Technical Appendix - Group 5

Contents

Section 1) Exploratory Data Analysis (EDA)

Section 1.1) EDA for the decades 2000's and 2010's

Below you can see all of the packages used, in order to carry out this report.

```
library(readxl)
library(rio)
library(magrittr)
library(lubridate)
library(devtools)
library(factoextra)
library(plyr)
library(digest)
library(tidyverse)
library(ggsci)
library(corrplot)
library(GGally)
```

Section 1.1.1) Data cleaning

This is loading in the data base and calling it Spotify. Then if you look at the AlbumReleaseDate column the dates inputted, are of different orders and in order to make them consistent we used the "parse_date_time" function. Then we extract the year and the decade from the AlbumReleaseYear column and attached these to the database as well. Now we can remove the AlbumReleaseDate column as we have the release year and decade in the database.

```
Spotify <- read_excel("edited_spotify.xlsx")
Spotify$AlbumReleaseDate =
  parse_date_time(Spotify$AlbumReleaseDate,orders=c("y","ym","ymd"))
AlbumReleaseYear <- year(Spotify$AlbumReleaseDate)
Spotify <- data.frame(Spotify, AlbumReleaseYear, check.names = TRUE)
AlbumReleaseDecade <- 10 * floor(Spotify$AlbumReleaseYear/10)
Spotify <- data.frame(Spotify, AlbumReleaseDecade, check.names = TRUE)
drop <- c("AlbumReleaseDate")
Spotify <- Spotify[ , !(names(Spotify) %in% drop)]
```

Section 1.1.2) Principal component analysis

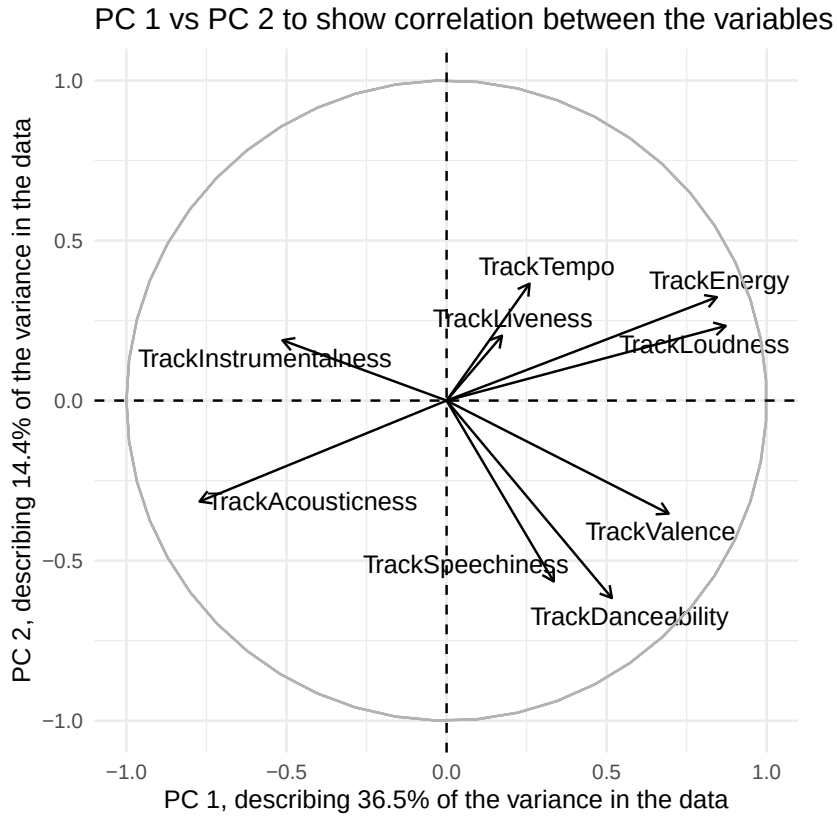
We want to carry out a principal component analysis (PCA) on the data from the 2000's and 2010's to be able to infer trends or relationships. PCA is a dimension reduction technique which allows us to look at our data on a specific 2-dimensional plane which maximises the variance of all our variables, rather than a multidimensional space (which would be impossible to visualise once we get past 3 variables/dimensions). Note that PCA only works on numerical values so we must drop all our non-numerical variables. In this specific PCA we have decided to focus on variables that describe the specific track's technical characteristics (9 in total which can be seen below) as it would be easier to interpret our PCA plot.

```
SpotifyPCA1 <- Spotify %>% filter(AlbumReleaseDecade%in%c(2010,2000))
drop <- c("Artist", "ArtistID", "ArtistNumFollowers",
          "AlbumName", "AlbumID", "AlbumBestChartPosition",
          "AlbumWeeksOnChart", "AlbumWeeksNumberOne",
          "AlbumPopularity","TrackName","TrackID",
          "AlbumReleaseYear", "TrackMode","TrackKey",
          "TrackTimeSignature", "TrackDuration",
          "ArtistPopularity", "TrackNumber",
          "AlbumReleaseDecade", "ArtistGenres")
SpotifyPCA1 <- SpotifyPCA1[ , !(names(Spotify) %in% drop)]
SpotifyPCA1 <- na.omit(SpotifyPCA1)
variable.names(SpotifyPCA1)
```

```
## [1] "TrackDanceability"     "TrackEnergy"           "TrackLoudness"
## [4] "TrackSpeechiness"      "TrackAcousticness"     "TrackInstrumentalness"
## [7] "TrackLiveness"         "TrackValence"          "TrackTempo"
```

Now to perform the PCA.
*Note: We have opted to represent our PCA here as a loading plot to see how the variables interact with each other.

```
SpotifyPCA1 <- na.omit(SpotifyPCA1)
X.pca1 <- prcomp(SpotifyPCA1, scale=TRUE, center=TRUE)
var <- get_pca_var(X.pca1)
a <- fviz_pca_var(X.pca1, col.var = "black", repel = TRUE) +
     xlab("PC 1, describing 36.5% of the variance in the data") +
     ylab("PC 2, describing 14.4% of the variance in the data")  +
     ggtitle("PC 1 vs PC 2 to show correlation between the variables") +
     theme_minimal()
a
```



Each axis of the PCA plot is a line in the multidimensional space of the original data, such that the variables vary the most along these two lines (maximising variance of the variables). We can see that we have accounted for (36.5+14.4=) 50.9% of the original variance of the data, we are still missing half of the variance so we must be cautious while interpreting our PCA plot as it does not give us the whole picture of the data.

Section 1.1.2.1) Interpreting the PCA plot

Each arrow corresponds to a variable, with the direction and size explaining relationships. The different cases are:

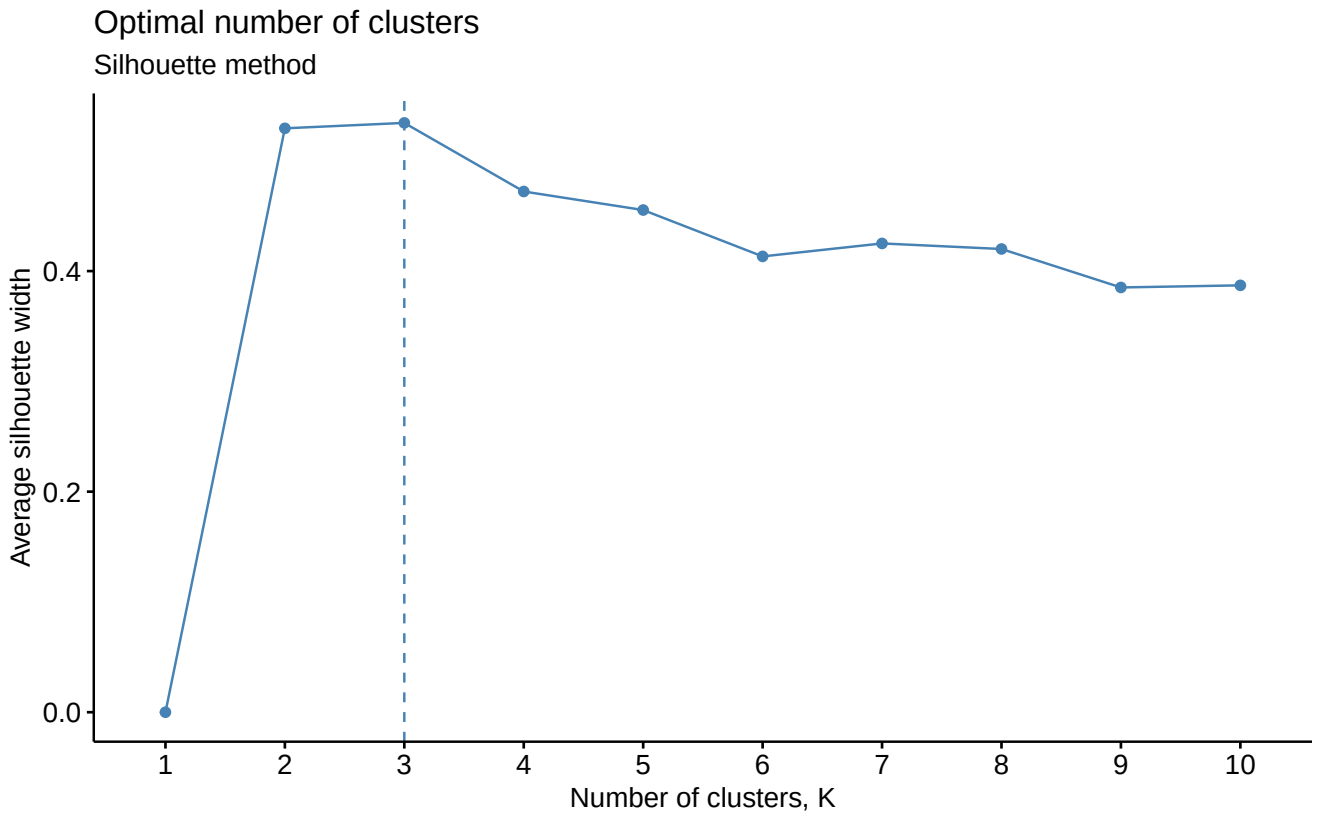
1. A longer arrows means that specific variable has a larger variance in the database
2. A shorter arrow means that the variable has a smaller variance in the database
3. Orthogonal/perpendicular arrows means that the two variables have zero or close to zero correlation
4. Arrows pointing in the same direction (small angle between them) are correlated
5. Arrows pointing in opposite directions of each other are negatively correlated (angle between them close to 180 degrees)

Remember that our PCA plot only describes 51% of the variance of our data, so our findings here cannot be taken as absolute truth.

Section 1.1.3) K-means clustering

Now we would like to cluster the data so we can see any grouping within the database and find out which variables define each one of the groups. We can do this by plotting the PC biplot with the data points and the clusters on top. We first need to decide how many clusters to use. We shall do this using the silhouette which determines how well each data points lie within its cluster, the larger the silhouette the better the clustering. This is seen below:

```
fviz_nbclust(SpotifyPCA1, kmeans, method = "silhouette") +
  labs(subtitle = "Silhouette method") +
  xlab("Number of clusters, K")
```



The above graph reveals that the largest average silhouette width is achieved at 3 clusters. Thus, the optimum number of clusters to use is 3 so from now on we will be using 3 clusters to display all of our results.

