Classification of Spotify Data

DATA 606 - W2023 Final Project

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

2023-02-12

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 Objective & Topic Importance

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:

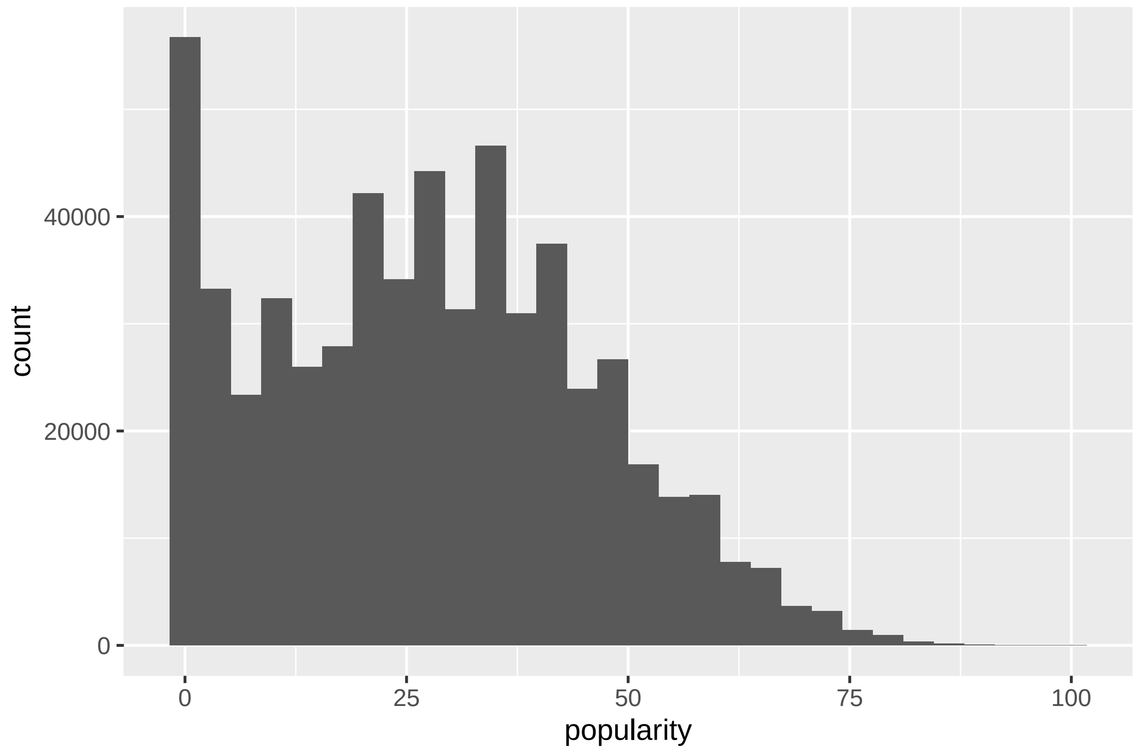
1.*duration_ms* - the length of the track in ms 2. *explicit* - explicit lyrics 3. *artists* - artist names 4. *danceability* - score for how suited a track is for dancing, 0.0-1.0. 5. *energy* - score for how energetic a track is perceived, 0.0-1.0. 6. *key* - maps Pitch class notation (E.g. 0 = C, 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 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.)

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

2.4 Data Cleaning

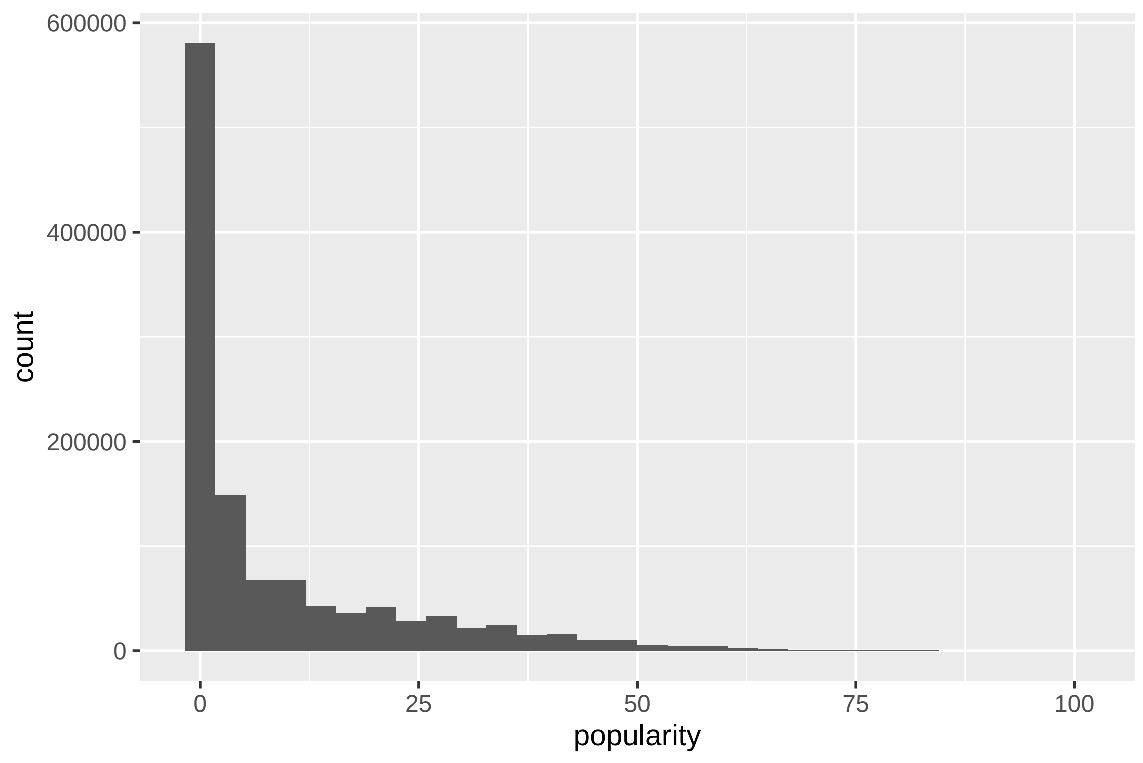
```
# head(artists_df) somthing here has characters that latex cant print
# head(tracks_df)

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
```

