```
---
header-includes:
  - \usepackage{booktabs}
#title: "Predicting Song Popularity on Spotify"
#author: "James B, Zach P, Matthew V, Nicholas B."
#date: "Semptember 10th, 2018"
output:
  pdf_document: default
  html_notebook:
    theme: lumen
    toc: yes
    toc_float: yes
  html_document:
    df_print: paged
    toc: yes
bibliography: bibliography.bib
---

# 1 Introduction

This century has so far been one of severe disruption for the recorded music industry. In 1999, the arrival of online file sharing service Napster heralded the beginning of the end for the recording industry's established business model. After peaking at \$21.5bn in 1999, US recorded music revenues collapsed to just $6.9bn in 2015.[^1]  In the last three years, however, revenues have started to rise again, driven by the explosive growth of music streaming services such as Apple Music and Spotify.

The surge in popularity of music streaming offers leading service providers a wealth data on consumer preferences and listening habits. By harnessing this newly available data, providers can improve user experience and maintain a competitive edge over rival platforms. An important feature for streaming services is customised music recommendations, and the effectiveness of these recommendations depends heavily on the ability to accurately predict a song's popularity.

This paper will aim to develop a model for predicting a song's popularity using data from Spotify. We consider eight potential predictor variables for a song's popularity, and after performing uni-variate and bi-variate analyses, seek fit an optimal linear model to the data. After selecting the model, we assess the validity of the model's assumptions, and then use the model to predict a song's popularity given new data.

[^1]: Recording Industry Association of America. *U.S. Sales Database* [Webpage]. Retrieved from https://www.riaa.com/u-s-sales-database/. Reported figures have been adjusted for inflation.

```{r Load Dependancies, warning=FALSE, message=FALSE, echo = FALSE}
library(tidyverse)
library(knitr)
library(kableExtra)
library(readxl)
library(broom)
library(reshape2)
library(modelr)
# Missing Data Visualisations
library(naniar)
```

```{r Global Chunk Options, echo = FALSE}
# Won't print warnings, messages or results to the output documents by default
knitr::opts_chunk$set(warning = FALSE, message = FALSE, results = 'hide', echo = FALSE, fig.pos = 'h')
options(knitr.table.format = 'markdown', width = 60)
```

```{r Loadd Custom Scripts}
# Ease to call basic plots for continuous and discrete univariate plots
source('Scripts//basicPlots.R')
# Upper, lower and interval confidences
source('Scripts//confInts.R')
# Johnos k fold cross val function
source('Scripts//crossValidate.R')
# Find the mode of a catagorical
source('Scripts//mode.R')

```

2 Data Decsription
```{r Data Load}
# Read in the excel spreadsheet and store in a dataframe
rawSpotifyData <- read_excel('Data//spotify.xlsx')

numberOfSamples = nrow(rawSpotifyData)

text_tbl <- data.frame(
  Types = c("**Identifiers**", "**Metrics**", "**Catagories**"),
  Variables = c(
    "Track Name, Artist",
    "Popularity, Danceability, Energy, Loudness, Duration",
    "Key, Mode, Time Signature, Decade"
  )
)
```
```

```
text_tbl <- kable(text_tbl) %>%
            kable_styling(full_width = F)

```


To obtain the data used for analysis a function was used that accesses Spotify's database and extracts the songs for a list of artists given to it. The decade variable is added manually to each of the artists. These artists where chosen to be a representation of their respective decade; namely Elvis Presley, The Beatles, David Bowie, Micheal Jackson, Blur and Beyonce, representing the 50s, 60s, 70, 80s, 90, and 00s respectively.

The raw data contains `r nrow(rawSpotifyData)` observations of `r ncol(rawSpotifyData)` variables. This initial data scrape is reduced in scope to the following variables:
`r text_tbl`

3 Cleaning
3.1 Loading and Type Assignment

Prior to analysis the data is loaded into the R studio environment and inspected to ensure that subsequent operations are applied to a consistent and uniform dataset. To achieve this, the variables that are out of scope in this analysis are removed. Each of the remaining categorical variables are inspected to ensure their levels are ordered correctly within each factor. Re-leveling was only necessary for the decade variable due to its chronological nature giving it an ordinal structure. The remaining factors' levels were left in alphabetical or numerical order as they are non-ordinal or this default ordering preserves their inherent order. Each of the newly created factor levels were then tabulated. Initially a time signature of zero was considered to be erroneous, but when reviewing the reference material it is not a known fixed quantity, as is with most sheet music. The value is estimated for the track by an algorithm, thus multiple rests or lack of percussion may lead to the estimated time signature falling bellow one and being rounded to zero. [^2] Due to this no remediation was used on the samples contained in this level.

[^2]: SpotifyAB, *Get Audio Features for a Track - Audio Features Object* [Webpage]. Retrieved from https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

```{r Removal of Other Variables}
# Only selecte the variable relevent to the analysis
rawSpotifyData <- rawSpotifyData %>%
  select(c('popularity','danceability', 'energy', 'key', 'loudness',
           'mode', 'duration_ms', 'time_signature', 'decade',
           'track_name', 'artist'))

```


```{r Classes of the Imported Data}
# Show each of the imported variables data-type
rawSpotifyData %>%
    summarise_all(funs(class(.))) %>%
    t()
```


```{r Re-map predictors to factors}
# Convert character variables to factors
convertToFactor = c('key', 'mode', 'decade', 'time_signature')

rawSpotifyData[convertToFactor] <- rawSpotifyData %>%
    select(convertToFactor) %>%
    lapply(factor)

# Relevel Decades as it is ornidal
rawSpotifyData$decade <- rawSpotifyData$decade %>%
  factor(levels = c('50s', '60s', '70s', '80s', '90s', '00s'))

# Sumarise the levels of each factor
rawSpotifyData[convertToFactor] %>%
  lapply(table)

```

3.2 Missing Values
Leveled factors and numerical variables are then subjected to missing value analysis to ensure samples are described completely. To do this, frequency counts for empty, non-applicable and null variable values across the dataset were extracted to produce a bar chart with each variable's respective missing value content shown as a percentage, \autoref{missing}. This analysis revealed that none of the variables that will be used in the analysis are missing any values.

