMSE:')
```

```
## [1] "RMSE:"
```

```
print(sqrt(mean((popularity_tree_predict - test$popularity)^2)))
```

```
## [1] 0.9239
```

```
print('Mean')
```

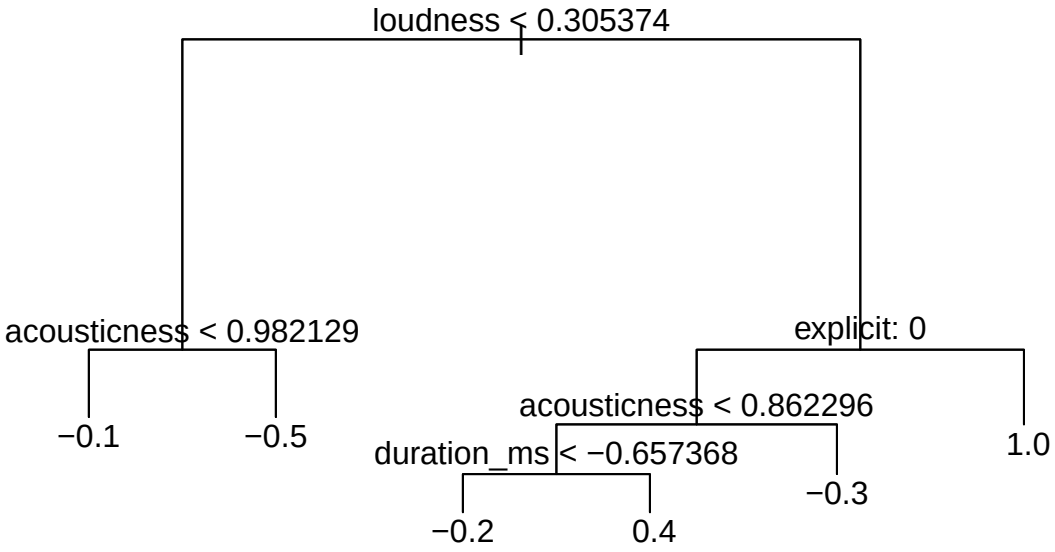
```
## [1] "Mean"
```

```
print(mean(popular_df2$popularity))
```

```
## [1] 0.00000000000000005539
```

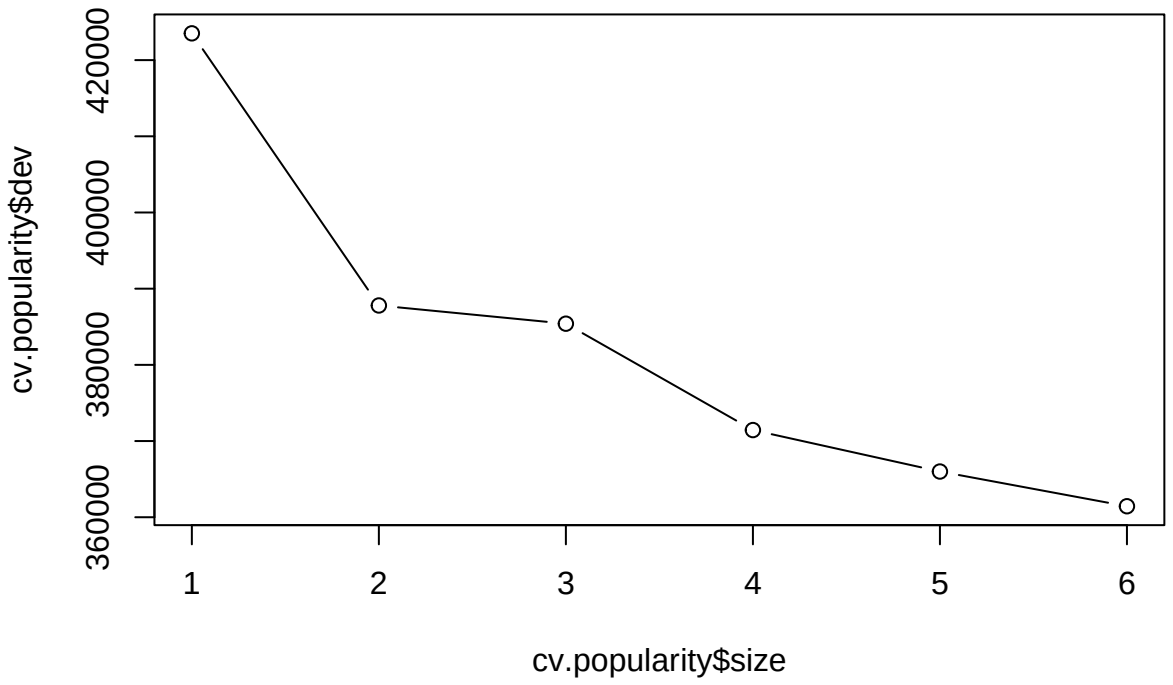
Here below we can observe the regression tree plotted.