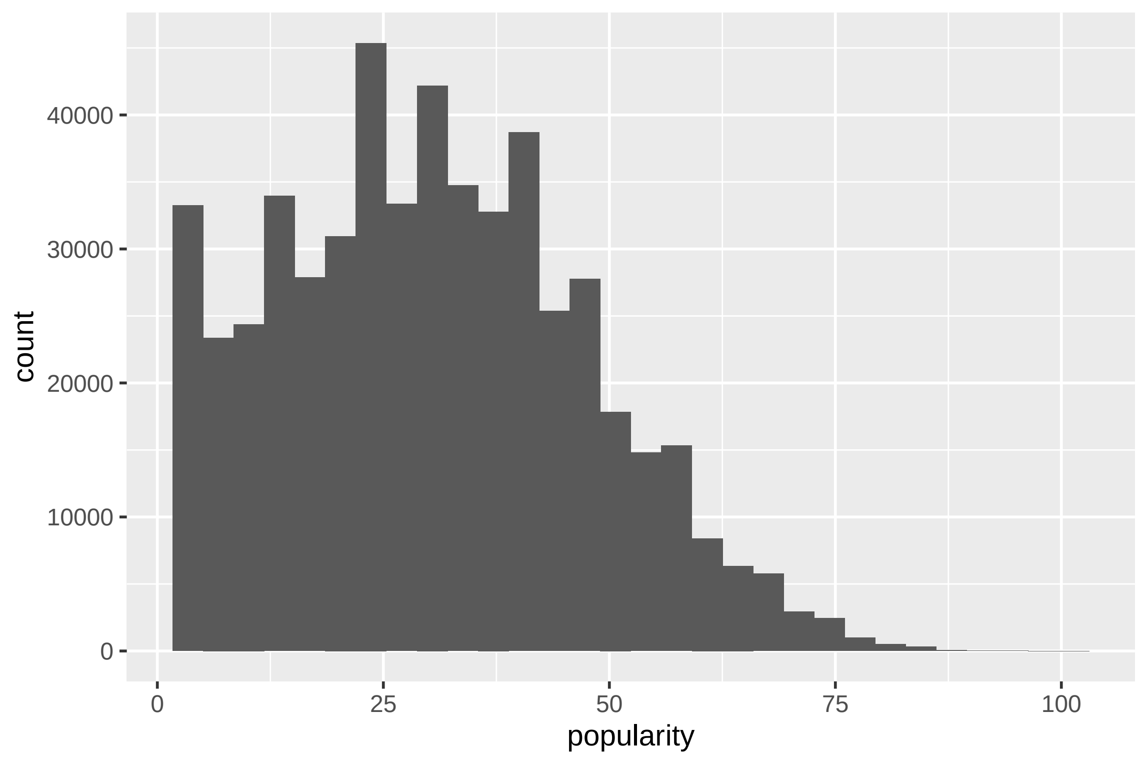
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
# 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)
```

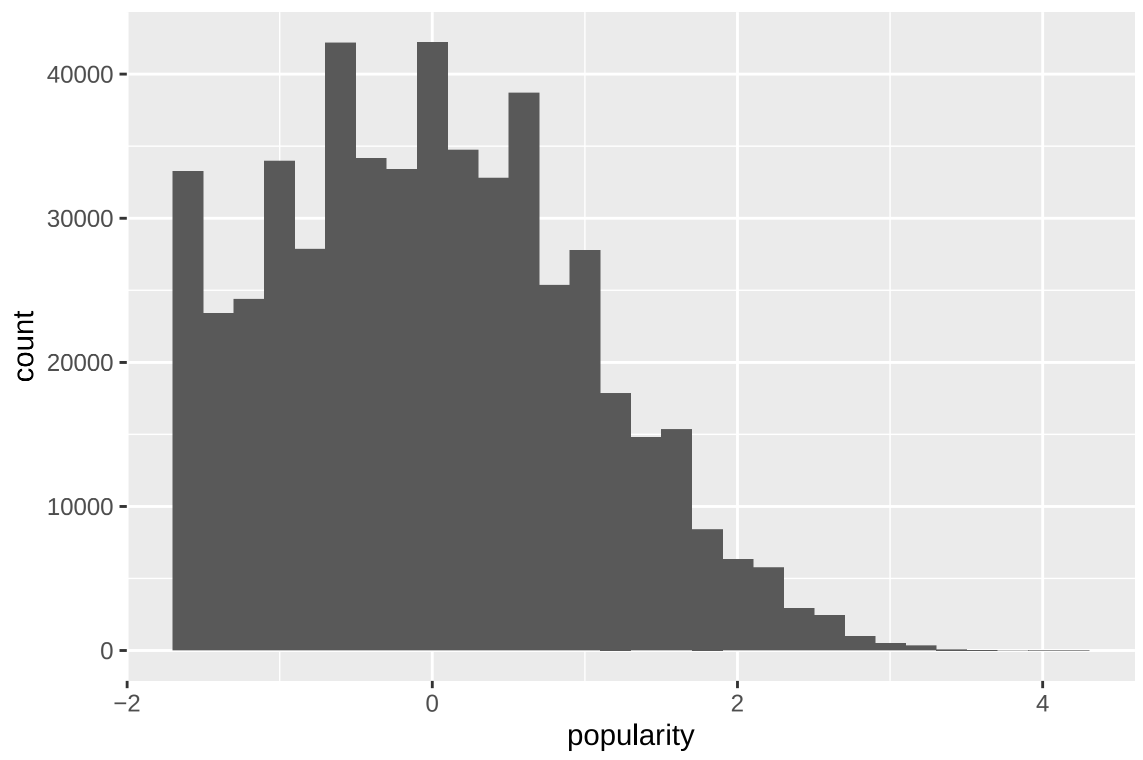
```
# Visualize filtered tracks popularity histogram
tracks_pop_filtered <- ggplot(data=popular_df, aes(x=popularity), title="Full Tracks Dataset Popularity
#artists_pop_filtered <- ggplot(data=artists_df, aes(x=popularity), title="Full Artists Dataset Popular
tracks_pop_scaled <- ggplot(data=popular_df2,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