As missing values in numerical data can often be encoded as zeros, these variables where tested for zero content. This revealed that the highest zero value content, 3.5%, is observed in the Popularity variable and 0.25% where found in the case of danceability. These low levels of zero content are likely appropriate for the given variables and are left unchanged.

```{r Frequency of NA and Zero, out.width = '0.49\\linewidth', figwidth = 3, figheight = 3, fig.show='hold', fig.align='center', results = 'show', fig.cap = "Bar charts showing the frequency of missing values (left) and zero
```

```
values (right) as a percentage of overall observations. \\label{missing}"}

# Plot a barchart showing NA for catagorical variables
miss_var_summary(rawSpotifyData) %>%
  #subset(pct_miss > 0.1) %>%
  ggplot(aes(x = reorder(variable, pct_miss), y = pct_miss)) +
  geom_bar(fill = "#599ad3", stat = "identity") +
  theme_minimal() + labs(x = "Variables", y = "Percent Missing") +
  coord_flip()

# Plot a barchart showing zeros for numerical data
rawSpotifyData %>%
    summarise_all(funs(sum(. == 0))) %>%
    mutate_all(funs(sum(100*(./numberOfSamples)))) %>%
    'rownames<-'('pct_zero') %>%
    t() %>%
    as.array() %>%
    subset(. > 0.1) %>%
    as_tibble(rownames = "id") %>%
    ggplot(aes(x = reorder(id, pct_zero), y = pct_zero)) +
    geom_bar(fill = "#599ad3", stat = "identity") +
    theme_minimal() + labs(x = "Variables", y = "Percent Zero") +
    coord_flip()

```

```{r Save as Cleaned}
cleanSpotifyData <- rawSpotifyData
```

The cleaned data has `r nrow(cleanSpotifyData)` observations of `r ncol(cleanSpotifyData)` variables. However, two of the remaining variables are retained for interpretation and labeling only, namely Artist and Track Name, leaving `r ncol(cleanSpotifyData)-2` possible predictors.

\clearpage

4. Description of Variables
4.1 Continuous Variables
Popularity appears to be bi-modal, with modes at 0 and 20. The mode present at zero might be due to a minimum threshold required before being rated on the interval from [0-100]. The majority of examples are present after the second mode giving the overall distribution a slight skew towards the right, \autoref{PopLoud} (left).

Loudness has a single mode at around -5 with most examples lying to the left of the mode. This skews the distribution to the left. The distribution has a low kurtosis, evidenced by the lack of a sharp mode, \autoref{PopLoud} (right).

```{r plot1, out.width = '0.49\\linewidth', figwidth = 3, figheight = 2, fig.show='hold', fig.align='center', results = 'show', fig.cap = "Histograms of the popularity (left) and loudness (right) variables imported from Spotify. \\label{PopLoud}"}

#              === Continuous Variables ===
# Plots a histogram of the numerical variable popularity
continuousPlot(cleanSpotifyData, cleanSpotifyData$popularity, "Popularity")

# Plots a histogram of the numerical variable loudness
continuousPlot(cleanSpotifyData, cleanSpotifyData$loudness, "Loudness (dB)")
```

Energy has an extremely low kurtosis, to the point that no easily identifiable mode. The examples are roughly uniformly distributed over the interval [0.3, 0.9] with a steep drop off in frequency either side of this interval, \autoref{EneDan} (left).

Danceability displays a distribution that is approximately normal. There is a single mode at 0.6 and the kurtosis of the distribution appears to be normal (~3), \autoref{EneDan} (right).

``` {r plot2, out.width = '0.49\\linewidth', figwidth = 3, figheight = 3, fig.show='hold',fig.align='center', results = 'show', fig.cap = "Histograms of the Energy (left) and Danceability (right) variables imported from Spotify. \\label{EneDan}"}

# Plots a histogram of the numerical variable enegery
continuousPlot(cleanSpotifyData, cleanSpotifyData$energy, "Energy")

# Plots a histogram of the numerical variable danceability
continuousPlot(cleanSpotifyData, cleanSpotifyData$danceability, "Danceability")
```

``` {r plot3, out.width = '0.49\\linewidth', figwidth = 3, figheight = 3, fig.show='hold', fig.align='center', results = 'show', wrapf = TRUE}
# Plots a histogram of the numerical variable enegery
```

```
continuousPlot(cleanSpotifyData, cleanSpotifyData$duration_ms, "Duration")
```

The figure above is a histogram of the Duration variable, which is the duration of songs in the Spotify dataset, measured in milliseconds. The distribution is unimodal with high kurtosis. There is a slight positive skew, with apparent outliers at the higher end of the distribution.

4.2 Catagorical Variables

\autoref{KeyMod} (left) indicates that Key has a strong mode in category C with other frequently sampled categories A, D and G. The Categories D#, F#, G# are the least likely key for a song to be in.