We now perform K means on our Data to get it into 3 clusters, this is useful to infer trends or spot patterns in the data. We then add another column to our database called "cluster" so that we know which cluster each track belongs to is visible in the dataset.

```
comp1 <- data.frame(X.pca1$x[,1:4])
k1 <- kmeans(comp1, 3, nstart=25, iter.max=1000)
SpotifyPCA1$Cluster <- as.factor(k1$cluster)
```

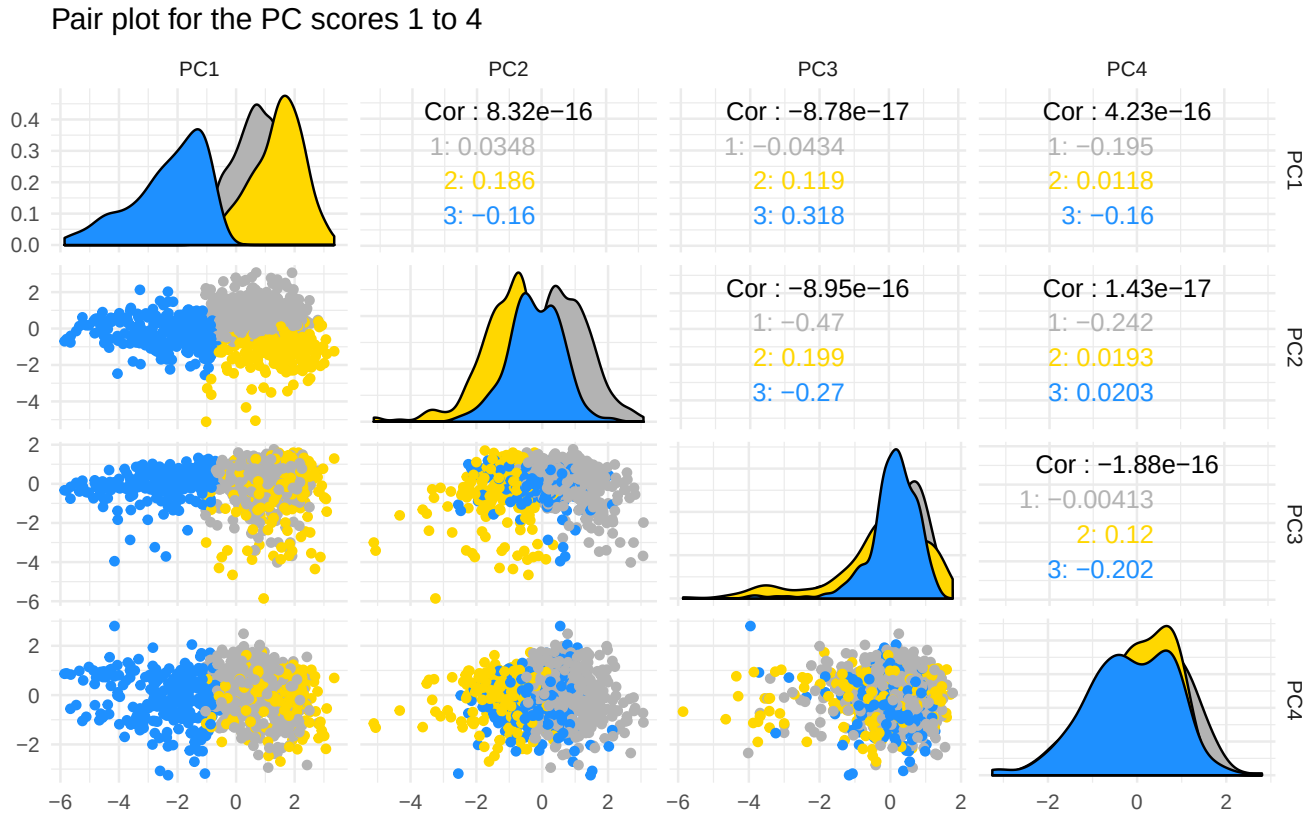
We would now like to double check our clustering. To do this we will plot the principal components (PC's) against each other and observe the clustering that 3 clusters give us. Since there are 9 PC's we cannot pick them all and thus we only pick the ones which describe most of the variance in our data. The table below is explaining the amount of variance captured by each one of the PC's:

```
library(kableExtra)
kable(get_eigenvalue(X.pca1),
      col.names = c("Eigenvalue", "% of variance explained","Cumulative variance as a %"),
      align = c('c','c','c'), digits = 3) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed")) %>%
  column_spec(1, width = "5cm", bold = TRUE, border_left = TRUE) %>%
  column_spec(4, border_right = TRUE)
```

	Eigenvalue	% of variance explained	Cumulative variance as a %
Dim.1	3.289	36.545	36.545
Dim.2	1.293	14.365	50.909
Dim.3	1.115	12.389	63.298
Dim.4	0.921	10.235	73.533
Dim.5	0.849	9.429	82.962
Dim.6	0.631	7.006	89.968
Dim.7	0.458	5.086	95.054
Dim.8	0.304	3.380	98.434
Dim.9	0.141	1.566	100.000

It can be seen from the table above in the cumulative frequency column that 4 PC's describe 73.5% of the variance in our data. This is a sufficient amount to proceed and plot the clusters down below:

```
Cluster1 <- k1[["cluster"]]
a1 <- cbind(comp1, Cluster1)
a1$Cluster1 <- factor(a1$Cluster1)
b1 <- ggpairs(a1, columns=1:4, aes(color=Cluster1)) +
  theme(legend.position = "bottom") +
  theme_minimal() +
  ggtitle("Pair plot for the PC scores 1 to 4")

for(i in 1:b1$nrow) {
  for(j in 1:b1$ncol){
    b1[i,j] <- b1[i,j] +
        scale_fill_manual(values=c("grey70", "gold1", "dodgerblue1")) +
        scale_color_manual(values=c("grey70", "gold1", "dodgerblue1"))
  }
}
b1
```

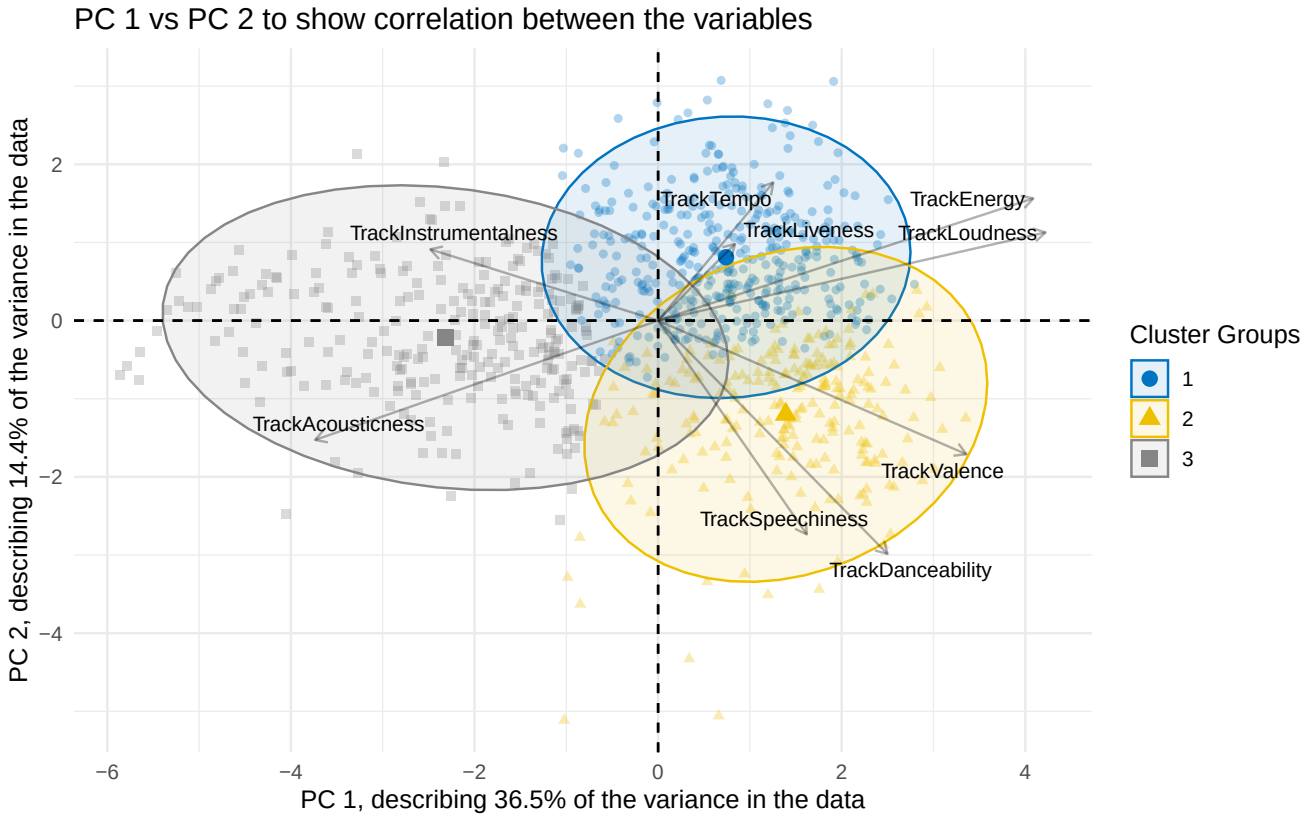


It is clear that our clustering choice provides us with distinct groups, so our choice is good.

Section 1.1.4) PCA biplot

A PCA biplot is a PC loading plot, but with the data points plotted on top of it. Since we have now clustered our data we can apply the clusters to the biplot to visually see how our data is grouped.

```
fviz_pca_biplot(X.pca1, col.ind=as.factor(k1$cluster),
                geom.ind = "point",col.var="black", palette = "jco",
                legend.title="Cluster Groups", addEllipses=TRUE,
                labelsize=3, repel = TRUE, alpha = 0.3) +
  xlab("PC 1, describing 36.5% of the variance in the data") +
  ylab("PC 2, describing 14.4% of the variance in the data")  +
  ggtitle("PC 1 vs PC 2 to show correlation between the variables") +
  theme_minimal()
```



We can see that cluster 1 is made up from songs of high valence, speechiness and danceability. Cluster 2 is made up from songs of high instrumentalness and acousticness and cluster 3 is made up from songs of high tempo, liveness, energy and loudness. However, we only have 50.9% of the variance explained with these two PC loadings, but an idea is given on how the data is grouped through the variables.

Section 1.2) EDA for the decades 1960's, 1970's, 1980's and 1990's

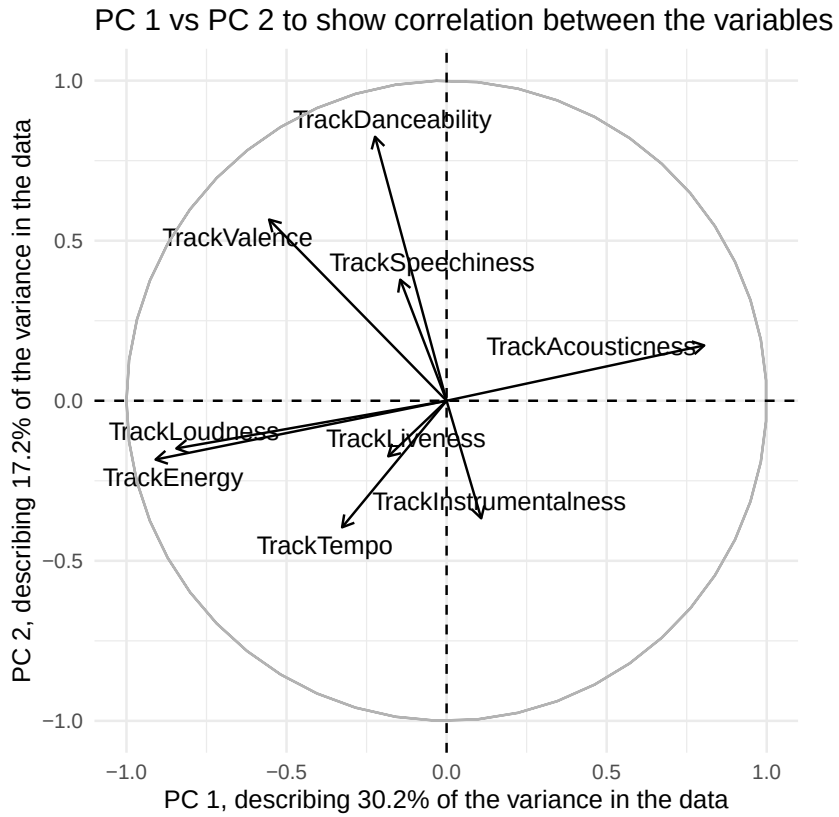
We now repeat the process of our EDA (sections 1.1.2 - 1.1.4 above) but for the albums released in the 1960's-1990's. *Note: See the EDA for 2000's & 2010's for technical details.

```
SpotifyPCA2 <- Spotify %>% filter(AlbumReleaseDecade%in%c(1960,1970,1980,1990))
drop <- c("Artist", "ArtistID", "ArtistNumFollowers",
          "AlbumName", "AlbumID", "AlbumBestChartPosition",
          "AlbumWeeksOnChart", "AlbumWeeksNumberOne",
          "AlbumPopularity", "TrackName","TrackID",
          "AlbumReleaseYear", "TrackMode", "TrackKey",
          "TrackTimeSignature", "TrackDuration",
          "ArtistPopularity", "TrackNumber",
          "AlbumReleaseDecade", "ArtistGenres")
SpotifyPCA2 <- SpotifyPCA2[ , !(names(Spotify) %in% drop)]
variable.names(SpotifyPCA2)
```

```
## [1] "TrackDanceability"     "TrackEnergy"           "TrackLoudness"
## [4] "TrackSpeechiness"      "TrackAcousticness"     "TrackInstrumentalness"
## [7] "TrackLiveness"         "TrackValence"          "TrackTempo"
```