2.5 Data Exploration

Correlation Matrix

```
dim(tracks_df)
```

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

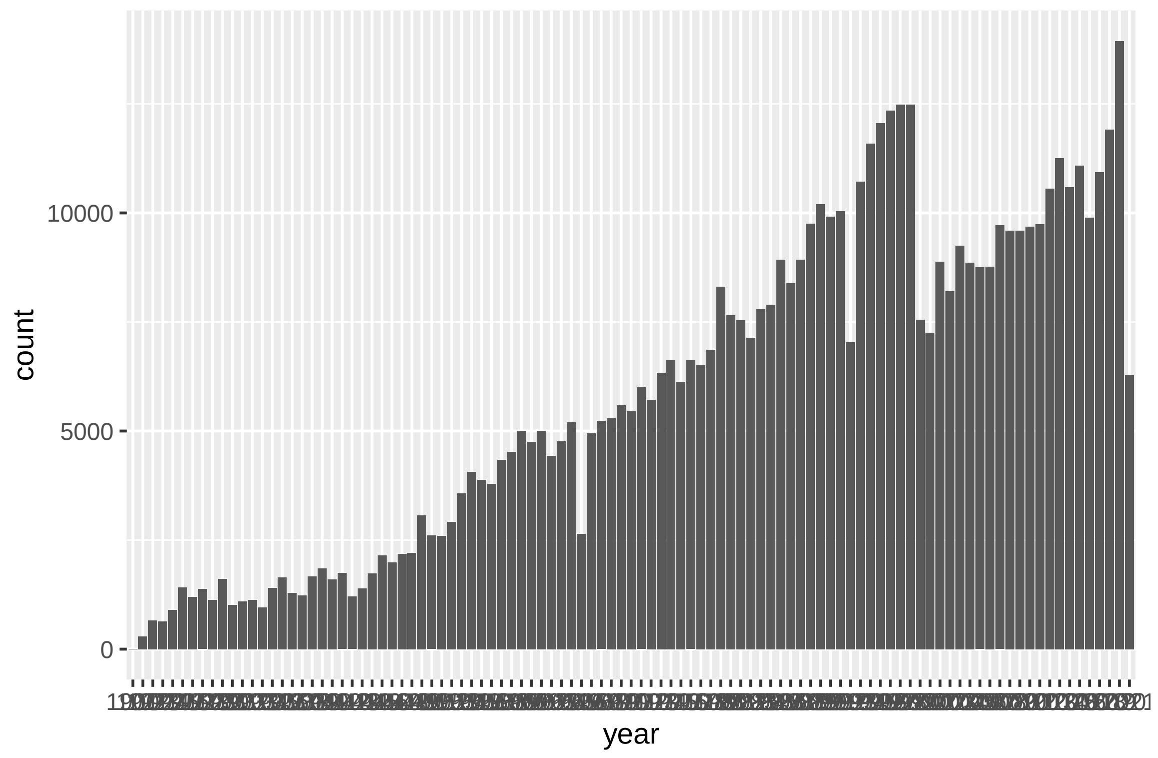
```
tracks_df$year <- eval(substr(tracks_df$release_date, 1,4))
```