Mode shows a strong preference for songs to be written in a major key as apposed to a minor.

```` {r plot4, out.width = '0.49\\linewidth', figwidth = 3, figheight = 3, fig.show='hold',fig.align='center', results = 'show', fig.cap = "Bar charts of the Key (left) and Mode (right) variables imported from the Spotify set. \\label{KeyMod}"}
#             === Catagorical Variables ===
# Plots a bar chart of the catagorical variable Key
discretePlot(cleanSpotifyData, cleanSpotifyData$key, "Key")

# Plots a bar chart of the catagorical variable Mode
discretePlot(cleanSpotifyData, cleanSpotifyData$mode, "Mode")
```

\autoref{TimDec} (left) shows the number of tracks grouped by their respective time signatures. The most common time signature observed is 4 with about 1800 tracks The second most common time signature is 3 with around 250 tracks. Very few tracks have time signatures of 0, 1 and 5.

\autoref{TimDec} (right) shows the number of tracks with respect to each decade analysed. The most common decade that songs originated from is the 50s, followed by the 70s, with about 600 and 550 songs respectively. The 60s, 80s, 90s, and 00s all have approximately 150 to 250 songs each.

```` {r plot5, out.width = '0.49\\linewidth', figwidth = 3, figheight = 3, fig.show='hold',fig.align='center', results = 'show', fig.cap = "Barcharts of the Time Signature (left) and Decade (right) variables imported from the Spotify set. \\label{TimDec}"}
# Plots a bar chart of the catagorical variable Time Signature
discretePlot(cleanSpotifyData, cleanSpotifyData$time_signature, "Time signature")
# Plots a bar chart of the catagorical variable decade
discretePlot(cleanSpotifyData, cleanSpotifyData$decade, "Decade")
```

\vspace{10cm}

# 5. Bivariate Analysis

Bi-variate analysis is the simultaneous analysis of two variables to explore the relationship between them. This section contains bi-variate analyses of the Popularity variable against eight potential predictor variables from the `spotify` dataset.

### 5.1 Continuous Variables

\autoref{DurEner} (left) shows a scatter plot of the Duration variable against Popularity. There appears to be very little association between the two variables, with no discernible direction, with data points appearing to be scattered randomly about the mean of Popularity. Their is no detectable non-linearity in the distribution of the data either. Four potential outliers are seen to appear down field, having high levels of Duration.

\autoref{DurEner} (right) appears to show no evidence of an association between the two variables, with the data points randomly scattered about the mean of Popularity. There is no indication of a nonlinear trend, or any outliers.

```` {r plot6, out.width = '0.49\\linewidth', figwidth = 3, figheight = 3, fig.show='hold',fig.align='center', results = 'show', fig.cap = "Scatter plots of the Duration (left) and Energy (right) variables against Popularity. \\label{DurEner}"}

# Generate scatterplot of `duration_ms` against `popularity`
scatterPlot(cleanSpotifyData, cleanSpotifyData$duration_ms, cleanSpotifyData$popularity,
            "Duration", "Popularity", "(ms)")

# Generate scatterplot of `energy` against `popularity`
scatterPlot(cleanSpotifyData, cleanSpotifyData$energy, cleanSpotifyData$popularity,
            "Energy", "Popularity")
```

\clearpage

In \autoref{DanLou} (left) there appears to be a weak positive association between Popularity and Danceability, though there appears to be a strong concentration of samples around a popularity of 20 across a wide range of Danceability. There is no evidence of a nonlinear trend in the data, or any apparent outliers.

A weak positive association is also evident in \autoref{DanLou} (right) between Popularity and Loudness, with some indication of curvature in the relationship. Again a dense section of samples are seen around popularity's mean that

constitutes over a the range of values loudness has. Possible outliers may be present with lower levels of Loudness and higher Popularity scores. The variance of the Popularity score appears to be increasing for higher levels of Loudness.

```` {r plot7, out.width = '0.49\\linewidth', figwidth = 3, figheight = 3, fig.show='hold',fig.align='center', results = 'show', fig.cap = "Scatter plots of the Danceability (left) and Loudness (right) variables against Popularity. \\label{DanLou}"}
# Generate scatterplot of `danceability` against `popularity`
scatterPlot(cleanSpotifyData, cleanSpotifyData$danceability, cleanSpotifyData$popularity,
            "Danceability", "Popularity")

# Generate scatterplot of `loudness` against `popularity`
scatterPlot(cleanSpotifyData, cleanSpotifyData$loudness, cleanSpotifyData$popularity,
            "Loudness", "Popularity", "(dB)")
````

5.2 Categorical Variables

```{r plot9, out.width = '0.49\\linewidth', figwidth = 3, figheight = 3, fig.show='hold', fig.align='center', results = 'show', fig.cap = "Box plots of sample Key (left) and Mode (right) variables against Popularity. \\label{KeyMod}"}
# Box plot of songs grouped by their key
boxPlot(cleanSpotifyData, cleanSpotifyData$key, cleanSpotifyData$popularity, "Key", "Popularity")

# Box plot of songs grouped by their mode
boxPlot(cleanSpotifyData, cleanSpotifyData$mode, cleanSpotifyData$popularity, "Mode", "Popularity")