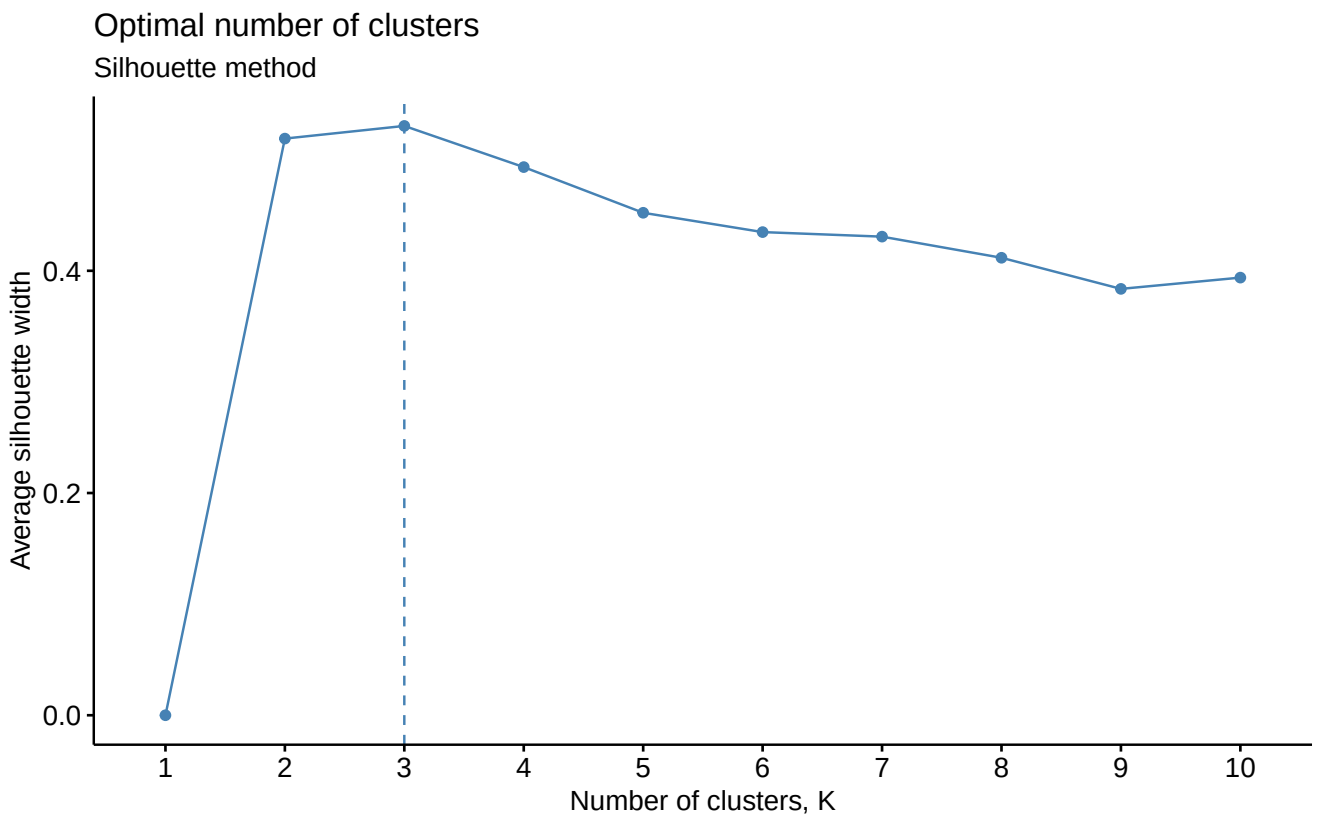
Here we have dropped the unwanted variables like in section 1.1.2 above.

```
SpotifyPCA2 <- na.omit(SpotifyPCA2)
X.pca2 <- prcomp(SpotifyPCA2, scale=TRUE, center=TRUE)
var <- get_pca_var(X.pca2)
fviz_pca_var(X.pca2, col.var = "black",  repel = TRUE) +
  xlab("PC 1, describing 30.2% of the variance in the data") +
  ylab("PC 2, describing 17.2% of the variance in the data")  +
  ggtitle("PC 1 vs PC 2 to show correlation between the variables") +
  theme_minimal()
```



Now we can see the PCA for the decades stated above. This is showing very similar trends like in the PCA plot in section 1.1.2 above. The same variables have the same correlations as before, however once again we only have (30.2+17.2=) 47.4% of the variance explained in the data. Therefore, we shouldn't receive anything concrete from this but it will give us an idea of what is happening in these decades like in section 1.1 above.

```
fviz_nbclust(SpotifyPCA2, kmeans, method = "silhouette") +
  labs(subtitle = "Silhouette method") +
  xlab("Number of clusters, K")
```



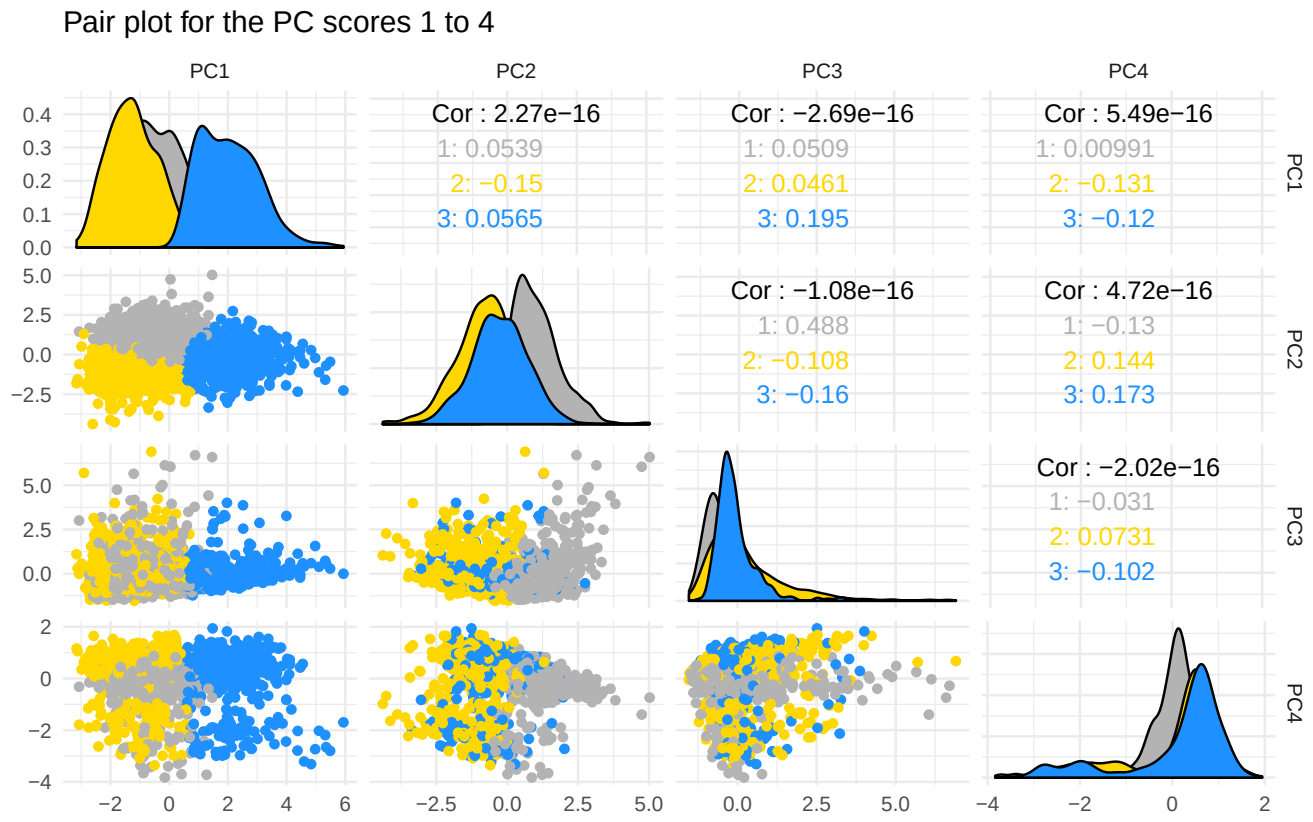
Once again, we see that we receive the largest average silhouette at 3 clusters again, meaning we will use 3 clusters for the following steps.

```
library("kableExtra")
kable(get_eigenvalue(X.pca2),
      col.names = c("Eigenvalue", "% of variance explained", "Cumulative variance as a %"),
      align = c('c','c','c'), digits = 3) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed")) %>%
  column_spec(1, width = "5cm", bold = TRUE, border_left = TRUE) %>%
  column_spec(4, border_right = TRUE)
```

	Eigenvalue	% of variance explained	Cumulative variance as a %
Dim.1	2.717	30.191	30.191
Dim.2	1.551	17.231	47.423
Dim.3	1.117	12.412	59.835
Dim.4	0.985	10.944	70.779
Dim.5	0.896	9.955	80.734
Dim.6	0.792	8.797	89.531
Dim.7	0.425	4.726	94.257
Dim.8	0.348	3.869	98.126
Dim.9	0.169	1.874	100.000