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

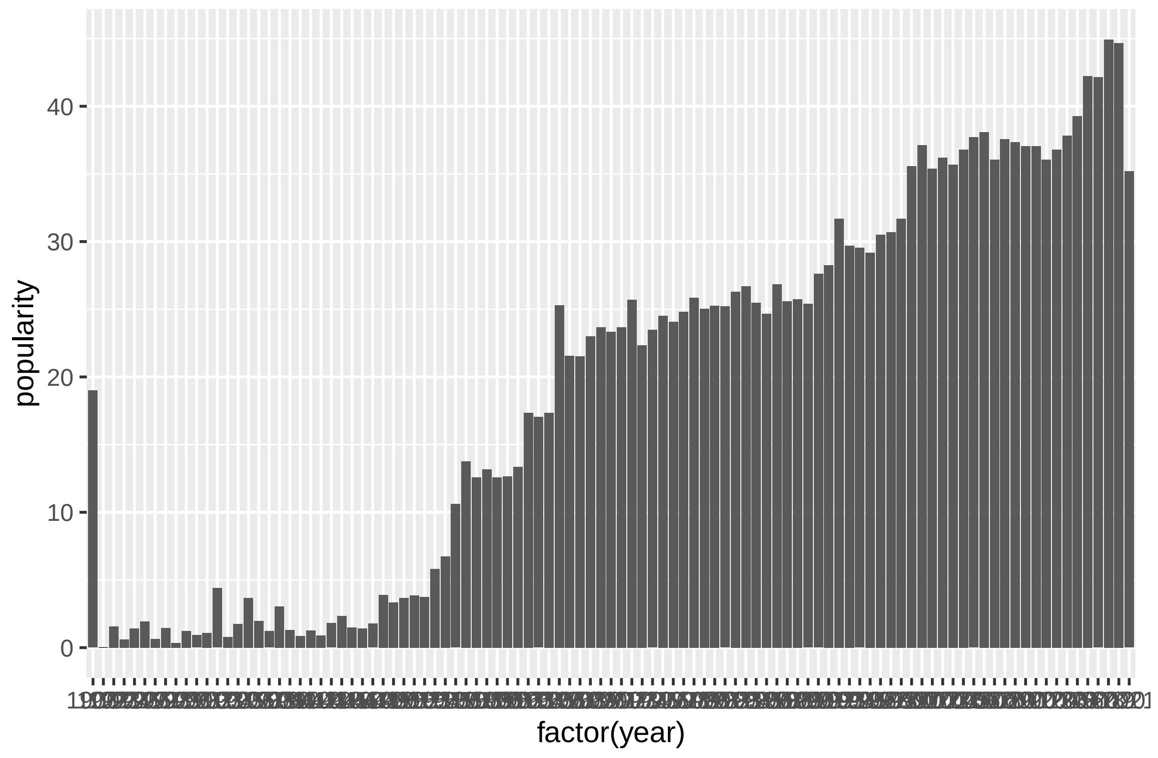
```
## [1] "1922"
```

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

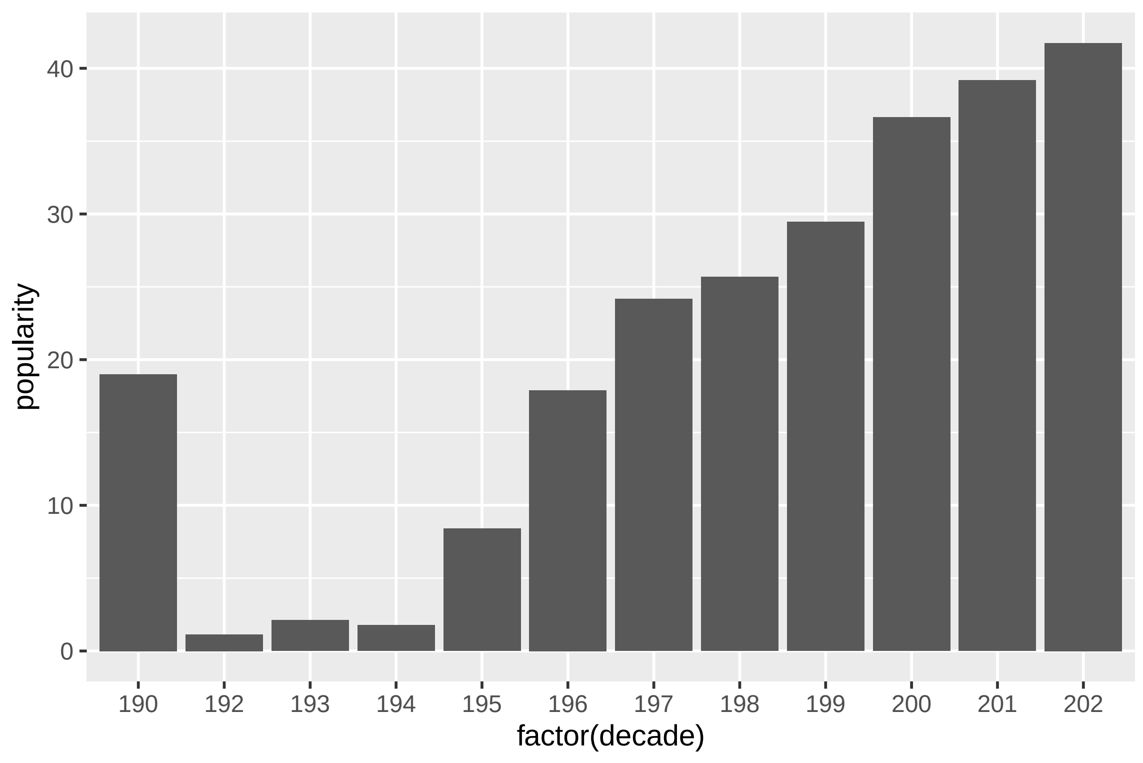
```
library(ggplot2)