```

Across the series of keys there is very little variation in group mean and variance. Some groups are are slightly smaller in range than others, but the distributions appear uniform. The only aspect in which they differ greatly is the density of examples present in the different categories; the sharp keys all being particularly sparse, \autoref{KeyMod}.

```{r plot10, out.width = '0.49\\linewidth', figwidth = 3, figheight = 3, fig.show='hold', fig.align='center', results = 'show', fig.cap = "Box plots of sample Time Signatue (left) and Decade (right) variables against Popularity. \\label{KeyMod}"}
# Box plot of songs grouped by their key
boxPlot(cleanSpotifyData, cleanSpotifyData$time_signature, cleanSpotifyData$popularity, "Time signature",
"Popularity")
# Box plot of songs grouped by their decade
boxPlot(cleanSpotifyData, cleanSpotifyData$decade, cleanSpotifyData$popularity, "Decade", "Popularity")
```

There is not much difference in the popularity between tracks with time signatures 1, 3 and 5. Tracks with a time signature of 4 appear slightly more popular than those with other time signatures. Tracks with a time signature of 0 have about 0 popularity. This may occur when a track has no sound or length. The largest difference between the time signatures is in the number of samples with that respective time signature, \autoref{KeyMod} (left).

Across the decades, the 50s and the 70s have a particularly dense amount of observations clustered around their respective medians, while the 00s and the 80s data have more of a spread, with the 60s in between the two. The 60s have the highest median popularity level, while the 00s, 80s, and 70s to a lesser degree have the most popular individual songs, \autoref{KeyMod} (right).

5.3. Bivariate Analysis Between Predictors

```{r Corelation Matrix, out.width = '0.75\\linewidth', figwidth = 3, figheight = 3, fig.align='center'}
# Obtain correlation matrix for continuous predictor variables
cleanSpotifyData %>%
  Filter(is.numeric, .) %>%
  cor() %>%
  round(2) %>%
  melt() %>%
  # Produce correlation matrix plot
  ggplot( ., aes(x=Var1, y=Var2, fill = value) ) +
    geom_tile() +
    geom_text( aes( Var2, Var1, label=value ), color = "white", size = 4 ) +
    theme_minimal() +
    theme( axis.title.x = element_blank(),
           axis.title.y = element_blank(),
           axis.ticks = element_blank(),
           legend.position = "none"
         )

```

The correlation matrix above indicates a fairly strong correlation between the Energy and Loudness variables. No other variables look to have significant associations with one another.

A one

\clearpage

```{r Anova}
# Cleaner way of selecting and getting data type names
cat_var_names <- cleanSpotifyData %>%
        Filter(is.factor, .) %>%
        colnames()

# Function to do the ANOVA of each term and second order interactions
anovaOut <- cleanSpotifyData %>%
  select(-c("artist", "track_name")) %>%
  lm(popularity ~ (. + .)^2, .) %>%
  anova()

# Can't find a good way to pipe this
colnames(anovaOut) <- c('DF', 'Sum Sq', 'Mean sq', "F Value", "P")

# Can pipe with dyplr methods arrange, but converts output
# to a tibble which drops row names and prints less nicely

# Select all P values above 0.05
anovaOut <- subset(anovaOut, P < 0.05)
anovaOut <- anovaOut[order(anovaOut$P),]

anovaOut

```

`r kable(anovaOut, digits = 3, caption = "ANOVA table showing all single variable and second order interations that are sigificant at the 0.05 level \\label{anova}", format = "latex", booktabs = T)`

\autoref{anova} was used to inform the scopes used in the subsequent model fitting processes. A specific scope was created with the terms identified as significant in the table. This was done to limit the available terms during modeling to those that already have a significant relationship between them and the predictor, reducing potential model complexity and computation. Additionally this excludes a large number of interaction terms that may otherwise be included in the model.

\clearpage

6. Model Fitting

To fit the linear model, forward, backward and step-wise algorithms for model selection were all used, with both the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) as heuristics. Each of the model selection algorithms iteratively adds and/or removes terms using appropriate measures of significance, until the optimal model has been identified. The model with all terms available for inclusion is called the full model. The model with the least possible terms is the null model.

The **forward** algorithm begins with the null model. At each iteration, the P-value is calculated for each term that is not currently included in the model. If the smallest P-value is less than a threshold (in our case 0.05), the term with that P-value is added to the model. The **backward** algorithm begins with the full model, and at each iteration, calculates the P-values for all terms in the model. If the highest P-value exceeds the chosen threshold, that term is removed from the model. This is repeated until all included terms are significant. The **stepwise** algorithm begins with the null model. At each iteration, one step of forward selection is performed, using a liberal P-value such as 0.20, and one step of backward elimination is performed using a lower P-value such as 0.05. This is repeated until no further changes occur in the model.