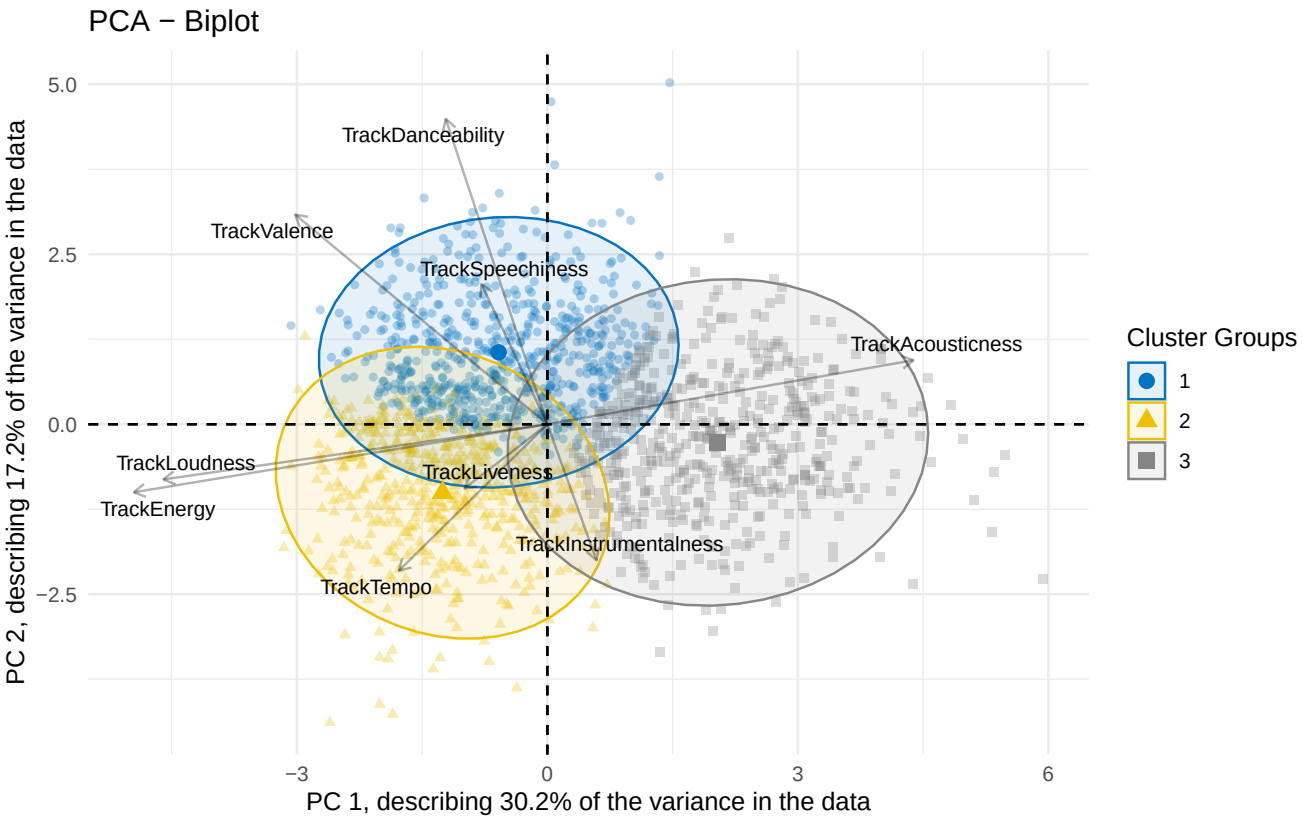
70.8% of the variance is described by the first 4 PC loadings, which is a sufficient amount and so we shall use 4 PC's again to check our clustering.

```
comp2 <- data.frame(X.pca2$x[,1:4])
k2 <- kmeans(comp2, 3, nstart=25, iter.max=1000)
Cluster2 <- k2[["cluster"]]
a2 <- cbind(comp2, Cluster2)
a2$Cluster2 <- factor(a2$Cluster2)
b2 <- ggpairs(a2, columns=1:4, aes(color=Cluster2)) +
  theme(legend.position = "bottom") +
  theme_minimal() +
  ggtitle("Pair plot for the PC scores 1 to 4")

for(i in 1:b2$nrow) {
  for(j in 1:b2$ncol){
    b2[i,j] <- b2[i,j] +
        scale_fill_manual(values=c("grey70", "gold1", "dodgerblue1")) +
        scale_color_manual(values=c("grey70", "gold1", "dodgerblue1"))
  }
}
b2
```



We can once again, see that we have distinct clusters in the graph above so the clustering is working as intended.

```
fviz_pca_biplot(X.pca2, col.ind=as.factor(k2$cluster), geom.ind = "point",
                col.var="black", palette = "jco", legend.title="Cluster Groups",
                addEllipses=TRUE, labelsize=3, repel = TRUE, alpha=0.3) +
  xlab("PC 1, describing 30.2% of the variance in the data") +
  ylab("PC 2, describing 17.2% of the variance in the data") +
  theme_minimal()
```



In general, the features that split the clusters are similar between the two EDA's implying the combinations of technical characteristics holds throughout the last 60+ years. Cluster 1 includes songs of high danceability, valance and speechiness, cluster 2 includes songs of high loudness, energy, liveness, tempo and instrumentalness and cluster 3 includes songs of high acousticness and high instrumentalness as well. Cluster 3 reveals that most of songs of that era have high acousticness was almost a whole category by themselves. However, with both the EDA's above we have roughly only 50% of the variance explained but we have an idea of the features of the data.

Section 2) Predicting Future Popular Music (Task 1)

Section 2.1) What genres are in the database?

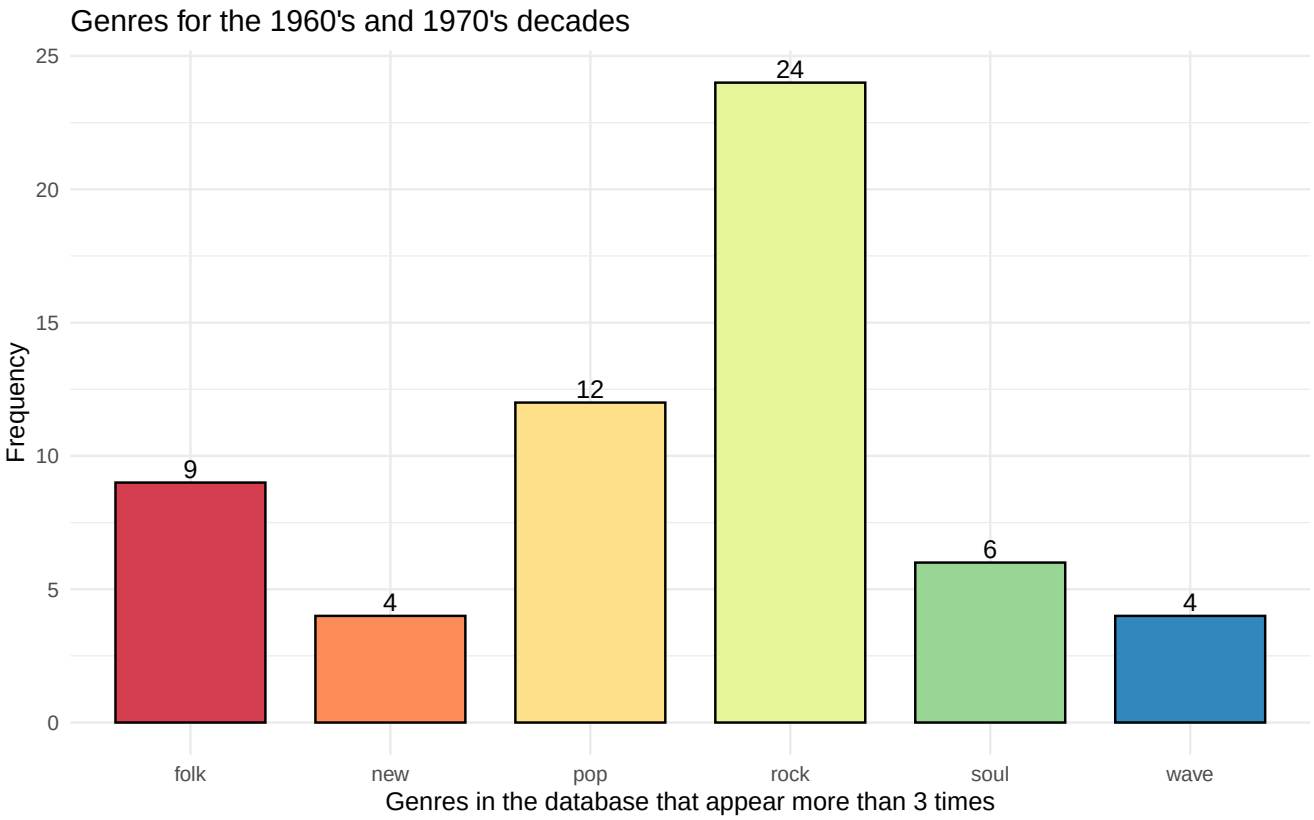
Firstly, we are going to split the database by the genres category so we can see what genres exist in each of the decades. We have 6 decades the 60's, the 70's and all the way up to the 2010's, so to make this easy we split the timeline into 3 different graphs. The data is firstly split and cleaned:

```
Spotify2 <- read_excel("edited_spotify.xlsx")
Spotify2 <- as_tibble(Spotify2)
Spotify2$AlbumReleaseDate =
  parse_date_time(Spotify2$AlbumReleaseDate,orders=c("y","ym","ymd"))
AlbumReleaseYear <- year(Spotify2$AlbumReleaseDate)
Spotify2 <- data.frame(Spotify2, AlbumReleaseYear, check.names = TRUE)
AlbumReleaseDecade <- 10 * floor(Spotify2$AlbumReleaseYear/10)
Spotify2 <- data.frame(Spotify2, AlbumReleaseDecade, check.names = TRUE)
Spotify2.genres <- Spotify2 %>%
  mutate(ArtistGenres = strsplit(ArtistGenres, ","))
```

This is loading in the database, setting it as a "tibble" and then adding on a years and decades columns into the dataset. Then we are mutating the genres column by splitting it, using the "strsplit" command. This splits it by the comma in the database. The reason why we are doing this is because in the database albums and songs have multiple genres, and they are split by a comma within the same cell. We proceed by using this command to split the genres by the comma and then regrouping the data.

Now lets see what genres appear in the 60's and 70's:

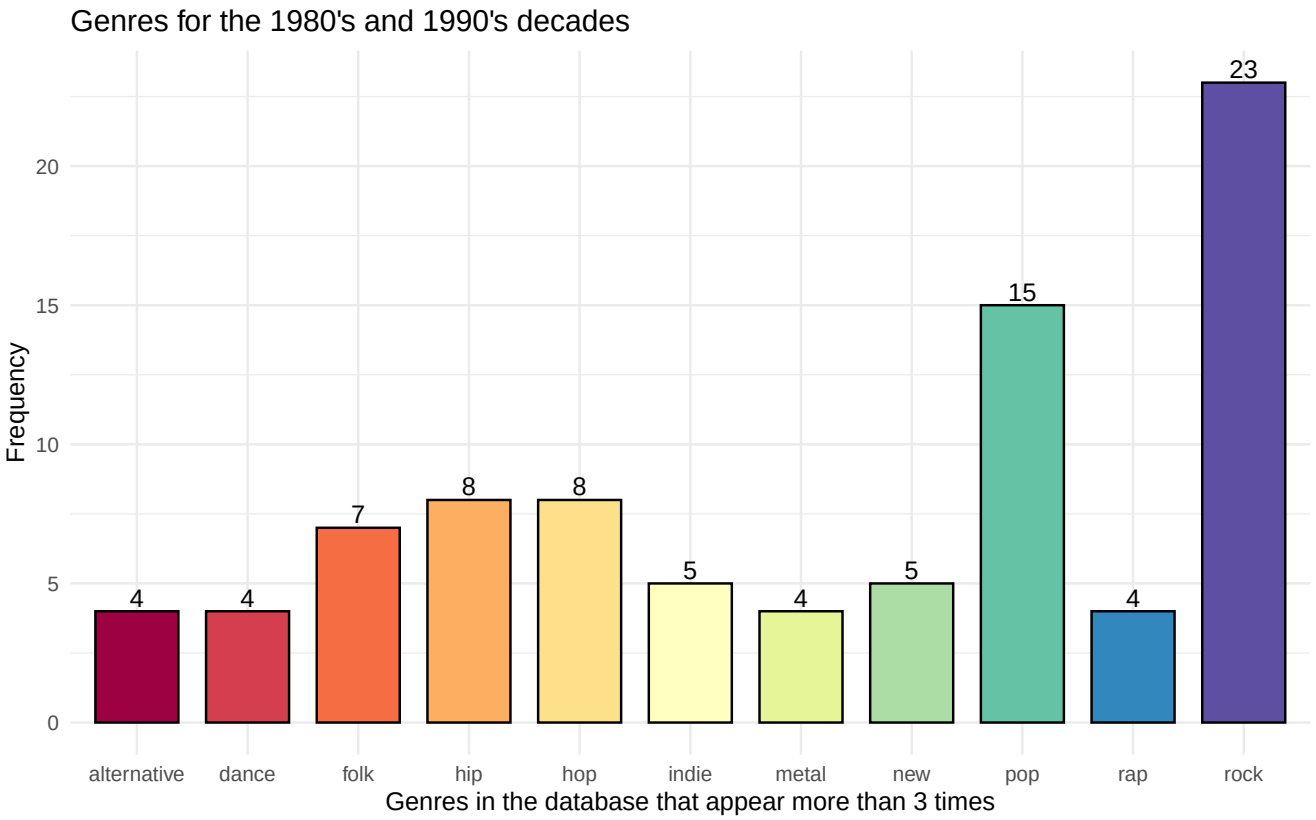
```
library(dplyr)
Spotify2.genres.1960.1970 <- Spotify2.genres %>%
  filter(AlbumReleaseDecade == "1960" | AlbumReleaseDecade == "1970")
genres <- na.omit(unique(unlist(Spotify2.genres.1960.1970$ArtistGenres)))
split_genres <- NULL
for(i in 1:length(genres)) {
  split_genres[i] <- strsplit(genres[i], " ")
}
genre_words <- unlist(split_genres)
unique_genre_words <- unique(genre_words)
genre_words_table <- subset(as.data.frame(table(genre_words)), (Freq > 3))
a <- ggplot(data=genre_words_table,
            aes(x=genre_words, y=Freq, fill=genre_words)) +
     geom_bar(stat="identity", width=0.75, colour="black") +
     theme_minimal() +
     scale_fill_brewer(palette = "Spectral") +
     ylab("Frequency") +
     xlab("Genres in the database that appear more than 3 times") +
     labs(title="Genres for the 1960's and 1970's decades") +
     theme(legend.position = "none") +
     geom_text(aes(label=Freq), position=position_dodge(width=0.9), vjust=-0.25)
print(a)
```