```
plot(popularity_tree_fit)
text(popularity_tree_fit ,pretty =0)
```



To see if tree pruning is needed, we plot the the cross-validation error and the size of the tree, we concluded that 6 terminal nodes is the best performing tree so no pruning is needed.

```
cv.popularity=cv.tree(popularity_tree_fit, K=10)
plot(cv.popularity$size, cv.popularity$dev,type='b')
```



Lastly, we did k-fold cross validation to see if we see an improvement. The RMSE obtained is very similar to the previous value, meaning that this model is not very useful to obtain the real value of popularity.

```
tree_model_reg <- train(popularity ~ duration_ms+explicit+danceability+energy+key+loudness+mode+speechi
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
print(tree_model_reg)
```

```
## CART
##
## 529958 samples
##     14 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 476962, 476962, 476963, 476962, 476962, 476962, ...
## Resampling results across tuning parameters:
##
##   cp       RMSE    Rsquared  MAE
##   0.01833  0.9424  0.11196   0.7670
##   0.02033  0.9516  0.09436   0.7758
##   0.08470  0.9788  0.08303   0.8021
##
```

```
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.01833.
```

3.4.1 Assumptions

For tree methods, we do not have any assumptions to test besides our observations being independant of each other, which they are.

4 Summary of Findings

5 Conclusion

6 References

1. https://investors.spotify.com/about/default.aspx
2. https://developer.spotify.com/documentation/web-api/reference/#/operations/get-audio-features
3. https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks
4. https://cdla.dev/sharing-1-0/


```
#clean the artists to those with only one genre and of the primary genre types

artist_genres <- artists_df %>%
                  filter(genres != "[]") %>%
                  mutate(genres = toupper(genres),
                         rock = grepl("ROCK", genres, fixed=TRUE),
                         rnb = grepl("R&B", genres, fixed=TRUE),
                         pop = grepl("POP", genres, fixed=TRUE),
                         country = grepl("COUNTRY", genres, fixed=TRUE),
                         rap = grepl("RAP", genres, fixed=TRUE),
                         jazz = grepl("JAZZ", genres, fixed=TRUE),
                         classical = grepl("CLASSICAL", genres, fixed=TRUE),
                         soul = grepl("SOUL", genres, fixed=TRUE),
                         funk = grepl("FUNK", genres, fixed=TRUE),
                         electronic = grepl("ELECTRONIC", genres, fixed=TRUE),
                         disco = grepl("DISCO", genres, fixed=TRUE),
                         num_genres = rock + rnb + pop + country + rap + jazz + classical + soul + funk
                  filter(num_genres == 1) %>%
                  mutate(genre = case_when(rock == 1 ~ "rock",
                                           rnb == 1 ~ "rnb",
                                           pop == 1 ~ "pop",
                                           country == 1 ~ "country",
                                           rap == 1 ~ "rap",
                                           jazz == 1 ~ "jazz",
                                           classical == 1 ~ "classical",
                                           soul == 1 ~ "soul",
                                           funk == 1 ~ "funk",
                                           electronic == 1 ~ "electronic",
                                           disco == 1 ~ "disco")) %>%
                dplyr::select(id, genre, name, popularity)
```