Three different full models were considered. The first full model included all predictors but no interaction terms. The second used only those terms, either individual predictors or interactions between them, that were identified as significant in the previous section. The third included all predictor terms and all interaction terms. The criteria scores and k-fold mean square error for models obtained using each algorithm and heuristic are presented on Tables 3 and 4.


```{r fitting, cache = TRUE}
# Get reduced dataset
reducedSpotify <- select(cleanSpotifyData, popularity:decade)

# Define smallest model
smallest <- popularity ~ 1

# Define the full model 1
full_1 <- popularity ~ decade + danceability + loudness + energy + duration_ms + key + time_signature + decade

# full_2 <- popularity ~ decade + danceability + loudness + energy + duration_ms + energy:duration_ms +
danceability:energy + energy:decade + duration_ms:decade

# Define full model 2
full_2 <- popularity ~ decade + danceability + loudness + energy + duration_ms + energy:duration_ms +
danceability:energy + energy:decade + duration_ms:decade + energy:loudness + key:mode + mode:decade + key +
mode:duration_ms + danceability:decade + danceability:loudness + loudness:decade + danceability:key + key:decade

# Define full model 3
full_3 <- popularity ~ decade + danceability + loudness + energy + energy:duration_ms + danceability:energy +
duration_ms + energy:decade + duration_ms:decade + energy:loudness + key:mode + mode:decade + key + mode:duration_ms
```

```
+ danceability:decade + danceability:loudness + loudness:decade + danceability:key + key:decade + energy:mode +
loudness:time_signature + time_signature + danceability:duration_ms + key:loudness + key:duration_ms +
mode:time_signature + loudness:duration_ms + loudness:mode + energy:key + danceability:time_signature +
energy:time_signature + mode + key:time_signature + duration_ms:time_signature + time_signature:decade +
danceability:mode

lm_full_1 <- lm(full_1, data=reducedSpotify)
lm_full_2 <- lm(full_2, data=reducedSpotify)
lm_full_3 <- lm(full_3, data=reducedSpotify)


# AIC Heuristic
lm_1_backward <- step(lm_full_1, direction="backward",trace=0)
lm_1_forward <- step(lm(smallest, data=reducedSpotify), scope=full_1, direction = "forward",trace=0)
lm_1_step <- step(lm_full_1, direction="both",trace=0)


lm_2_backward <- step(lm_full_2, direction="backward",trace=0)
lm_2_forward <- step(lm(smallest, data=reducedSpotify), scope=full_2, direction = "forward",trace=0)
lm_2_step <- step(lm_full_2, direction="both",trace=0)

lm_3_backward <- step(lm(full_3, data=reducedSpotify), direction="backward",trace=0)
lm_3_forward <- step(lm(smallest, data=reducedSpotify), scope=full_3, direction = "forward",trace=0)
lm_3_step <- step(lm(full_3, data=reducedSpotify), direction="both",trace=0)

#BIC Heuristic ( set k=log(n) )

n <- nrow(rawSpotifyData)

lm_1_backward_BIC <- step(lm_full_1, direction="backward", k = log(n), trace=0)
lm_1_forward_BIC <- step(lm(smallest, data=reducedSpotify), scope=full_1, direction="forward", k = log(n),trace=0)
lm_1_step_BIC <- step(lm_full_1, direction="both", k = log(n), trace=0)

lm_2_backward_BIC <- step(lm_full_2, direction="backward", k = log(n), trace=0)
lm_2_forward_BIC <- step(lm(smallest, data=reducedSpotify), scope=full_2, direction="forward", k = log(n),trace=0)
lm_2_step_BIC <- step(lm_full_2, direction="both", k = log(n), trace=0)

lm_3_backward_BIC <- step(lm_full_3, direction="backward", k = log(n), trace=0)
lm_3_forward_BIC <- step(lm(smallest, data=reducedSpotify), scope=full_3, direction="forward", k = log(n), trace=0)
lm_3_step_BIC <- step(lm_full_3, direction="both", k = log(n), trace=0)