It is clear that the data from the 60's and 70's is mostly composed of rock, pop and folk songs.

Now lets look at the 80's and 90's data:

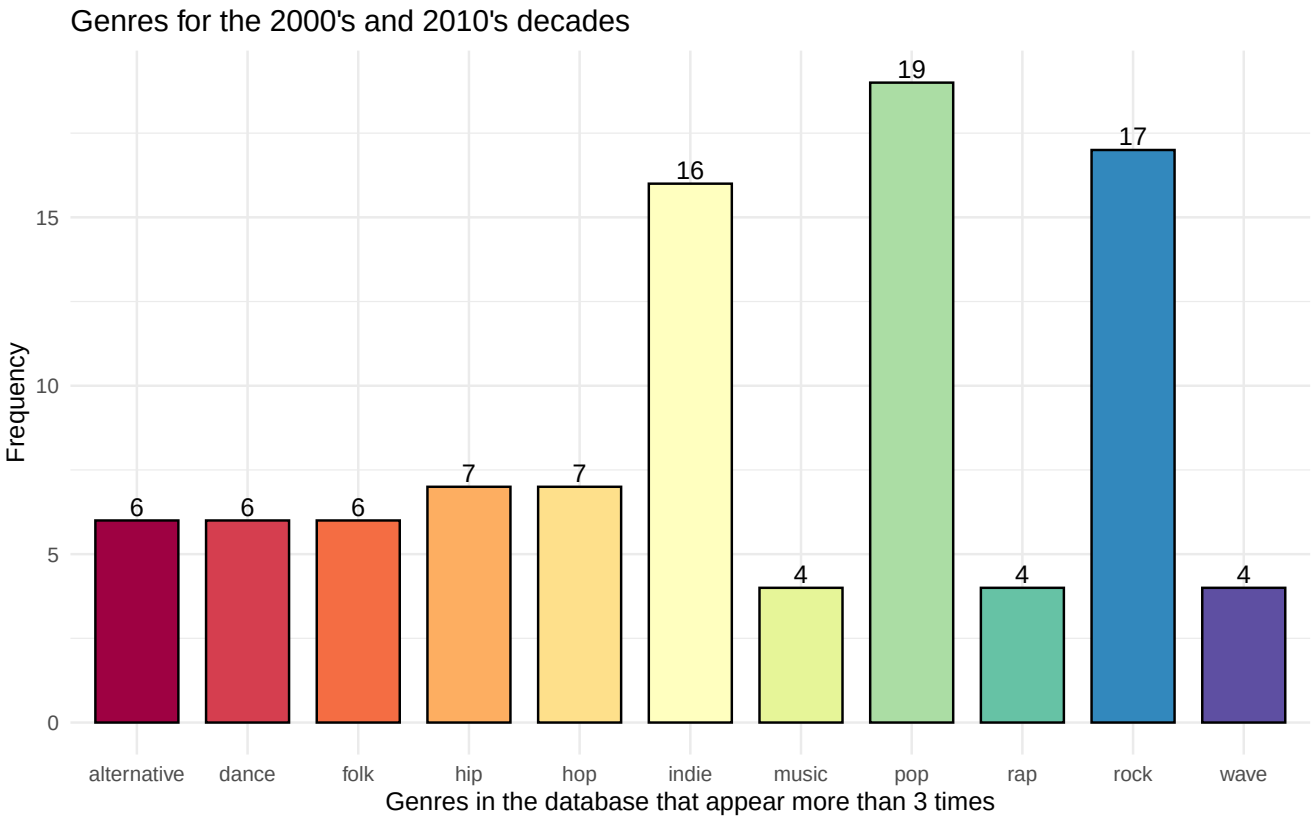
```
Spotify2.genres.1980.1990 <- Spotify2.genres %>%
  filter(AlbumReleaseDecade == "1980" | AlbumReleaseDecade == "1990")
genres <- na.omit(unique(unlist(Spotify2.genres.1980.1990$ArtistGenres)))
split_genres <- NULL
for(i in 1:length(genres)) {
  split_genres[i] <- strsplit(genres[i], " ")
}
genre_words <- unlist(split_genres)
unique_genre_words <- unique(genre_words)
genre_words_table2 <- subset(as.data.frame(table(genre_words)), (Freq > 3))
a <- ggplot(data=genre_words_table2,
            aes(x=genre_words, y=Freq, fill=genre_words)) +
     geom_bar(stat="identity", width=0.75, colour="black") +
     theme_minimal() +
     scale_fill_brewer(palette = "Spectral") +
     ylab("Frequency") +
     xlab("Genres in the database that appear more than 3 times") +
     labs(title="Genres for the 1980's and 1990's decades") +
     theme(legend.position = "none") +
     geom_text(aes(label=Freq), position=position_dodge(width=0.9), vjust=-0.25)
print(a)
```



Now we have the vast majority of the data from the 80's and 90's falls into the rock and pop genre, which is very different from the 60's and 70's.

Now let's see what genres make up the data from the 2000's and 2010's:

```
Spotify2.genres.2000.2010 <- Spotify2.genres %>%
  filter(AlbumReleaseDecade == "2000" | AlbumReleaseDecade == "2010")
genres <- na.omit(unique(unlist(Spotify2.genres.2000.2010$ArtistGenres)))
split_genres <- NULL
for(i in 1:length(genres)) {
  split_genres[i] <- strsplit(genres[i], " ")
}
genre_words <- unlist(split_genres)
unique_genre_words <- unique(genre_words)
genre_words_table3 <- subset(as.data.frame(table(genre_words)), (Freq > 3))
a <- ggplot(data=genre_words_table3,
            aes(x=genre_words, y=Freq, fill=genre_words)) +
     geom_bar(stat="identity", width=0.75, colour="black") +
     theme_minimal() +
     scale_fill_brewer(palette = "Spectral") +
     ylab("Frequency") +
     xlab("Genres in the database that appear more than 3 times") +
     labs(title="Genres for the 2000's and 2010's decades") +
     theme(legend.position = "none") +
     geom_text(aes(label=Freq), position=position_dodge(width=0.9), vjust=-0.25)
print(a)
```



Now we can see once again the genres for these two decades massively differ again. This leads to a conclusion that the genres in the database used, change a lot throughout the decades. We also conclude that in order to see what is going to be popular we should look at the most recent decades and to build models for the data in this database i.e. from just the 2000's and 2010's.

Before we create the models, the database must be altered:

```
SpotifyData = {Spotify2 %>%
    group_by(Artist,ArtistID,ArtistGenres,ArtistPopularity,ArtistNumFollowers,
             AlbumName,AlbumID,AlbumBestChartPosition,AlbumWeeksOnChart,
             AlbumWeeksNumberOne,AlbumPopularity,
             AlbumReleaseYear, AlbumReleaseDecade) %>%
    summarize(AlbumAcousticness = mean(TrackAcousticness),
              AlbumValence = mean(TrackValence),
              AvgTrackDuration = mean(TrackDuration),
              AlbumDanceability = mean(TrackDanceability),
              AlbumEnergy = mean(TrackEnergy),
              AlbumLoudness = mean(TrackLoudness),
              AvgTrackMode = mean(TrackMode),
              AlbumSpeechiness = mean(TrackSpeechiness),
              AlbumInstrumentalness = mean(TrackInstrumentalness),
              AlbumLiveness = mean(TrackLiveness),
              AvgTrackTempo = mean(TrackTempo),
              AvgTrackTimeSignature = mean(TrackTimeSignature))}

SpotifyData1 <- filter(SpotifyData, AlbumReleaseDecade ==
                         "2000" | AlbumReleaseDecade == "2010")

## Excluding outliers was considered but it excluded too much data and no accurate
## model could be fitted.
#outliers <- sort(boxplot.stats(SpotifyData1$ArtistNumFollowers)$out, decreasing = TRUE)[1:5]
#SpotifyData1 <- SpotifyData1[!SpotifyData1$ArtistNumFollowers %in% outliers,]
#outliers <- boxplot.stats(SpotifyData1$ArtistPopularity)$out
#SpotifyData1 <- SpotifyData1[!SpotifyData1$ArtistPopularity %in% outliers,]
```

This groups the data by those variables, meaning that we remove the track features and turn them into album averaged features. This must be done due to different albums having different numbers of songs inside them, this may infact which may affect our models and plots. It is also a good way to make the database smaller and easy to deal with. Then we are making sure that we only have data from the 2000's and 2010's specified above and storing this altered database as "SpotifyData1".