ggplot(data = tracks_df, mapping = aes(x = year)) + geom_bar()
```



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

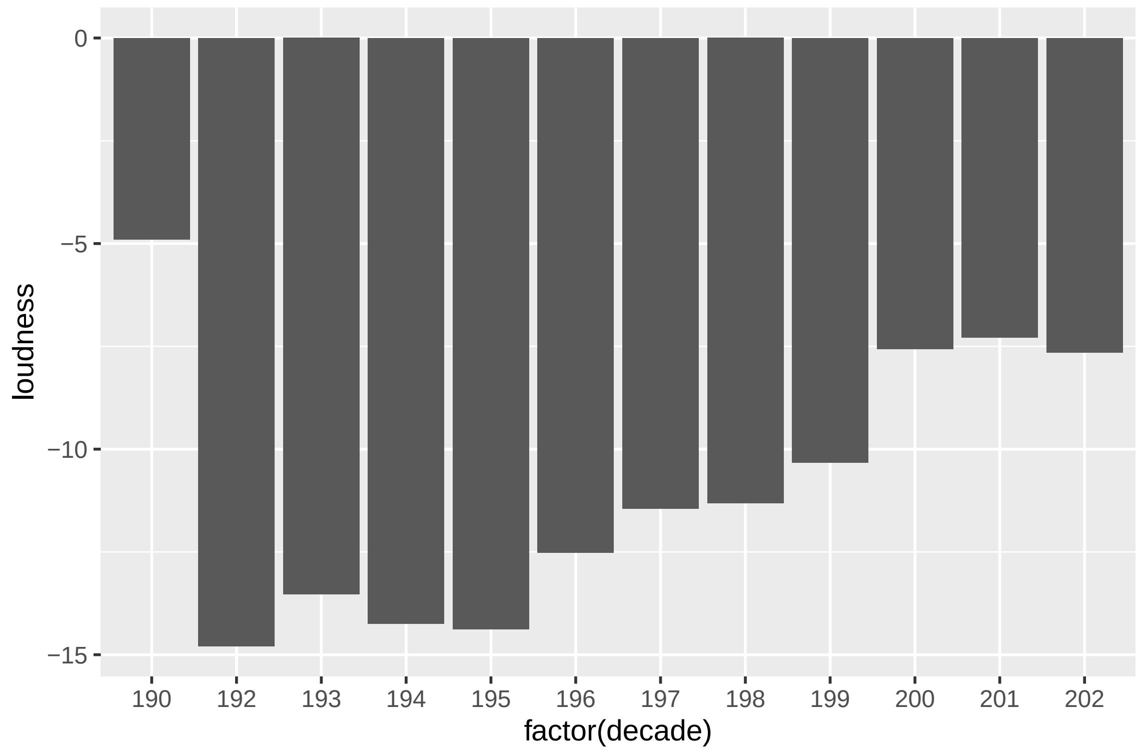


```r
ggplot(tracks_df, aes(x = factor(decade), y = popularity)) +
  geom_bar(stat = "summary", fun = "mean")
```

