Spotify Popularity Prediction

```r
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(purrr)
library(leaps)
library(boot)
set.seed(743)
```

```r
# Set it as current directory
data_dir <- "C:/Users/Richard/Spotify-Popularity-Predictor"

# List all the CSV files in the directory
csv_files <- list.files(paste0(data_dir, "/data"), pattern = ".*csv", full.names = TRUE)

# Read and combine the CSV files using map function and read_csv
combined_data <- map(csv_files, read_csv, show_col_types = FALSE) %>%
  bind_rows()

saveRDS(combined_data, file = "combined_data.rds", compress = FALSE)
```

```r
combined_data <- readRDS("combined_data.rds")
head(combined_data, n = 15)
```

```
## # A tibble: 15 x 20
##    track_id        artists album_name track_name popularity duration_ms explicit
##    <chr>           <chr>   <chr>      <chr>           <dbl>       <dbl> <lgl>
##  1 5SuOikwiRyPMVo~ Gen Ho~ Comedy     Comedy             73      230666 FALSE
##  2 4qPNDBW1i3p13q~ Ben Wo~ Ghost (Ac~ Ghost - A~         55      149610 FALSE
##  3 1iJBSr7s7jYXzM~ Ingrid~ To Begin ~ To Begin ~         57      210826 FALSE
##  4 6lfxq3CG4xtTiE~ Kina G~ Crazy Ric~ Can't Hel~         71      201933 FALSE
##  5 5vjLSffimiIP26~ Chord ~ Hold On    Hold On            82      198853 FALSE
##  6 01MVOl9KtVTNfF~ Tyrone~ Days I Wi~ Days I Wi~         58      214240 FALSE
##  7 6Vc5wAMmXdKIAM~ A Grea~ Is There ~ Say Somet~         74      229400 FALSE
##  8 1EzrEOXmMH3G43~ Jason ~ We Sing. ~ I'm Yours          80      242946 FALSE
##  9 0IktbUcnAGrvD0~ Jason ~ We Sing. ~ Lucky              74      189613 FALSE
## 10 7k9GuJYLp2Azqo~ Ross C~ Hunger     Hunger             56      205594 FALSE
## 11 4mzP5mHkRvGxdh~ Zack T~ Episode    Give Me Y~         74      244800 FALSE
## 12 5ivF4eQBqJiVL5~ Jason ~ Love Is a~ I Won't G~         69      240165 FALSE
## 13 4ptDJbJl35d7gQ~ Dan Be~ Solo       Solo               52      198712 FALSE
## 14 0X9MxHR1rTkEHD~ Anna H~ Bad Liar   Bad Liar           62      248448 FALSE
## 15 4LbWtBkN82ZRhz~ Chord ~ Hold On (~ Hold On --         56      188133 FALSE
## # i 13 more variables: danceability <dbl>, energy <dbl>, key <dbl>,
## #   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   time_signature <dbl>, track_genre <chr>
```

Quantifying Data Properties

```r
number <- dim(combined_data)
number
```

```
## [1] 114000     20
```

There are 114000 rows and 20 columns

Each row represents a track on Spotify, with details about the track such as `artist`, `album`, `popularity` and other musical attributes like `tempo` and `key`.

```r
unique_genres <- unique(combined_data$track_genre)
number_of_genres <- length(unique_genres)
number_of_genres
```

```
## [1] 114
```

There are 114 genres in this data set.