Section 2.2) Understanding relationships between popularity and track characteristics

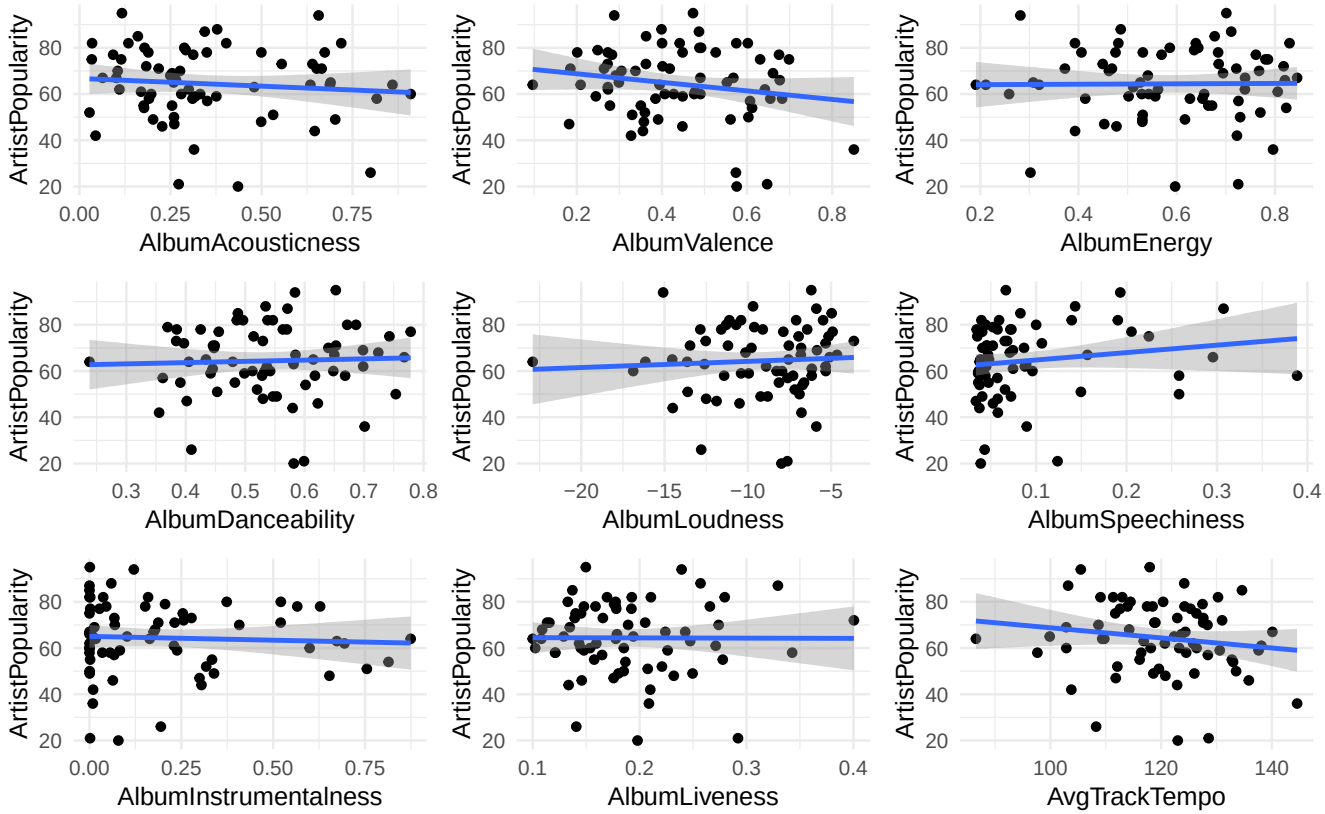
To start building our models we need to understand the relationship between each of the variables describing the track and the variables we define as popular.

Section 2.2.1) Graph plotting

To start with building the models, we have plotted the relationships between each of the descriptive variables and the variables we defined as popularity. We have done this by plotting scatter plots for each of the variables against each other. Below you can see the relationships between the nine descriptive variables against ArtistPopularity are revealed:

```
library(gridExtra)
plot.Acousticness <- ggplot(data=SpotifyData1, aes(x=AlbumAcousticness,y=ArtistPopularity)) +
    geom_point() +
    geom_smooth(method = "lm") +
  theme_minimal()
ArtPop <- SpotifyData1$ArtistPopularity
Acousticness <- SpotifyData1$AlbumAcousticness

plot.Valence <- ggplot(data=SpotifyData1, aes(x=AlbumValence,y=ArtistPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Valence <- SpotifyData1$AlbumValence
ArtPop <- SpotifyData1$ArtistPopularity

plot.Energy <- ggplot(data=SpotifyData1, aes(x=AlbumEnergy,y=ArtistPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Energy <- SpotifyData1$AlbumEnergy
ArtPop <- SpotifyData1$ArtistPopularity

plot.Danceability <- ggplot(data=SpotifyData1, aes(x=AlbumDanceability,y=ArtistPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Danceability <- SpotifyData1$AlbumDanceability
ArtPop <- SpotifyData1$ArtistPopularity

plot.Loudness <- ggplot(data=SpotifyData1, aes(x=AlbumLoudness,y=ArtistPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Loudness <- SpotifyData1$AlbumLoudness
ArtPop <- SpotifyData1$ArtistPopularity

plot.Speechiness <- ggplot(data=SpotifyData1, aes(x=AlbumSpeechiness,y=ArtistPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Speechiness <- SpotifyData1$AlbumSpeechiness
ArtPop <- SpotifyData1$ArtistPopularity

plot.Instrumentalness <- ggplot(data=SpotifyData1, aes(x=AlbumInstrumentalness,
```

```
                                                    y=ArtistPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Instrumentalness <- SpotifyData1$AlbumInstrumentalness
ArtPop <- SpotifyData1$ArtistPopularity

plot.Liveness <- ggplot(data=SpotifyData1, aes(x=AlbumLiveness,y=ArtistPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Liveness <- SpotifyData1$AlbumLiveness
ArtPop <- SpotifyData1$ArtistPopularity

plot.Tempo <- ggplot(data=SpotifyData1, aes(x=AvgTrackTempo,y=ArtistPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Tempo <- SpotifyData1$AvgTrackTempo
ArtPop <- SpotifyData1$ArtistPopularity

grid.arrange(plot.Acousticness, plot.Valence, plot.Energy,
             plot.Danceability, plot.Loudness, plot.Speechiness,
             plot.Instrumentalness, plot.Liveness,
             plot.Tempo,ncol=3, nrow =3)
```

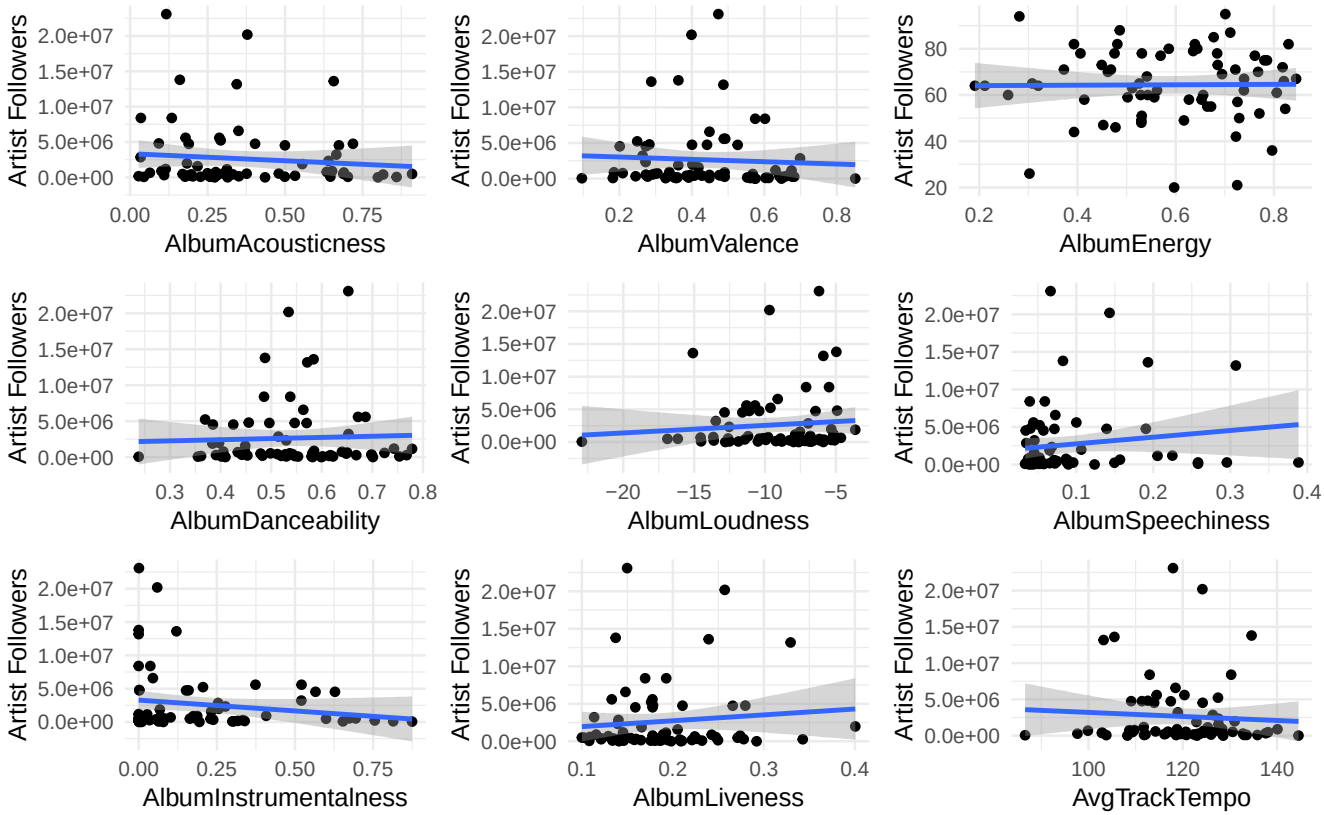


Now we have done the same for each of the 9 variables against ArtistNumFollowers:

```
plot.Acousticness <- ggplot(data=SpotifyData1, aes(x=AlbumAcousticness,y=ArtistNumFollowers)) +
    geom_point() +
```

```
    geom_smooth(method = "lm") +
  theme_minimal() +
  ylab("Artist Followers")
ArtFollowers <- SpotifyData1$ArtistNumFollowers
Acousticness <- SpotifyData1$AlbumAcousticness

plot.Valence <- ggplot(data=SpotifyData1, aes(x=AlbumValence,y=ArtistNumFollowers)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal() +
  ylab("Artist Followers")
Valence <- SpotifyData1$AlbumValence

plot.Energy <- ggplot(data=SpotifyData1, aes(x=AlbumEnergy,y=ArtistPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal() +
  ylab("Artist Followers")
Energy <- SpotifyData1$AlbumEnergy

plot.Danceability <- ggplot(data=SpotifyData1, aes(x=AlbumDanceability,y=ArtistNumFollowers)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal() +
  ylab("Artist Followers")
Danceability <- SpotifyData1$AlbumDanceability

plot.Loudness <- ggplot(data=SpotifyData1,
                        aes(x=AlbumLoudness,y=ArtistNumFollowers)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal() +
  ylab("Artist Followers")
Loudness <- SpotifyData1$AlbumLoudness

plot.Speechiness <- ggplot(data=SpotifyData1, aes(x=AlbumSpeechiness,y=ArtistNumFollowers)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal() +
  ylab("Artist Followers")
Speechiness <- SpotifyData1$AlbumSpeechiness

plot.Instrumentalness <- ggplot(data=SpotifyData1, aes(x=AlbumInstrumentalness,
                                                       y=ArtistNumFollowers)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal() +
  ylab("Artist Followers")
Instrumentalness <- SpotifyData1$AlbumInstrumentalness

plot.Liveness <- ggplot(data=SpotifyData1,
                        aes(x=AlbumLiveness,y=ArtistNumFollowers)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal() +
  ylab("Artist Followers")
```