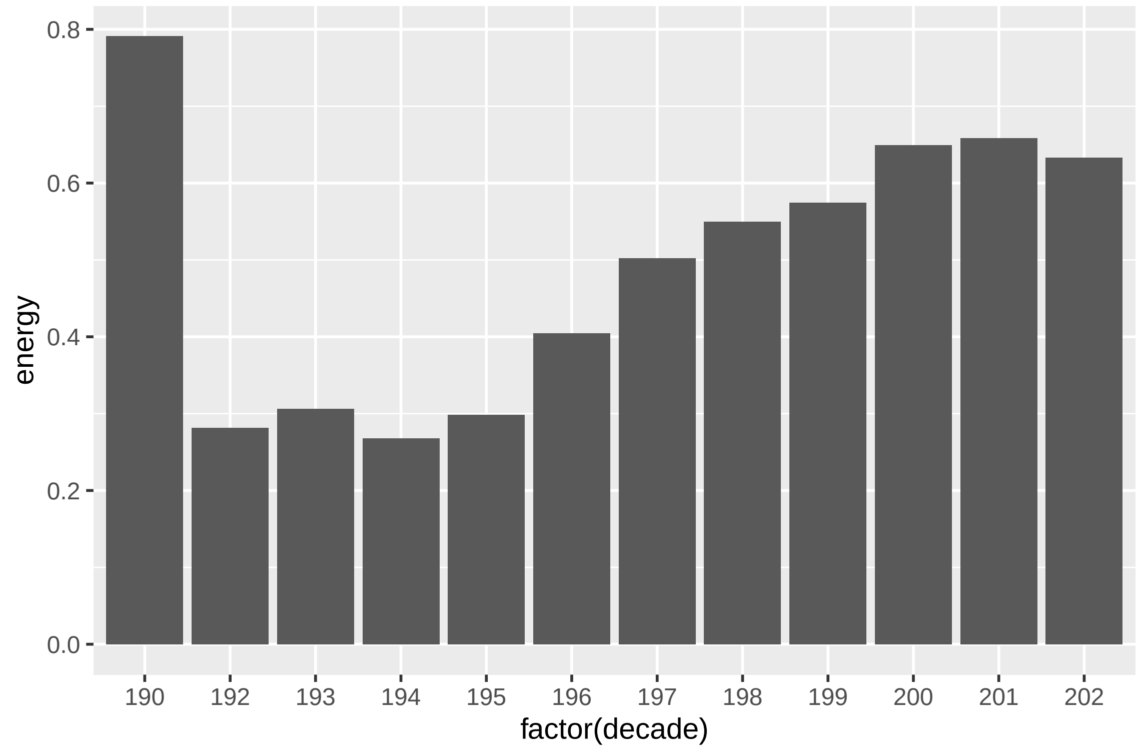


```
#+ guide_axis()
```

```
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")
```



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

```r
# dim(tracks_190)
```

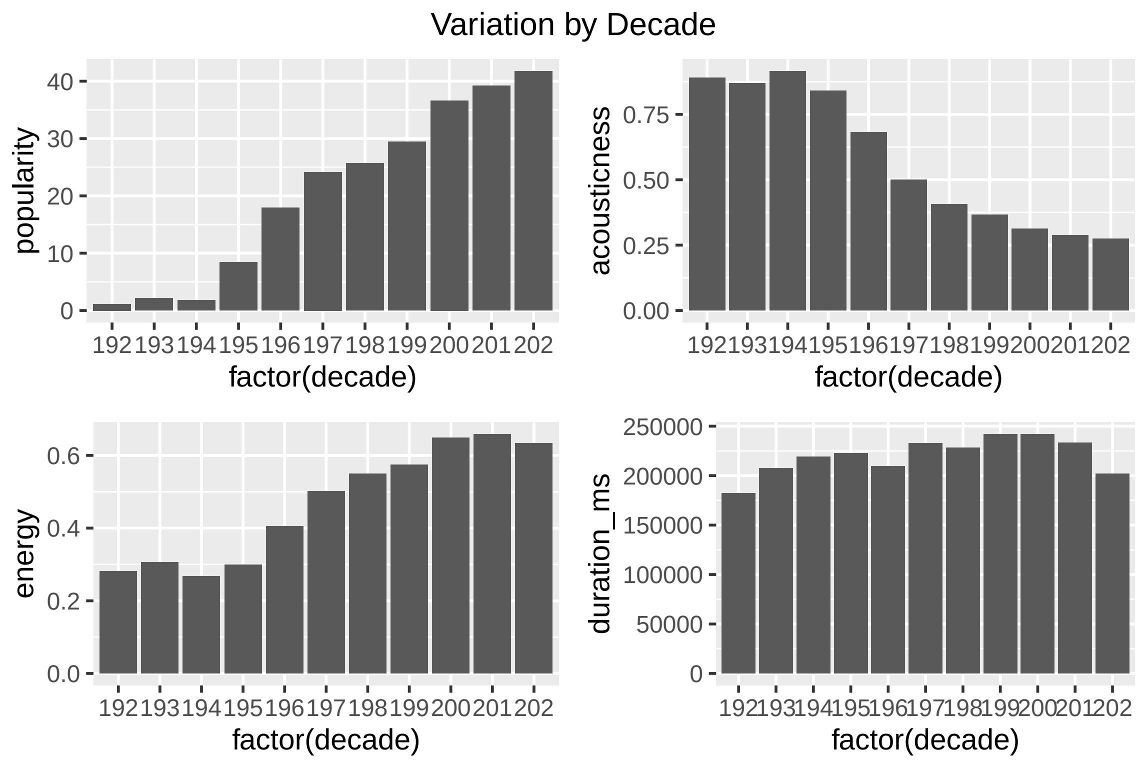
```r
tracks_df[478628,]
```

```
##                             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
```