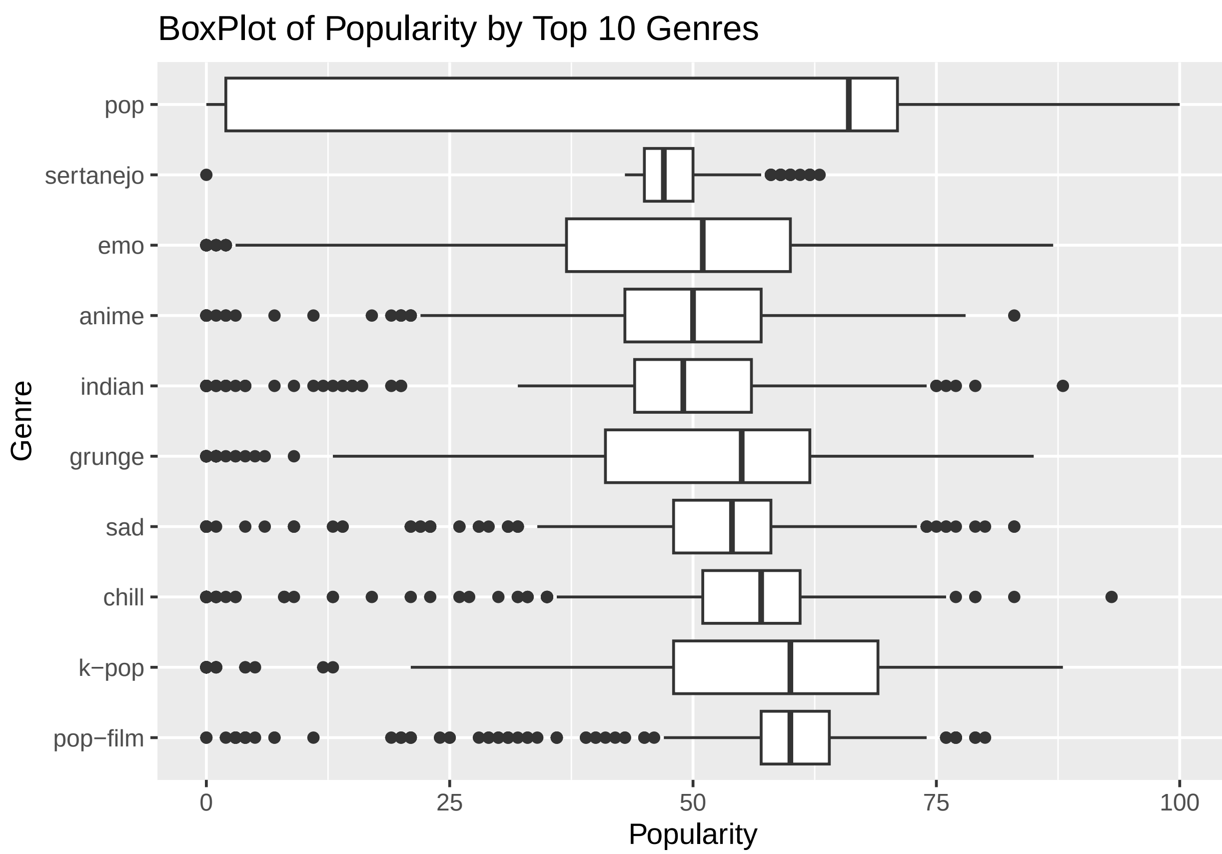
First we need to remove the songs with zero popularity.

```r
filtered_data <- combined_data %>% filter(popularity > 0)
```

```r
# Find the top N genres by mean popularity
top_10_genres <- combined_data %>%
  group_by(track_genre) %>%
  summarise(mean_popularity = mean(popularity)) %>%
  top_n(10, mean_popularity)

# Filter the data to include only the top 10 genres
top_genres_data <- combined_data %>%
  filter(track_genre %in% top_10_genres$track_genre)

# Create the boxplot
ggplot(top_genres_data, aes(x = popularity, y = reorder(track_genre, -popularity))) +
  geom_boxplot() +
  xlab("Popularity") +
  ylab("Genre") +
  ggtitle("BoxPlot of Popularity by Top 10 Genres")
```




```r
num_rows_removed <- nrow(combined_data) - nrow(filtered_data)
num_rows_removed
```

```
## [1] 16020
```

16020 Rows of data removed when filtering songs with 0 popularity.

We filtered out the data with 0 popularity as it could represent missing or skewed data. This could affect our prediction model.

```r
colSums(is.na(filtered_data))
```

```
##         track_id          artists       album_name       track_name 
##                0                0                0                0 
##       popularity      duration_ms         explicit     danceability 
##                0                0                0                0 
##           energy              key         loudness             mode 
##                0                0                0                0 
##      speechiness     acousticness instrumentalness         liveness 
##                0                0                0                0 
##          valence            tempo   time_signature      track_genre 
##                0                0                0                0
```

There are no NA values in the `filtered_data`.

Some inappropriate variables for modelling are `track_id` because this is just a unique identifier and does not help with predictions. `album_name` and `track_name` doesn't really help with predictions either, however, `artists` may help because a song from Taylor Swift for example will be much more popular than a random indie pop band.

`track_genre` is a categorical variable and needs to be encoded for modelling. We can use one-hot encoding to transform these into numerical variables if we want to use it for our model.