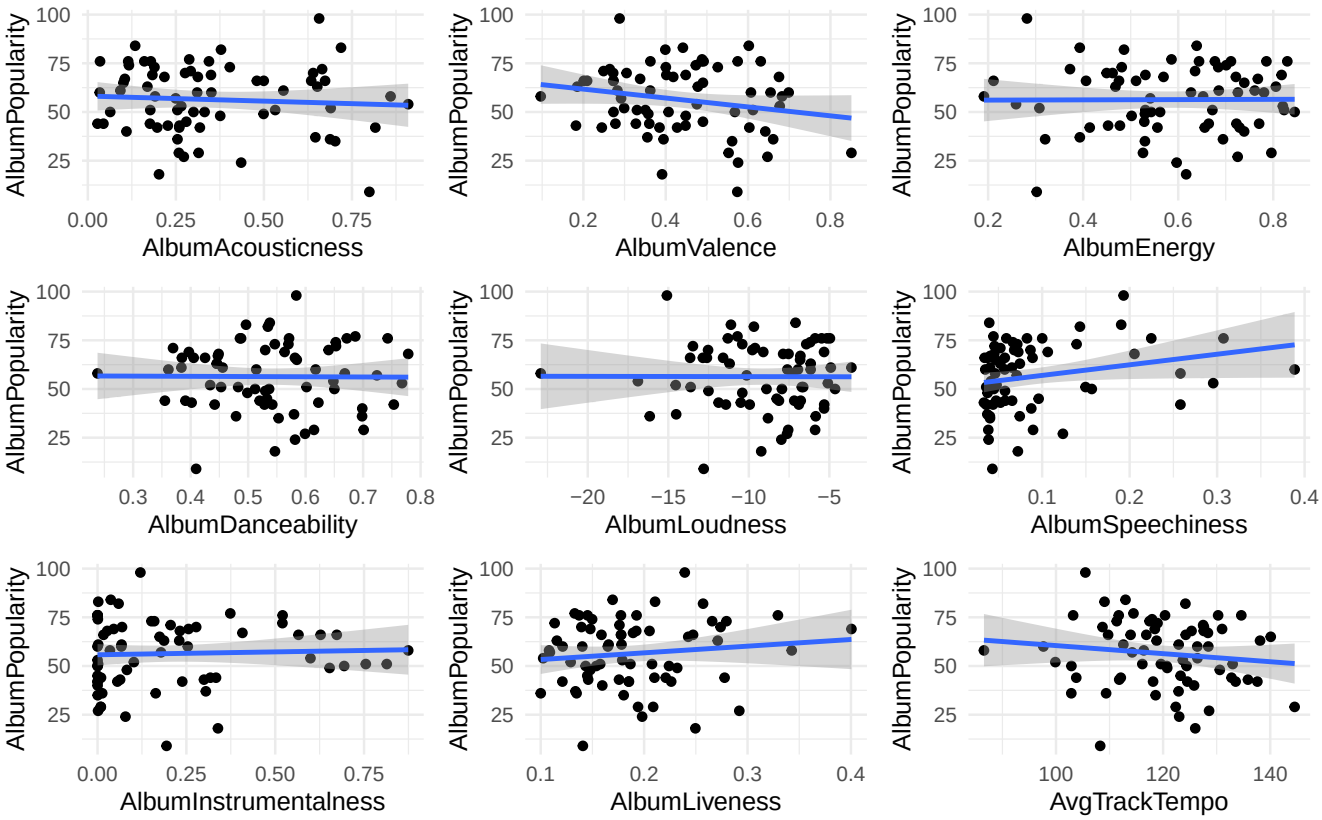
```
Liveness <- SpotifyData1$AlbumLiveness

plot.Tempo <- ggplot(data=SpotifyData1, aes(x=AvgTrackTempo,y=ArtistNumFollowers)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal() +
  ylab("Artist Followers")
Tempo <- SpotifyData1$AvgTrackTempo

grid.arrange(plot.Acousticness, plot.Valence, plot.Energy,
             plot.Danceability, plot.Loudness, plot.Speechiness,
             plot.Instrumentalness, plot.Liveness, plot.Tempo,
             ncol=3, nrow =3)
```



Now we have plotted each of the 9 variables against AlbumPopularity:

```
plot.Acousticness <- ggplot(data=SpotifyData1, aes(x=AlbumAcousticness,y=AlbumPopularity)) +
    geom_point() +
    geom_smooth(method = "lm") +
  theme_minimal()
AlbPop <- SpotifyData1$AlbumPopularity
Acousticness <- SpotifyData1$AlbumAcousticness

plot.Valence <- ggplot(data=SpotifyData1, aes(x=AlbumValence,y=AlbumPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Valence <- SpotifyData1$AlbumValence

plot.Energy <- ggplot(data=SpotifyData1, aes(x=AlbumEnergy,y=AlbumPopularity)) +
```

```
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Energy <- SpotifyData1$AlbumEnergy

plot.Danceability <- ggplot(data=SpotifyData1, aes(x=AlbumDanceability,y=AlbumPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Danceability <- SpotifyData1$AlbumDanceability

plot.Loudness <- ggplot(data=SpotifyData1, aes(x=AlbumLoudness,y=AlbumPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Loudness <- SpotifyData1$AlbumLoudness

plot.Speechiness <- ggplot(data=SpotifyData1, aes(x=AlbumSpeechiness,y=AlbumPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Speechiness <- SpotifyData1$AlbumSpeechiness

plot.Instrumentalness <- ggplot(data=SpotifyData1, aes(x=AlbumInstrumentalness,
                                                       y=AlbumPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Instrumentalness <- SpotifyData1$AlbumInstrumentalness

plot.Liveness <- ggplot(data=SpotifyData1, aes(x=AlbumLiveness,y=AlbumPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Liveness <- SpotifyData1$AlbumLiveness

plot.Tempo <- ggplot(data=SpotifyData1, aes(x=AvgTrackTempo,y=AlbumPopularity)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_minimal()
Tempo <- SpotifyData1$AvgTrackTempo

grid.arrange(plot.Acousticness, plot.Valence, plot.Energy,
             plot.Danceability, plot.Loudness, plot.Speechiness,
             plot.Instrumentalness, plot.Liveness, plot.Tempo,
             ncol=3, nrow =3)
```


AlbumPopularity
AlbumAcousticness
AlbumValence
AlbumEnergy
AlbumDanceability
AlbumLoudness
AlbumSpeechiness
AlbumInstrumentalness
AlbumLiveness
AvgTrackTempo

Section 2.2.2) Correlation coeffecients

```
# If you go through and write cor() between all of the variables
# with ArtistPopularity you get all of these correlation coeffecients
ArtistPop <- c(-0.09796291, -0.1847146, 0.009400858, 0.0397447,
  0.06076765, 0.1549412, -0.05011585, -0.004105714,
  -0.1534358)
# If you go through and write cor() between all of the variables
# with ArtistNumFollowers you get all of these correlation coeffecients
ArtistNumFoll <-c(-0.09900453, -0.05534053, 0.01922734, 0.03963681,
                        0.0889159, 0.1446111, -0.1660594, 0.1025913,
                       -0.06602305)
# If you go through and write cor() between all of the variables
# with AlbumPopularity you get all of these correlation coeffecients
AlbumPop<- c(-0.06870483, -0.2055328, 0.006223914, -0.007292709,
                      -0.002637922, 0.2365357, 0.04031781, 0.1200788,
                      -0.1310008)
a <- cbind(ArtistPop, ArtistNumFoll)
a <-as.data.frame(cbind(a, AlbumPop))
rownames(a) <- c("Acousticness", "Valence", "Energy", "Danceability", "Loudness",
                 "Speechiness", "Instrumentalness", "Liveness", "Tempo")
library(kableExtra)
library(knitr)
a %>%
  kable(col.names = c("Artist Popularity", "Artist Followers", "Album Popularity"),
      align = c('c','c','c'), digits = 3) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed")) %>%
  column_spec(1, width = "5cm", bold = TRUE,
              border_left = TRUE, border_right = TRUE) %>%
  column_spec(4, border_right = TRUE)
```

	Artist Popularity	Artist Followers	Album Popularity
Acousticness	-0.098	-0.099	-0.069
Valence	-0.185	-0.055	-0.206
Energy	0.009	0.019	0.006
Danceability	0.040	0.040	-0.007
Loudness	0.061	0.089	-0.003
Speechiness	0.155	0.145	0.237
Instrumentalness	-0.050	-0.166	0.040
Liveness	-0.004	0.103	0.120
Tempo	-0.153	-0.066	-0.131

From the correlations between the variables we can see that anything which is larger than |0.05| is significant to the ArtistPopularity. This is giving us a good starting point for the predictors to use for the model for predicting the album popularity, so the variables we will start with, are the "AlbumAcousticness", "AlbumValence", "AlbumSpeechiness", "AvgTrackTempo", "AlbumInstrumentalness" and "AlbumLoudness".

For ArtistNumFollowers the variables with a correlation coefficient of around |0.05| or greater seems to be significant for a good starting point for the model building. These variables we will start with are "AlbumAcousticness", "AlbumValence", "AlbumSpeechiness", "AvgTrackTempo", "AlbumLoudness", "AlbumInstrumentalness" and "AlbumLiveness" for the model for artist number of followers.

Again the variables with a correlation of aroud |0.05| or greater seem to be significant to be a good starting point for the model building for AlbumPopularity. These variables we will start with are "AlbumAcousticness", "AlbumValence", "AlbumSpeechiness", "AvgTrackTempo" and "AlbumLivess" for the model for album popularity.

Now we have a good starting point for our models for each of these variables to move forward with model building.

Section 2.3) Model fitting

Now we have to build models for each of the 3 variables we defined as popular, these were "ArtistNumFollowers", "ArtistPopularity" and "AlbumPopularity". We are going to use generalized linear models for these with a log link function. This is due to variables do not all have a normal distributions so we cannot simply use normal linear models. Also not all of the variables have a range of 0-1 so we can't use beta regression models. This means generalized linear models are the best option for this.
The first model we are going to build, is the one for ArtistPopularity. To start with we will use the predictors we saw as significant from the EDA above. Any indicator with a correlation coefficient greater than |0.05| is initially included to ensure enough indicators are considered.
We will be using two different tests to see which variables are significant, these tests are F-tests and Chi squared tests. For the F-test we will be using a 95% significance level, so we are looking for p-values of around 0.05 or smaller, but the F-test specifically is testing against the model with just 1 as the intercept. The ChiSq test is also just testing the goodness of fit for each of the variables, we are looking for a Pr(Chi) value of around 0.05 or smaller again.

```
library(MASS)
artistNumfol <- glm(ArtistNumFollowers ~ AlbumSpeechiness + AlbumValence + AlbumLoudness +
                    AvgTrackTempo + AlbumAcousticness + AlbumInstrumentalness + AlbumLiveness,
                    data=SpotifyData1,
                    family = quasi(link = "log", variance = "mu^2"))
dropterm(artistNumfol, test=c("Chisq"))
```

```
## Single term deletions
##
## Model:
## ArtistNumFollowers ~ AlbumSpeechiness + AlbumValence + AlbumLoudness +
##     AvgTrackTempo + AlbumAcousticness + AlbumInstrumentalness +
##     AlbumLiveness
##                       Df Deviance scaled dev. Pr(Chi)
## <none>                     178.08
## AlbumSpeechiness       1   181.46      1.5310 0.21596
## AlbumValence           1   192.60      6.5887 0.01026 *
## AlbumLoudness          1   178.70      0.2813 0.59587
## AvgTrackTempo          1   178.85      0.3497 0.55431
## AlbumAcousticness      1   188.13      4.5610 0.03271 *
## AlbumInstrumentalness  1   181.10      1.3685 0.24207
## AlbumLiveness          1   178.73      0.2928 0.58846
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dropterm(artistNumfol, test=c("F"))
```

```
## Single term deletions
##
## Model:
## ArtistNumFollowers ~ AlbumSpeechiness + AlbumValence + AlbumLoudness +
##     AvgTrackTempo + AlbumAcousticness + AlbumInstrumentalness +
##     AlbumLiveness
##                       Df Deviance F value   Pr(F)
## <none>                     178.08
## AlbumSpeechiness       1   181.46  1.1558 0.28657
## AlbumValence           1   192.60  4.9739 0.02942 *
## AlbumLoudness          1   178.70  0.2123 0.64658
## AvgTrackTempo          1   178.85  0.2640 0.60927
## AlbumAcousticness      1   188.13  3.4432 0.06835 .
## AlbumInstrumentalness  1   181.10  1.0331 0.31345
## AlbumLiveness          1   178.73  0.2210 0.63995
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Both of these tests suggest that we should drop the AlbumLoudness, AlbumLiveness and AvgTrackTempo from the model as we have very insignificant p values under both of these tests. Now to plot the model without these three indicators.