```



```{r}
# === AIC HUERISTIC MODELS ===

# Calculate AIC for each model
backward_1_AIC <- extractAIC(lm_1_backward)[2]
forward_1_AIC <- extractAIC(lm_1_forward)[2]
step_1_AIC <- extractAIC(lm_1_step)[2]
backward_2_AIC <- extractAIC(lm_2_backward)[2]
forward_2_AIC <- extractAIC(lm_2_forward)[2]
step_2_AIC <- extractAIC(lm_2_step)[2]
backward_3_AIC <- extractAIC(lm_3_backward)[2]
forward_3_AIC <- extractAIC(lm_3_forward)[2]
step_3_AIC <- extractAIC(lm_3_step)[2]

# Calculate BIC for each
backward_1_BIC <- BIC(lm_1_backward)
forward_1_BIC <- BIC(lm_1_forward)
step_1_BIC <- BIC(lm_1_step)
backward_2_BIC <- BIC(lm_2_backward)
forward_2_BIC <- BIC(lm_2_forward)
step_2_BIC <- BIC(lm_2_step)
backward_3_BIC <- BIC(lm_3_backward)
forward_3_BIC <- BIC(lm_3_forward)
step_3_BIC <- BIC(lm_3_step)

# Calculate cross validation for each model
backward_1_CV <- crossValidate( formula(lm_1_backward), reducedSpotify, 6 )
forward_1_CV <- crossValidate( formula(lm_1_forward), reducedSpotify, 6 )
step_1_CV <- crossValidate( formula(lm_1_step), reducedSpotify, 6 )
backward_2_CV <- crossValidate( formula(lm_2_backward), reducedSpotify, 6 )
forward_2_CV <- crossValidate( formula(lm_2_forward), reducedSpotify, 6 )
step_2_CV <- crossValidate( formula(lm_2_step), reducedSpotify, 6 )
backward_3_CV <- crossValidate( formula(lm_3_backward), reducedSpotify, 6 )
forward_3_CV <- crossValidate( formula(lm_3_forward), reducedSpotify, 6 )
step_3_CV <- crossValidate( formula(lm_3_step), reducedSpotify, 6 )

# === BIS HUERISTIC MODELS ===
# Calculate AIC for each model
backward_1_AIC_B <- extractAIC(lm_1_backward_BIC)[2]
forward_1_AIC_B <- extractAIC(lm_1_forward_BIC)[2]
step_1_AIC_B <- extractAIC(lm_1_step_BIC)[2]
```

```
backward_2_AIC_B <- extractAIC(lm_2_backward_BIC)[2]
forward_2_AIC_B <- extractAIC(lm_2_forward_BIC)[2]
step_2_AIC_B <- extractAIC(lm_2_step_BIC)[2]
backward_3_AIC_B <- extractAIC(lm_3_backward_BIC)[2]
forward_3_AIC_B <- extractAIC(lm_3_forward_BIC)[2]
step_3_AIC_B <- extractAIC(lm_3_step_BIC)[2]

# Calculate BIC for each
backward_1_BIC_B <- BIC(lm_1_backward_BIC)
forward_1_BIC_B <- BIC(lm_1_forward_BIC)
step_1_BIC_B <- BIC(lm_1_step_BIC)
backward_2_BIC_B <- BIC(lm_2_backward_BIC)
forward_2_BIC_B <- BIC(lm_2_forward_BIC)
step_2_BIC_B <- BIC(lm_2_step_BIC)
backward_3_BIC_B <- BIC(lm_3_backward_BIC)
forward_3_BIC_B <- BIC(lm_3_forward_BIC)
step_3_BIC_B <- BIC(lm_3_step_BIC)

# Calculate cross validation for each model
backward_1_CV_B <- crossValidate( formula(lm_1_backward_BIC), reducedSpotify, 6 )
forward_1_CV_B <- crossValidate( formula(lm_1_forward_BIC), reducedSpotify, 6 )
step_1_CV_B <- crossValidate( formula(lm_1_step_BIC), reducedSpotify, 6 )
backward_2_CV_B <- crossValidate( formula(lm_2_backward_BIC), reducedSpotify, 6 )
forward_2_CV_B <- crossValidate( formula(lm_2_forward_BIC), reducedSpotify, 6 )
step_2_CV_B <- crossValidate( formula(lm_2_step_BIC), reducedSpotify, 6 )
backward_3_CV_B <- crossValidate( formula(lm_3_backward_BIC), reducedSpotify, 6 )
forward_3_CV_B <- crossValidate( formula(lm_3_forward_BIC), reducedSpotify, 6 )
step_3_CV_B <- crossValidate( formula(lm_3_step_BIC), reducedSpotify, 6 )
```

```{r Model Summary AIC}

# Store model selection measures in dataframe
model_summary <- data.frame(
  c(backward_1_AIC,forward_1_AIC,step_1_AIC,backward_2_AIC,forward_2_AIC,
    step_2_AIC,backward_3_AIC,forward_3_AIC,step_3_AIC),
  c(backward_1_BIC,forward_1_BIC,step_1_BIC,backward_2_BIC,forward_2_BIC,
    step_2_BIC,backward_3_BIC,forward_3_BIC,step_3_BIC),
  c(backward_1_CV,forward_1_CV,step_1_CV,backward_2_CV,forward_2_CV,
    step_2_CV,backward_3_CV,forward_3_CV,step_3_CV)
  ) %>%
  lapply(round, 1)

model_summary <- data.frame(
  c("Backward", "Forward", "Step","Back", "Forward", "Step",
    "Back", "Forward", "Step"),
  model_summary)

# Label columns and rows
colnames(model_summary) <- c("Algorithm", "AIC",
                             "BIC", "k-Fold MSE")

model_summary <- t(model_summary)

kableModelsAIC <- model_summary %>%
  kable(digits = 1, caption = "Summary of metrics used to assess the different
        models produced from various scopes using different methods. Each method
        used the Aikaike Information Criterion as a heuristic. \\label{AICModles}",
        format = "latex", booktabs = T) %>%
  add_header_above(c(" " = 1, "No Interactions" = 3, "Only Significant Terms" = 3, "All Terms" = 3)) %>%
  add_header_above(c(" " = 1, "Heuristic - Akaike Information Criterion" = 9))

```


`r kableModelsAIC`


```{r Model Summary BIC}

# Store model selection measures in dataframe
model_summary <- data.frame(
  c(backward_1_AIC_B, forward_1_AIC_B, step_1_AIC_B, backward_2_AIC_B, forward_2_AIC_B,
    step_2_AIC_B, backward_3_AIC_B, forward_3_AIC_B, step_3_AIC_B),
  c(backward_1_BIC_B, forward_1_BIC_B, step_1_BIC_B, backward_2_BIC_B, forward_2_BIC_B,
    step_2_BIC_B, backward_3_BIC_B, forward_3_BIC_B, step_3_BIC_B),
  c(backward_1_CV_B, forward_1_CV_B, step_1_CV_B,backward_2_CV_B, forward_2_CV_B,
    step_2_CV_B, backward_3_CV_B, forward_3_CV_B, step_3_CV_B)
  ) %>%
  lapply(round, 1)

model_summary <- data.frame(
  c("Backward", "Forward", "Step","Back", "Forward", "Step",
    "Back", "Forward", "Step"),
  model_summary)
```

```
# Label columns and rows
colnames(model_summary) <- c("Algorithm", "AIC",
                             "BIC", "k-Fold MSE")

model_summary <- t(model_summary)

kableModelsBIC <- model_summary %>%
  kable(digits = 1, caption = "Summary of metrics used to assess the different
        models produced from various scopes using different methods. Each method
        used the Bayesian Information Criterion as a heuristic. \\label{BICModles}",
        format = "latex", booktabs = T) %>%
  add_header_above(c(" " = 1, "No Interactions" = 3, "Only Significant Terms" = 3, "All Terms" = 3)) %>%
  add_header_above(c(" " = 1, "Heuristic - Bayesian Information Criterion" = 9))