```
#clean the track artist lists
rep_str = c("\\[" = "", "\\]"="", "'"="")

track_artists <- tracks_df
track_artists$id_artists <- str_replace_all(track_artists$id_artists, rep_str)
track_artists$id_artists <- as.list(track_artists$id_artists)
```

```
#inner join so only the artists with single genre remain alongsige the track data
genre_tracks <- merge(x=artist_genres, y=track_artists, by.x="id", by.y="id_artists")

colnames(genre_tracks)
```

```
##  [1] "id"               "genre"            "name.x"           "popularity.x"
##  [5] "id.y"             "name.y"           "popularity.y"     "duration_ms"
##  [9] "explicit"         "artists"          "release_date"     "danceability"
## [13] "energy"           "key"              "loudness"         "mode"
## [17] "speechiness"      "acousticness"     "instrumentalness" "liveness"
## [21] "valence"          "tempo"            "time_signature"   "year"
## [25] "decade"
```

```
colnames(genre_tracks) <- c("artist_id", "genre", "artist_name", "artist_popularity", "song_id", "song_
```

```
#count the strata to understand what should be used to test/train
n = 150000
N = dim(genre_tracks)[1]

order <- unique(genre_tracks$genre)
order
```

```
##  [1] "pop"        "rock"       "disco"      "jazz"       "rap"
##  [6] "country"    "electronic" "rnb"        "soul"       "classical"
## [11] "funk"
```

```
strata <- genre_tracks %>%
            count(genre) %>%
            rename(Nh = n) %>%
            mutate(nh = round(Nh*n/N)) %>%
            slice(match(order, genre))
strata
```

```
##         genre     Nh    nh
## 1         pop 112838 81023
## 2        rock  60051 43119
## 3       disco   1026   737
## 4        jazz  16592 11914
## 5         rap   8347  5994
## 6     country   3913  2810
## 7  electronic    985   707
## 8         rnb    336   241
## 9        soul   1502  1079
## 10  classical   2612  1876
## 11       funk    698   501
```

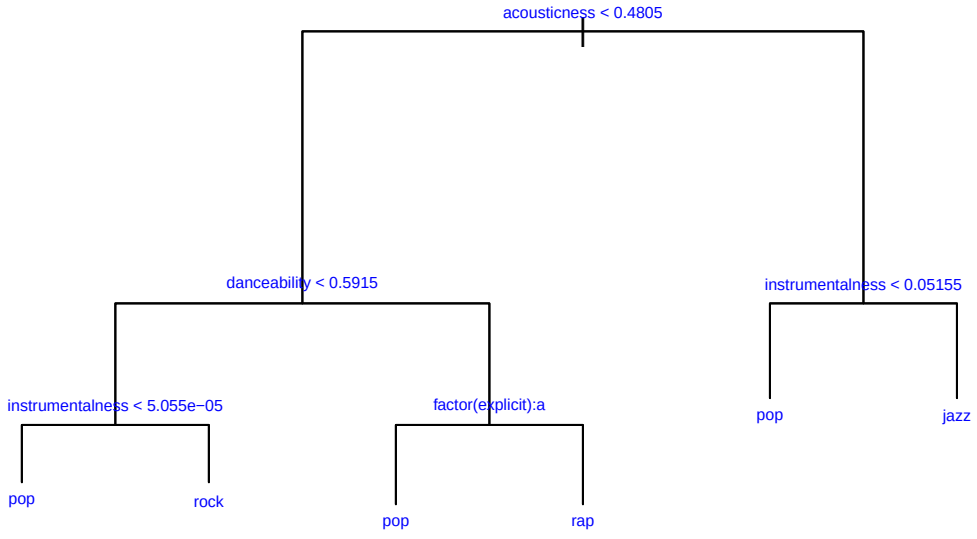
```r
strata_sizes <- strata$nh
```

```r
#perform the stratified sample
idx_strat <- sampling:::strata(genre_tracks, stratanames=c("genre"), size=strata_sizes, method="srswor")
train_genre <- genre_tracks[idx_strat$ID_unit,]
test_genre <- genre_tracks[-idx_strat$ID_unit,]
```

```r
#build tree genre
tree.genre <- tree(factor(genre)~duration_ms+factor(explicit)+danceability+energy+factor(key)+loudness+
summary(tree.genre)
```