Model selection choices: Target Variable - `popularity` will be the target variable.

```
allyhat <- function(xtrain, ytrain, xtest, lambdas, nvmax = 50) {
  # Number of observations in the training data
  n <- nrow(xtrain)

  # Initialise a matrix to store the predicted responses
  yhat <- matrix(nrow = nrow(xtest), ncol = length(lambdas))

  # Perform backward subset selection on the training data
  search <- regsubsets(xtrain, ytrain, nvmax = nvmax, method = "back")

  # Get a summary of the subset selection result
  summ <- summary(search)

  # Loop over each value of lambda
  for (i in 1:length(lambdas)) {
    # Calculate the penalized MSE for models with different numbers of predictors
    penMSE <- n * log(summ$rss) + lambdas[i] * (1:nvmax)

    # Find the model with the smallest penalized MSE
    best <- which.min(penMSE)

    # Get the coefficients of the best model
    betahat <- coef(search, best)

    # Get the predictors in the best model
    xinmodel <- cbind(1, xtest)[, summ$which[best, ]]

    # Calculate the predicted responses for the test data
    yhat[, i] <- xinmodel %*% betahat
  }

  # Return the matrix of predicted responses
  yhat
}
```

```
# Create a new column with the selected variables
spotify_data <- filtered_data[, c(
  "danceability",
  "energy",
  "loudness",
  "speechiness",
  "acousticness",
  "instrumentalness",
  "liveness",
  "valence",
  "tempo",
  "time_signature",
  "popularity"
)]
```

```r
X <- as.matrix(filtered_data[, -c(1:5, 20)])

# Extract the response variable
y <- filtered_data$popularity

set.seed(743)

folds <- sample(rep(1:10, length.out = nrow(X)))

# Define a set of lambda values
lambdas <- c(2, 4, 6, 8, 10, 12)

# Init a matrix to store fitted values
fitted <- matrix(nrow = nrow(X), ncol = length(lambdas))

# Perform cross-validation using the allyhat function
for (k in 1:10) {
  train <- (1:nrow(X))[folds != k]
  test <- (1:nrow(X))[folds == k]
  fitted[test, ] <- allyhat(X[train, ], y[train], X[test, ], lambdas, nvmax = 14)
}

result <- rbind(lambdas, colMeans((y - fitted) ^ 2))
print(result)
```

```
##             [,1]     [,2]     [,3]     [,4]    [,5]     [,6]
## lambdas   2.0000   4.0000   6.0000   8.0000  10.000  12.0000
##         344.1721 344.1721 344.1721 344.1831 344.216 344.2164
```

```r
opt_lambda = 6
```

seems like $\lambda = 2,4,6$ has the lowest MSPE of 344.1721

```r
search <- regsubsets(X, y, nvmax = 14, method = "backward")
summ <- summary(search)


penMSE <- nrow(X)*log(summ$rss)+ opt_lambda*(1:14)
best <- which.min(penMSE)
betahat <- coef(search, best)
betahat |>
  as_tibble(rownames = "Variable") |>
  rename(Coeff = value) |>
  mutate(Coeff = round(Coeff, 3)) |>
  knitr::kable()
```

Variable	Coeff
(Intercept)	49.516
duration_ms	0.000
explicit	4.280
danceability	7.064
energy	-7.495
loudness	0.238
mode	-0.656

Variable	Coeff
speechiness	-20.977
acousticness	-1.368
instrumentalness	-11.785
liveness	-2.113
valence	-7.480
tempo	-0.007
time_signature	0.695

Based on this, we can select the model predictors. It seems like `speechiness`, `instrumentalness`, `valence`, `energy`, `dancebility`and `,explicit` all has a relative big impact on the popularity.

Our target variable is still `popularity`.

I used a linear regression model with all two-way interactions. This allows for a quick model and capture the potential relationships between the features without increasing computational complexity too much.

I used a lambda value of 6 at the end, even though 2, 4 and 6 all shared the same MSPE.