```

`r kableModelsBIC`

```{r Model Formulas}
sprintf("AIC")
formula(lm_1_backward)
formula(lm_1_forward)
formula(lm_1_step)

formula(lm_2_backward)
formula(lm_2_forward)
formula(lm_2_step)

formula(lm_3_backward)
formula(lm_3_forward)
formula(lm_3_step)

sprintf("BIC")
formula(lm_1_backward_BIC)
formula(lm_1_forward_BIC)
formula(lm_1_step_BIC)

formula(lm_2_backward_BIC)
formula(lm_2_forward_BIC)
formula(lm_2_step_BIC)

formula(lm_3_backward_BIC)
formula(lm_3_forward_BIC)
formula(lm_3_step_BIC)
```

\clearpage

Using the first full model, where no interaction terms were included, led to the selection of the same best model irrespective of algorithm or heuristic. The AIC for this model was the highest overall. The BIC was somewhat lower, though not the lowest overall. The k-fold MSE scores were the highest. Given these results, this model was a poor candidate for final selection.

For the remaining two models, we found that the best models obtained from using the BIC heuristic contained substantially fewer terms than the those obtained under AIC. Despite being much smaller models, those obtained under the BIC heuristic were not penalised with higher significantly higher AIC or MSE scores. Therefore, it was determined that the final model would be selected from among those obtained using the BIC heuristic. Of these, the forward algorithm produced models with higher MSE than those of backward and step-wise. For the two remaining full models, backward and step-wise algorithms arrived at the same best model in each case. Of these two models, there was very little difference in MSE, so it was determined that the simplest model would be chosen as the final model.

7. Final Model

The results from the selection process indicate that the model obtained using the 'only significant terms' full model, then applying the backward (or step-wise) algorithm with the BIC heuristic, was the optimal choice. The formula for this model is:

`r format(formula(lm_2_step_BIC))`

The interaction terms included in this model were judged to be meaningful and therefore reasonable to include in the model. Longer songs may be expected to exhibit a higher level of energy compared to shorter songs. Likewise, it is feasible that loudness of songs has changed in different decades, as decade is a proxy for Artist.

The coefficients for this model are presented in Table 5. The coefficients are the predicted change in popularity for an increase of 1 in the predictor term.

```{r Final Model}
# Output final model formula
coefficients <- tidy(lm_2_step_BIC)[1:2]

kableCoefficients <- coefficients %>%
  kable(digits = 3, caption = "Final model coefficients \\label{BICModles}",
        format = "latex", booktabs = T)
```

```

`r kableCoefficients`

# 8. Assumption Checking

The assumptions of the final linear model are:

* **Linearity**. There exists a linear relationship between the target and predictor variables in the model.
* **Homoscedasticity**. The variance of the error terms is constant.
* **Normality**. The error terms are normally distributed with a mean of zero.
* **Independence**. Error terms are independent of one another.

These assumptions are checked using the plots below.

```{r Assumption Checking Plot 1, out.width = '0.49\\linewidth', figwidth = 3, figheight = 3, fig.show='hold', fig.align='center', results = 'show', fig.cap = "Plots of the residual versus the fitted values for the two proposed models. \\label{ResFit}"}
# Generate plot for assumption checking linearity, homoscedasticity and normality
plot(lm_2_step_BIC, which=1)
plot(lm_2_step_BIC, which=2)
```

Linearity
From the above plot on the left of the residuals vs fitted values, we see no significant patterns or deviations from the middle, horizontal line, when the residuals have value of zero. Thus the residuals satisfy the linearity assumption.

Homoscedasticity
The residuals are distributed fairly evenly above and below the zero line, and they have no distinct kinks or turns, including at the low and high ends of the fitted values. However, we see the residuals forming a straight, diagonal line at the fitted values of around 15 to 30, and the residual values of around -20 to -40. This may be due to the fact that the dependent variable, popularity, is from 0 to 100, and hence for a fitted value of, say, 20, it is impossible to have a residual smaller than -20. Aside from this quite minor problem, the residuals show a good degree of homoscedasticity.

Normality
The above Q-Q plot shows a good degree of normality, aside from the tails of the plot, which see small kinks at either end, suggesting slightly heavier tails than a normal distribution. However, as these kinks are only small, we can still accept from this plot that the normality assumption is met.

```{r Assumption Checking Plot 2, out.width = '0.7\\linewidth', figwidth = 3, figheight = 3, fig.show='hold', fig.align='center', results = 'show', fig.cap = "Scatter plot of model residuals vs popularity, by artist. \\label{ResScatter}"}

#Define residuals, then scatter plot
popularity.res = resid(lm_2_step_BIC)
ggplot(cleanSpotifyData,aes(x = popularity.res,
                        y = popularity, col=artist))+geom_point()
```

Independence
The independence assumption is checked by assessing the residuals, and seeing if any patterns with variables arise.

From the below plot, we see a clear pattern forming in the residuals. There seems to be a distinct hierarchy among the residuals, in that the residuals have formed "layers", with The Beatles and Elvis having the highest popularity for a given value of one of our residuals, followed by Beyonce, David Bowie, Blur and Elvis, in that order. In general, these distinct layers seem to not be violated, which suggest that the residuals are not truly random. This makes sense, since if an artist has a popular song, this may affect the popularity of other songs on that album, meaning the samples aren't truly independent, and hence the independence assumption is not satisfied.

9. Prediction