```
##
## Classification tree:
## tree(formula = factor(genre) ~ duration_ms + factor(explicit) +
##     danceability + energy + factor(key) + loudness + factor(mode) +
##     speechiness + acousticness + instrumentalness + liveness +
##     valence + tempo, data = train_genre)
## Variables actually used in tree construction:
## [1] "acousticness"     "danceability"     "instrumentalness" "factor(explicit)"
## Number of terminal nodes:  6
## Residual mean deviance:  2.2 = 330000 / 150000
## Misclassification error rate: 0.392 = 58764 / 150001
```

```r
plot(tree.genre)
text(tree.genre, cex=0.5, col="blue")
```



```r
#build lda genre
lda.genre <- lda(factor(genre)~duration_ms+factor(explicit)+danceability+energy+factor(key)+loudness+fa
lda.genre
```

```
## Call:
## lda(factor(genre) ~ duration_ms + factor(explicit) + danceability +
##     energy + factor(key) + loudness + factor(mode) + speechiness +
##     acousticness + instrumentalness + liveness + valence + tempo,
##     data = train_genre)
##
## Prior probabilities of groups:
##  classical    country      disco electronic       funk       jazz        pop
##   0.012507   0.018733   0.004913   0.004713   0.003340   0.079426   0.540150
##        rap        rnb       rock       soul
##   0.039960   0.001607   0.287458   0.007193
##
## Group means:
##            duration_ms factor(explicit)1 danceability energy factor(key)1
## classical       254490         0.0005330       0.4125 0.3526      0.07196
## country         186634         0.0039146       0.5838 0.4510      0.05302
## disco           286246         0.0054274       0.7091 0.7151      0.06649
## electronic      297593         0.0212164       0.5838 0.6152      0.09760
## funk            254061         0.0459082       0.6630 0.7268      0.09980
## jazz            233199         0.0002518       0.5337 0.3679      0.05363
## pop             233172         0.0189946       0.5735 0.5671      0.06514
## rap             217738         0.4723056       0.7166 0.6835      0.15332
## rnb             267374         0.0456432       0.6400 0.5512      0.10373
## rock            241778         0.0314942       0.5161 0.6508      0.05427
## soul            219827         0.0176089       0.5926 0.5221      0.07600
##            factor(key)2 factor(key)3 factor(key)4 factor(key)5 factor(key)6
## classical       0.11087      0.06983      0.07676      0.09701      0.04478
## country         0.09858      0.04306      0.09715      0.09039      0.04448
## disco           0.07327      0.04071      0.05292      0.09769      0.07191
## electronic      0.07638      0.03253      0.08487      0.09335      0.07496
## funk            0.08782      0.01996      0.05788      0.07186      0.06587
## jazz            0.08595      0.05758      0.03962      0.16653      0.02761
## pop             0.11149      0.03792      0.08483      0.09265      0.05439
## rap             0.07508      0.02469      0.06623      0.06640      0.09476
## rnb             0.08714      0.03734      0.06639      0.10373      0.07054
## rock            0.13660      0.02233      0.10791      0.06276      0.04701
## soul            0.08619      0.03429      0.06858      0.10380      0.04356
##            factor(key)7 factor(key)8 factor(key)9 factor(key)10 factor(key)11
## classical       0.12633      0.07143      0.09328       0.07090       0.04051
## country         0.13416      0.05836      0.13060       0.09217       0.05658
## disco           0.11262      0.04749      0.12347       0.05427       0.09362
## electronic      0.10042      0.06365      0.12588       0.06082       0.08204
## funk            0.12375      0.05988      0.11976       0.08383       0.11178
## jazz            0.13774      0.09401      0.06832       0.08973       0.02636
## pop             0.12316      0.05450      0.10951       0.06240       0.07139
## rap             0.10060      0.08108      0.07207       0.08342       0.10110
## rnb             0.09544      0.13278      0.05394       0.07054       0.07054
## rock            0.13171      0.03734      0.15731       0.03931       0.07674
## soul            0.14365      0.06951      0.10380       0.08526       0.06580
```