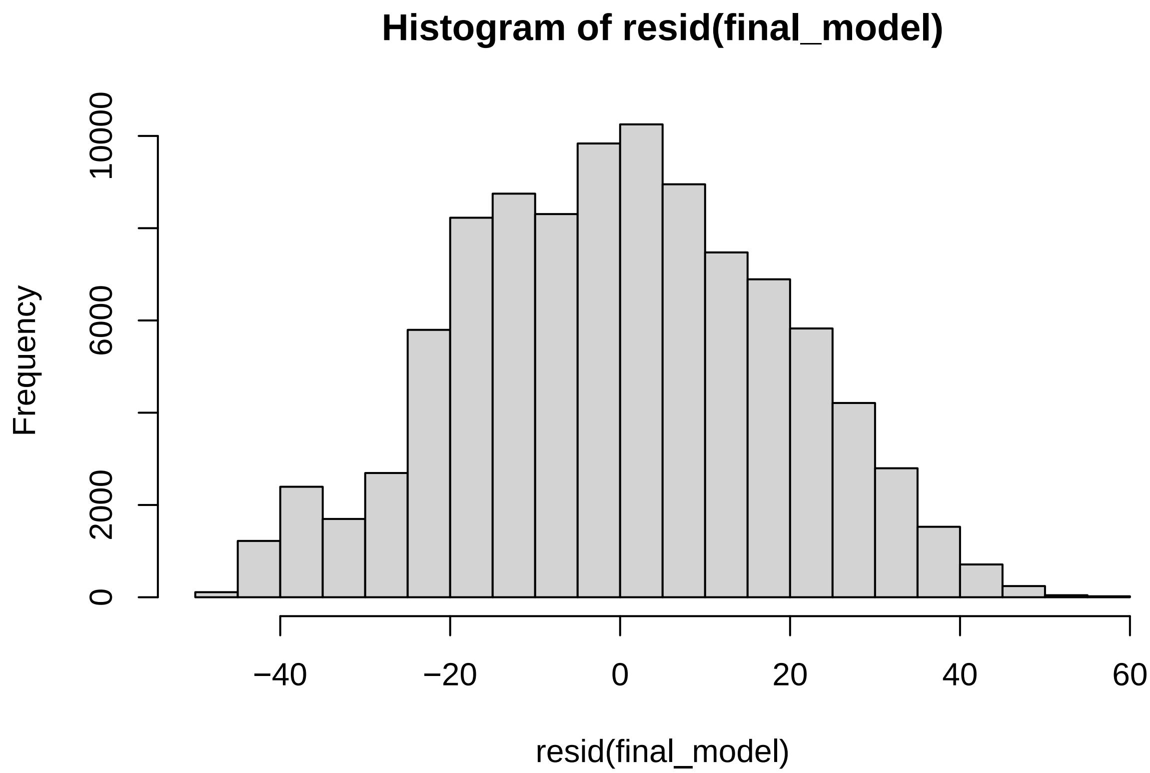
However, we haven't checked the assumptions such as linearity, normality of residuals and homoscedascity yet. ## Task 1.2.4

```r
# Fit final model using selected predictors
final_model <- lm(popularity ~ speechiness + instrumentalness + valence + energy + danceability + expli
            data = filtered_data)

# Summary of the final model
summary(final_model)
```

```
##
## Call:
## lm(formula = popularity ~ speechiness + instrumentalness + valence +
##     energy + danceability + explicit, data = filtered_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.796 -13.642  -0.023  13.410  58.598
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        42.7994     0.2604  164.35   <2e-16 ***
## speechiness       -22.9594     0.5743  -39.98   <2e-16 ***
## instrumentalness -12.9822     0.2006  -64.70   <2e-16 ***
## valence            -7.8233     0.2764  -28.30   <2e-16 ***
## energy             -3.2377     0.2485  -13.03   <2e-16 ***
## danceability        9.4622     0.3924   24.11   <2e-16 ***
## explicitTRUE        4.6638     0.2257   20.66   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.59 on 97973 degrees of freedom
## Multiple R-squared:  0.06246,    Adjusted R-squared:  0.0624
## F-statistic:  1088 on 6 and 97973 DF,  p-value: < 2.2e-16
```

```r
hist(resid(final_model))
```



The residuals look pretty normally distributed, which means our model is fine.

After fitting the final model with our selected predictors, I ran it through the summary function. All the predictors/coefficients have p-value of <0.05, which means they are all statistically significant.

However, we have a R-squared value of 0.062, which means our model only explains 6.2% of the variations in `popularity`. This means that while the model does capture some underlying pattern in the data, the low R-squared sugggests there are other factors that I haven't included in this model that contributes to the popularity.

Testing the prediction with a random song selection

```r
set.seed(743)

# Choose a song from the cleaned dataset
my_song <- filtered_data |>
  slice(614)

# Display the data for the song using kable()
knitr::kable(my_song)
```

track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature	track_genre
4Akic5cbJRHHKTdQXY7A4K	Jason Mraz	Know.	Plans	48	148000	FALSE	0.491	0.304	9	-9.379	1	0.0291	0.5	0	0.0975	0.459	149.288	4	acoustic

```r
# Prediction function for bootstrapping
pred_func <- function(data, i){
  fit <- lm(popularity ~ speechiness + instrumentalness + valence + energy + danceability + explicit,
            data = data[i,])
  predict(fit, my_song, interval = "none")
}

# Perform bootstrapping with at least 1000 iterations

boots <- boot(filtered_data, pred_func, R = 1000)


# Calculate 95% confidence interval for the prediction
ci <- boots$t0 + c(-1.96, 1.96) * sd(boots$t)
round(ci, 2)
```

```
## [1] 41.98 42.42
```

The confidence interval is 41.98 to 42.42

References

Maharshi Pandya. (2022). Spotify Tracks Dataset [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/4372070