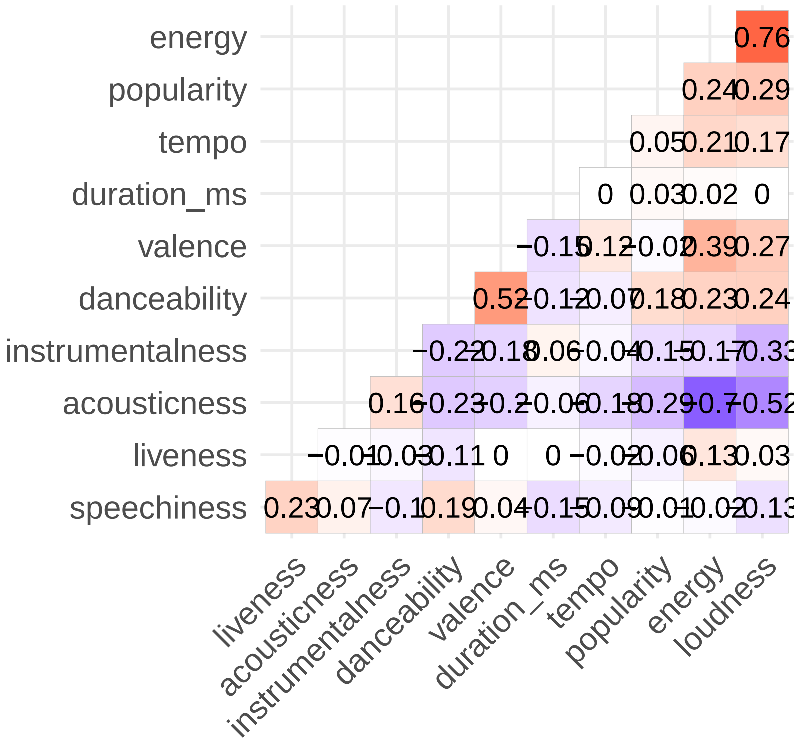
```r
tracks_clean <- tracks_df[-478628,]
```

```r
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")
```

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



```r
# 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
           )
```

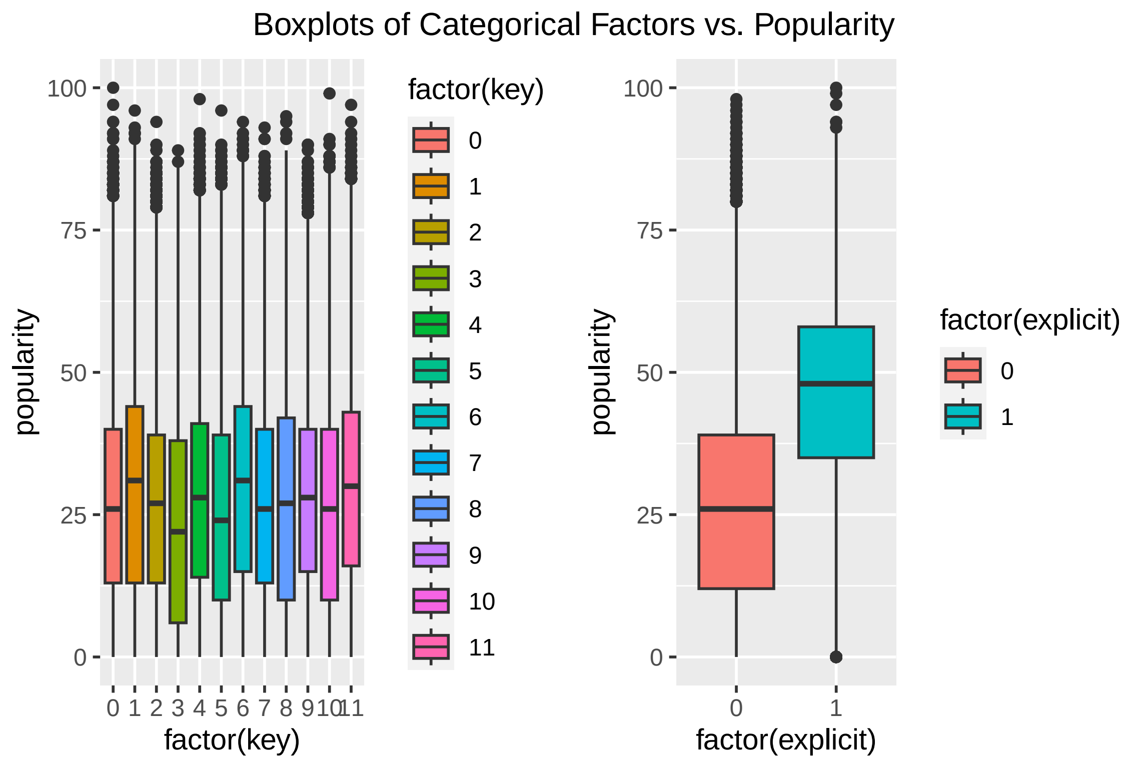


Popularity and Hit Boxplots

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:

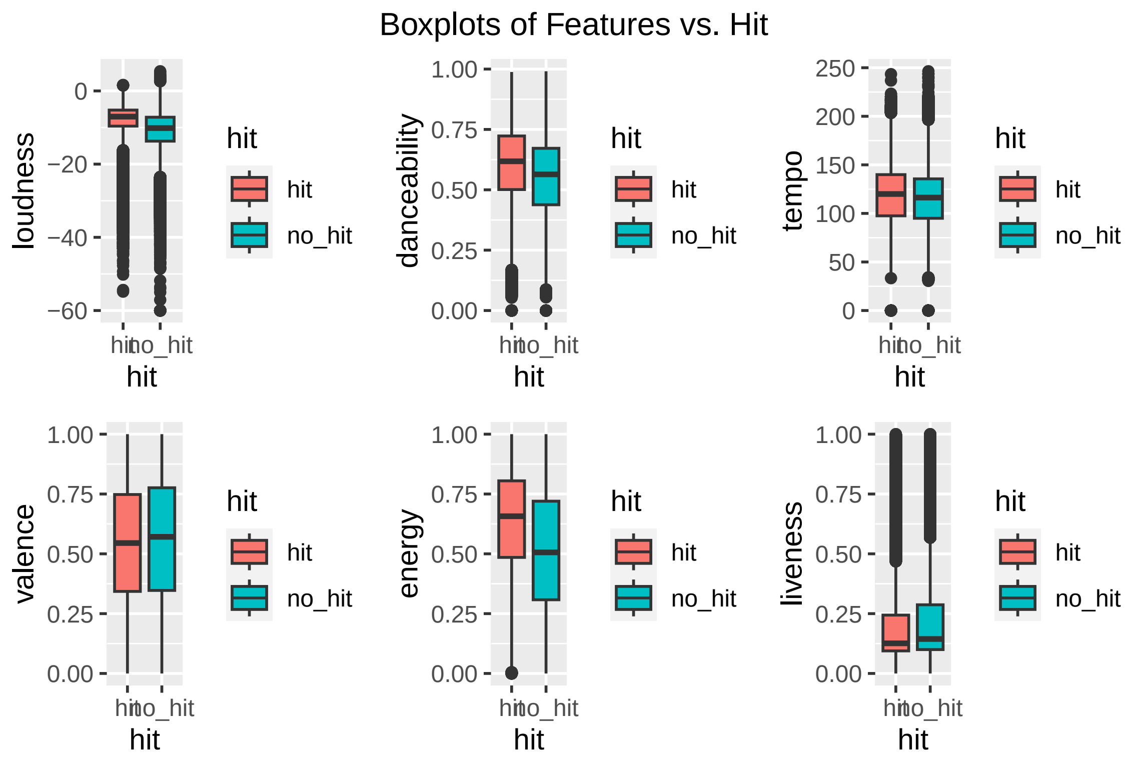
```
box1 <- ggplot(data = tracks_df, aes(x=factor(key), y=popularity, fill=factor(key))) + geom_boxplot()
box2 <- ggplot(data = tracks_df, aes(x=factor(explicit), y=popularity, fill=factor(explicit))) + geom_bo

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 popularity as an integer, another objective 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). To begin investigating this idea visually, we first classified a track as “hit” or “no_hit” based on being in the top quartile of popularity. Once this is done, we can produce boxplots of “hit” or “no_hit” for different features such as *loudness* and *danceability*:

```r
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".

3 Genre Classification

```
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)
```

4.1 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, 476963, 476962, 476962, 476962, 476963, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.7572    0.1722
```

4.1.1 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    44.42 7402.88     0
## Kurtosis   196.09   49.64     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  82    1.52  0.003 **
##       917
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.2 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
```

4.2.1 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)
```

4.3 Classification Tree

```
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, 476962, 476962, 476962, ...
## Resampling results across tuning parameters:
##
##   cp        Accuracy  Kappa
##   0.002342  0.7559    0.15523
##   0.006242  0.7528    0.11305
##   0.024959  0.7469    0.06357
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.002342.
```

4.3.1 Assumptions

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

4.4 Regression Tree

```
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, 476962, 476961, 476963, 476964, ...
## Resampling results across tuning parameters:
##
##   cp       RMSE    Rsquared  MAE
##   0.01833  0.9423  0.1121    0.7671
##   0.02033  0.9527  0.0924    0.7767
##   0.08470  0.9743  0.0837    0.7977
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was cp = 0.01833.
```

4.4.1 Assumptions

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

4.5 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/