```
##            loudness factor(mode)1 speechiness acousticness instrumentalness
## classical   -14.679        0.6253     0.07079       0.7458          0.43367
## country     -10.442        0.9078     0.07058       0.6511          0.04560
## disco        -9.128        0.5550     0.06249       0.1267          0.15546
## electronic  -10.794        0.5177     0.06430       0.2422          0.54172
## funk         -7.580        0.6008     0.11479       0.2930          0.12246
## jazz        -12.739        0.6386     0.06312       0.7632          0.26925
## pop          -8.954        0.6362     0.06264       0.4269          0.03892
## rap          -6.919        0.5192     0.19587       0.2233          0.01463
## rnb          -7.959        0.6763     0.05970       0.3183          0.01112
## rock         -9.186        0.6847     0.06622       0.2517          0.08454
## soul        -10.015        0.7099     0.06013       0.4201          0.04874
##            liveness valence tempo
## classical    0.1985  0.3793 112.4
## country      0.2064  0.6385 118.7
## disco        0.1878  0.7611 123.5
## electronic   0.1737  0.3979 121.0
## funk         0.3151  0.6505 127.7
## jazz         0.1931  0.5441 113.6
## pop          0.1993  0.5541 120.5
## rap          0.1955  0.5586 116.5
## rnb          0.1752  0.4881 115.7
## rock         0.2273  0.5390 123.8
## soul         0.1966  0.6291 117.8
##
## Coefficients of linear discriminants:
##                              LD1          LD2           LD3          LD4
## duration_ms       -0.0000002582  0.000000114  0.0000005944 -0.000003046
## factor(explicit)1  3.7279865107  1.982267092  1.5165339849  1.739520172
## danceability       1.7947536033  2.449003024 -2.0300094859 -4.824757979
## energy            -0.0237738164 -1.719079970  1.8107434031 -0.307922643
## factor(key)1       0.2400895995  0.253460606 -0.0415967793  0.038138864
## factor(key)2       0.1075220719 -0.252529338  0.0560092128  0.343115462
## factor(key)3       0.0128598509  0.253071461 -0.0399668986 -0.097642910
## factor(key)4       0.1748855687 -0.438804613  0.0388921060  0.572152177
## factor(key)5      -0.1174183691  0.337351045 -0.0219285736  0.017998730
## factor(key)6       0.2833639620 -0.047167977 -0.0927703442  0.018177348
## factor(key)7       0.0552061208 -0.038514158  0.0621428121  0.293999854
## factor(key)8      -0.0045685062  0.482492191  0.0119713842  0.019902225
## factor(key)9       0.0788305956 -0.435631359  0.1390299145  0.514876447
## factor(key)10      0.0529693175  0.334091955 -0.1134797337  0.225314752
## factor(key)11      0.2302291990 -0.250174654 -0.1048014888  0.266970347
## loudness           0.0387162556  0.073938124 -0.1399053655 -0.041078468
## factor(mode)1     -0.0433958246 -0.266894762  0.0110321859  0.735792024
## speechiness        3.7562695564  2.062790924  2.0786068350  2.036216432
## acousticness      -1.1698768726  2.073859796 -0.8865243560  0.554126784
## instrumentalness  -1.2624734702  1.597482797  3.2920894027 -1.744368113
## liveness          -0.0558907883 -0.121881212 -0.1460906530  0.285075643
## valence           -0.9533355110  0.181560834 -0.2547253305  1.775051708
## tempo             -0.0003882549 -0.000944489 -0.0026281337 -0.003896076
##                              LD5          LD6           LD7          LD8
## duration_ms        0.0000001879 -0.000002474  0.0000007108  0.000001855
## factor(explicit)1 -0.2502230150 -1.237151183 -0.7304504227 -0.317501476
## danceability       2.0136659211  2.514600251  0.2451457894  0.449211948
```