```{r Prediction}
meanPredictorValues <- cleanSpotifyData %>%
  select(-c("key", "decade", "duration_ms", "artist", "track_name", "mode", "time_signature")) %>%
  lapply(mean)

modePredictorValues <- cleanSpotifyData %>%
  select(c("mode", "time_signature")) %>%
  lapply(mode)

sampleSong = data.frame(key = "C", decade = "90s", duration_ms = 180000, meanPredictorValues, modePredictorValues)

# popularity ~ decade + danceability + loudness + energy + duration_ms +
#    energy:duration_ms + decade:loudness
predict(lm_2_step_BIC, sampleSong, interval="confidence")
```

Using the final model, the predicted popularity for a three minute song form the 90s in the key of C, with all other
```

variables set to their sample mean, is `r round(predict(lm_2_step_BIC, sampleSong), 2)`. The confidence interval for this samples popularity is (`r round(predict(lm_2_step_BIC, sampleSong, interval="confidence")[2], 2)`, `r round(predict(lm_2_step_BIC, sampleSong, interval="confidence")[3], 2)`)

10. Conclusion

Nine potential predictor variables were considered to enable the construction of a linear regression model that would enable the prediction of a song's popularity. The data was loaded into the R Studio environment and examined for inconsistency. The dataset was adjusted to ensure a consistent and uniform database. Bi-variate and uni-variate analyses offered valuable insight into the type of data in the data set and relationships within the data set. Only two variables, Loudness and Energy, appeared strongly correlated.

Analysis of Variance was used to evaluate the significance of the relationship between variables and their second order interactions. Three different scopes were used when building models; one without any interaction terms, one with only the significant terms and one with all terms.Three different algorithms from model fitting were used; backward, forward and step-wise were applied in each of the previously outlined scopes. This was further expanded by using both the Akaike and Bayesian Information Criterion as heuristics. This resulted in 18 potential methods of obtaining fitted linear model. From this pool of models the general formula:

`r format(formula(lm_2_step_BIC))`

was selected based on cross validated MSE score and model complexity. The assumptions for this model of linearity, homoscedasticity, normality and independence were checked with diagnostic plots. Although some of the assumption are support by these plots some underlying sampling bias prevents this model complying with the assumption of independence. In this way the model is likely to generalise poorly to other Artists than those not considered in this study.
The model was then used to predict the popularity of a three minute song from the 90s in the Key of C, all other predictors are set to their respective mean or mode. Predicting a value of 23.80 with a 95% confidence interval of (21.25, 26.35). This model can be used to predict the popularity of any song given data 'decade', 'danceability', 'loudness', 'energy' and 'duration'. Cross validating on the given data set, the final model scored 166.8 MSE (mean squared error), or 12.9 RMSE (root mean squared error), from the true popularity.

To expand upon the work done thus far random sampling of more songs would enable stronger assumptions about independence of residuals and promote diversity in training data and thus generalisation of the model. In addition to this more complex forms of modeling and analysis could be used to contrast the relatively simple model proposed. Although in general models that involve boosting or bagging will decrease the interpret-ability of the model.

---
nocite: |
  @saleDB, @spotify
...

References

```{r}
# Scripts imported

# Define some grpahing functions

# Univariate
continuousPlot <- function(data, variable, variableLabel)
{
  ggplot(data, aes(variable)) +
    geom_histogram(fill = "#599ad3") + labs(x = variableLabel, y = "Frequency") +
    theme_minimal()
}

discretePlot <- function(data, variable, variableLabel)
{
  ggplot(data, aes(variable)) +
    geom_bar(fill = "#599ad3") + labs(x = variableLabel, y = "Frequency") +
    theme_minimal()
}

# Bivariate
boxPlot <- function(data, variableX, variableY, variableXLabel, variableYLabel)
{
    ggplot(data, aes(x = fct_reorder(variableX, variableY), y = variableY, fill = variableX)) +
      scale_y_continuous(limits = range(variableY)+c(-1,1)) +
      geom_jitter(stat = "identity", shape = 21, colour ="#000000", fill = "#32CD32") +
      stat_boxplot(geom = "errorbar") +
      geom_boxplot(alpha = 0.5, outlier.size = -10) +
      theme_minimal() +
      theme(axis.text.x = element_text(angle = 270, hjust = 0)) +
      labs(x = "", y = variableYLabel) +
      theme(legend.position = "none")
}
```

```
scatterPlot <- function(data, variableX, variableY, variableXLabel, variableYLabel, unitsX = "", unitsY = "")
{
  ggplot(data, aes(x = variableX, y =  variableY)) +
    geom_point(stat = "identity", shape = 21, fill = "#599ad3") +
    theme_minimal() +
    labs(x = paste(variableXLabel, unitsX, sep =" "), y = paste(variableYLabel, unitsY, sep = " "))
}

# Johnos cross validation function
crossValidate <- function( formula, data, k )
{
  folds <- crossv_kfold( data, k )
  models <- map( folds$train, ~lm( formula, data = . ) )
  get_pred <- function( model, test_data ){ return( add_predictions( as.data.frame(test_data), model ) ) }
  results <- map2_df( models, folds$test, get_pred, .id = "Fold" )
  MSE <- results %>%
    group_by(Fold) %>%
    summarise( MSE = mean( ( popularity - pred )^2 ), n=n() )
  CV <- sum( MSE$MSE * MSE$n ) / sum( MSE$n )
  return(CV)
}

# Simple Mode function
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}


```