```
## energy            -0.7187662302  0.335757035  4.0851097319  2.961808576
## factor(key)1       0.0014484137  1.017036306 -0.2864013610  0.604338375
## factor(key)2      -0.1268118828  0.660153363  0.0282337602  0.013453408
## factor(key)3       0.0235853837  0.480249841 -0.3339556032  1.689622616
## factor(key)4       0.1537605546  1.642373191 -0.5653052877  0.449539971
## factor(key)5       0.2937116516 -0.066171349 -0.2027391212 -1.327111261
## factor(key)6       0.0966009557  1.437227402 -0.4161463194  1.385911358
## factor(key)7       0.1722775839  0.501785435 -0.0636413724 -0.292180838
## factor(key)8       0.1805979256  0.136525833 -0.4262379870 -1.220852816
## factor(key)9       0.4163184639  1.288613760 -0.1628750110  0.038534077
## factor(key)10      0.4789814625  1.034286982 -0.2294012775 -0.163483339
## factor(key)11      0.3032506975  1.466423559 -0.1043997081  0.777966944
## loudness          -0.0627300961  0.073006345 -0.0344444589 -0.169360595
## factor(mode)1      0.7459552882  1.243809464 -0.5376885828 -0.128936244
## speechiness       -0.0866058743  1.652877796  2.8701710528  1.204265132
## acousticness      -0.9389513496  0.681752112  2.3740618314  1.107397925
## instrumentalness   0.6765211295  1.434657065 -0.2114530182 -1.165474470
## liveness           1.1499894431  0.590462374  2.6351706095 -2.167625409
## valence            2.8960848802 -2.269814225 -0.2642831809  0.189283564
## tempo              0.0018748491  0.005843165  0.0067991646 -0.003220971
##                           LD9         LD10
## duration_ms       -0.000002201 -0.000005443
## factor(explicit)1  1.220965754 -0.592082180
## danceability       0.338713602 -0.688009152
## energy             2.629065676 -1.483239096
## factor(key)1      -1.908059968  1.374387438
## factor(key)2      -0.255512866 -0.115518888
## factor(key)3      -0.890060151 -0.172624585
## factor(key)4      -0.100798132  0.012866026
## factor(key)5      -0.000001716 -0.066035853
## factor(key)6      -0.877081226  0.162455246
## factor(key)7      -0.511455603  0.899776090
## factor(key)8      -1.958530289 -0.978655873
## factor(key)9       0.054276665  0.227157090
## factor(key)10     -1.225028100  1.359141260
## factor(key)11     -0.961286247  0.686701100
## loudness           0.007347111  0.021456905
## factor(mode)1     -0.089875143 -1.011791196
## speechiness       -2.740215458 -0.130746667
## acousticness       1.432718210 -0.947325633
## instrumentalness   0.776847898  0.549017091
## liveness          -2.560655599  0.494810480
## valence            0.018315994  0.545583443
## tempo              0.005731609  0.000050539
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4    LD5    LD6    LD7    LD8    LD9   LD10
## 0.4630 0.3078 0.1679 0.0329 0.0156 0.0068 0.0035 0.0014 0.0008 0.0003
```

```
#test the lda misclass

lda.predict<-predict(lda.genre, test_genre)
table <- table(lda.predict$class, test_genre$genre)
table
```

```
##
##              classical country disco electronic  funk  jazz   pop   rap   rnb
##   classical        248       7     1         30     0   391   224     0     0
##   country            0       0     0          0     0     0     0     0     0
##   disco              1       0     2          1     0     2    41     0     0
##   electronic        24       0    33        128    15   112   322    13     1
##   funk               0       0     0          0     0     0     1     0     0
##   jazz             124      51     2          7     3   968   578     2     3
##   pop              242     973   207         63   145  3033 26922  1106    85
##   rap                6       9     1          6     5    20   729  1149     3
##   rnb                0       0     0          0     0     0     0     0     0
##   rock              91      63    43         43    29   152  2998    83     3
##   soul               0       0     0          0     0     0     0     0     0
##
##               rock  soul
##   classical    201     0
##   country        0     0
##   disco          2     0
##   electronic   414     4
##   funk           0     0
##   jazz         193     7
##   pop         8925   346
##   rap          650    13
##   rnb            0     0
##   rock        6547    53
##   soul           0     0
```

```
misclass = 1 - sum(diag(table)) / sum(table)
misclass
```

```
## [1] 0.3894
```