```
artistNumfol <- glm(ArtistNumFollowers ~ AlbumSpeechiness + AlbumAcousticness + AlbumValence +
                    AlbumInstrumentalness, data=SpotifyData1,
                    family = quasi(link = "log", variance = "mu^2"))
dropterm(artistNumfol, test=c("F"))
```

```
## Single term deletions
##
## Model:
## ArtistNumFollowers ~ AlbumSpeechiness + AlbumAcousticness + AlbumValence +
##     AlbumInstrumentalness
##                       Df Deviance F value   Pr(F)
## <none>                     180.96
## AlbumSpeechiness       1   186.54  1.9750 0.16475
## AlbumAcousticness      1   190.76  3.4667 0.06721 .
## AlbumValence           1   193.58  4.4649 0.03850 *
## AlbumInstrumentalness  1   183.77  0.9947 0.32234
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dropterm(artistNumfol, test=c("Chisq"))
```

```
## Single term deletions
##
## Model:
## ArtistNumFollowers ~ AlbumSpeechiness + AlbumAcousticness + AlbumValence +
##     AlbumInstrumentalness
##                       Df Deviance scaled dev. Pr(Chi)
## <none>                     180.96
## AlbumSpeechiness       1   186.54      2.4794 0.11534
## AlbumAcousticness      1   190.76      4.3522 0.03696 *
## AlbumValence           1   193.58      5.6052 0.01791 *
## AlbumInstrumentalness  1   183.77      1.2488 0.26378
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on these tests we shall remove AlbumInstrumentalness due to its high p-value. Now to carry out the same tests again using the remaining three track characteristics; AlbumSpeechiness, AlbumAcousticness and AlbumValence.

```
artistNumfol <- glm(ArtistNumFollowers ~ AlbumSpeechiness + AlbumAcousticness +
                    AlbumValence, data=SpotifyData1,
                    family = quasi(link = "log", variance = "mu^2"))
dropterm(artistNumfol, test=c("F"))
```

```
## Single term deletions
##
## Model:
## ArtistNumFollowers ~ AlbumSpeechiness + AlbumAcousticness + AlbumValence
##                     Df Deviance F value   Pr(F)
## <none>                   183.77
## AlbumSpeechiness     1   196.46  4.4883 0.03795 *
## AlbumAcousticness    1   199.25  5.4739 0.02239 *
## AlbumValence         1   195.15  4.0248 0.04900 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dropterm(artistNumfol, test=c("Chisq"))
```

```
## Single term deletions
##
## Model:
## ArtistNumFollowers ~ AlbumSpeechiness + AlbumAcousticness + AlbumValence
##                    Df Deviance scaled dev. Pr(Chi)
## <none>                  183.77
## AlbumSpeechiness    1   196.46      5.2918 0.02143 *
## AlbumAcousticness   1   199.25      6.4538 0.01107 *
## AlbumValence        1   195.15      4.7453 0.02938 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Once we have removed the AlbumInstrumentalness as a predictor we can see under both of these tests all of the p-values are now significant (≤ 0.05) so we are going to use this model to predict the ArtistNumFollowers and the coeffecients for these are:

```
coef(artistNumfol)
```

```
##       (Intercept)  AlbumSpeechiness AlbumAcousticness      AlbumValence
##         16.930916          6.520384         -2.826896         -4.312284
```

This means the AlbumValence is decreasing the ArtistNumFollowers, AlbumSpeechiness is increasing the ArtistNumFollowers and AlbumAccousticness.
Now lets see the model for ArtistPopularity with the most highly correlated predictors identified in the EDA:

```
artistPop <- glm(ArtistPopularity ~ AlbumValence + AlbumSpeechiness +
                   AlbumAcousticness + AlbumLoudness + AlbumInstrumentalness +
                   AvgTrackTempo, data=SpotifyData1,
                 family = quasi(link = "log", variance = "mu^2"))
dropterm(artistPop, test=c("Chisq"))
```

```
## Single term deletions
##
## Model:
## ArtistPopularity ~ AlbumValence + AlbumSpeechiness + AlbumAcousticness +
##     AlbumLoudness + AlbumInstrumentalness + AvgTrackTempo
##                         Df Deviance scaled dev.  Pr(Chi)
## <none>                       4.5105
## AlbumValence             1   4.9278      7.0930 0.007739 **
## AlbumSpeechiness         1   4.6838      2.9467 0.086052 .
## AlbumAcousticness        1   4.5742      1.0827 0.298083
## AlbumLoudness            1   4.5189      0.1428 0.705526
## AlbumInstrumentalness    1   4.5138      0.0561 0.812706
## AvgTrackTempo            1   4.5453      0.5926 0.441416
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dropterm(artistPop, test=c("F"))
```

```
## Single term deletions
##
## Model:
## ArtistPopularity ~ AlbumValence + AlbumSpeechiness + AlbumAcousticness +
##     AlbumLoudness + AlbumInstrumentalness + AvgTrackTempo
##                         Df Deviance F value   Pr(F)
## <none>                       4.5105
## AlbumValence             1   4.9278  5.7371 0.01964 *
```

```
## AlbumSpeechiness       1   4.6838  2.3834 0.12772
## AlbumAcousticness      1   4.5742  0.8758 0.35299
## AlbumLoudness          1   4.5189  0.1155 0.73513
## AlbumInstrumentalness  1   4.5138  0.0454 0.83196
## AvgTrackTempo          1   4.5453  0.4793 0.49132
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Once again we can see that the AlbumLoudness, AlbumInstrumentalness and AvgTrackTempo have highly insignificant p-values for both of these tests so we should drop them and see if we get a more accurate model.

```
artistPop <- glm(ArtistPopularity ~ AlbumValence + AlbumSpeechiness +
            AlbumAcousticness, data=SpotifyData1,
          family = quasi(link = "log", variance = "mu^2"))
dropterm(artistPop, test=c("Chisq"))
```

```
## Single term deletions
##
## Model:
## ArtistPopularity ~ AlbumValence + AlbumSpeechiness + AlbumAcousticness
##                    Df Deviance scaled dev. Pr(Chi)
## <none>                  4.5657
## AlbumValence        1   5.0411      8.3023 0.00396 **
## AlbumSpeechiness    1   4.8636      5.2022 0.02256 *
## AlbumAcousticness   1   4.7796      3.7362 0.05324 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dropterm(artistPop, test=c("F"))
```

```
## Single term deletions
##
## Model:
## ArtistPopularity ~ AlbumValence + AlbumSpeechiness + AlbumAcousticness
##                    Df Deviance F value   Pr(F)
## <none>                  4.5657
## AlbumValence        1   5.0411  6.7685 0.01148 *
## AlbumSpeechiness    1   4.8636  4.2412 0.04346 *
## AlbumAcousticness   1   4.7796  3.0460 0.08566 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now we have all of the predictors significant or extremely close to significant so this leads us to believe this is the best model for predicting the ArtistPopularity. The coeffecients for each of these predictors are:

```
coef(artistPop)
```

```
##       (Intercept)      AlbumValence  AlbumSpeechiness AlbumAcousticness
##         4.4590226        -0.6583867         0.9544420        -0.2713191
```

This implies that AlbumValence and AlbumAcousticness are decreasing the ArtistPopularity, and that AlbumSpeechiness is increasing it.

Now let's plot the model for AlbumPopularity selecting characteristics to initially use as predictors in the same way as before.

```
albumPop <- glm(AlbumPopularity ~ AlbumValence + AlbumSpeechiness +
            AlbumAcousticness + AlbumLiveness + AvgTrackTempo,
            data=SpotifyData1,
          family = quasi(link = "log", variance = "mu^2"))
dropterm(albumPop, test=c("Chisq"))
```

```
## Single term deletions
##
## Model:
## AlbumPopularity ~ AlbumValence + AlbumSpeechiness + AlbumAcousticness +
##     AlbumLiveness + AvgTrackTempo
##                   Df Deviance scaled dev.  Pr(Chi)
## <none>                 7.0848
## AlbumValence       1   8.0678     10.4940 0.001198 **
## AlbumSpeechiness   1   7.8316      7.9723 0.004750 **
## AlbumAcousticness  1   7.4582      3.9866 0.045864 *
## AlbumLiveness      1   7.1026      0.1901 0.662849
## AvgTrackTempo      1   7.0866      0.0189 0.890719
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dropterm(albumPop, test=c("F"))
```

```
## Single term deletions
##
## Model:
## AlbumPopularity ~ AlbumValence + AlbumSpeechiness + AlbumAcousticness +
##     AlbumLiveness + AvgTrackTempo
##                   Df Deviance F value    Pr(F)
## <none>                 7.0848
## AlbumValence       1   8.0678  8.7415 0.004374 **
## AlbumSpeechiness   1   7.8316  6.6409 0.012320 *
## AlbumAcousticness  1   7.4582  3.3208 0.073154 .
## AlbumLiveness      1   7.1026  0.1583 0.692037
## AvgTrackTempo      1   7.0866  0.0157 0.900607
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can see that under both of these tests AvgTrackTempo and AlbumLiveness are highly insignificant so we should try dropping these to see how it alters the model.

```
albumPop <- glm(AlbumPopularity ~ AlbumValence + AlbumSpeechiness +
            AlbumAcousticness, data=SpotifyData1,
          family = quasi(link = "log", variance = "mu^2"))
dropterm(albumPop, test=c("Chisq"))
```

```
## Single term deletions
##
## Model:
## AlbumPopularity ~ AlbumValence + AlbumSpeechiness + AlbumAcousticness
##                   Df Deviance scaled dev.   Pr(Chi)
## <none>                 7.1028
## AlbumValence       1   8.1481     11.5599 0.0006739 ***
## AlbumSpeechiness   1   8.0117     10.0515 0.0015222 **
## AlbumAcousticness  1   7.4769      4.1376 0.0419404 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dropterm(albumPop, test=c("F"))
```

```
## Single term deletions
##
## Model:
## AlbumPopularity ~ AlbumValence + AlbumSpeechiness + AlbumAcousticness
##                   Df Deviance F value    Pr(F)
```

```
## <none>                 7.1028
## AlbumValence       1   8.1481  9.5663 0.002920 **
## AlbumSpeechiness   1   8.0117  8.3181 0.005318 **
## AlbumAcousticness  1   7.4769  3.4241 0.068798 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here we have that all of the predictors are significant or very close to significant (in the F test) under these tests. This means we should be using the model with the following predictors; "AlbumValence", "AlbumSpeechiness" and "AlbumAcousticness".

Now let's see the coefficients for these variables:

```
coef(albumPop)
```

```
##       (Intercept)      AlbumValence  AlbumSpeechiness AlbumAcousticness
##         4.4338679        -0.9885016         1.6605352        -0.3648088
```

We can see that the AlbumValence and AlbumAcousticness decreases the AlbumPopularity and AlbumSpeechiness increases the AlbumPopularity.

Section 2.4) The models

Using the 3 variables we defined as to be popular we now have 3 models to use to predict these and they are:

$$ArtistNumFollowers = \alpha + \beta_1 AlbumValence + \beta_2 AlbumSpeechiness + \beta_3 AlbumAcousticness$$
$$ArtistPopularity = \alpha + \beta_1 AlbumValence + \beta_2 AlbumSpeechiness + \beta_3 AlbumAcousticness$$
$$AlbumPopularity = \alpha + \beta_1 AlbumValence + \beta_2 AlbumSpeechiness + \beta_3 AlbumAcousticness$$

Looking above we notice that under all of these models, the AlbumValence decreased all 3 of the variables we defined as popularity, the AlbumSpeechiness increased all 3 variables we defined as popularity and the AlbumAcousticness decreased all 3 variables we defined as popularity. This leads us to believe that investors should invest in artists that make music that has a low Valence, low Acousticness and high speechiness to have the best chance at being popular.

Section 2.4.1) Model limitations

1) We have a very small database for the 2000's and 2010's

2) The tests we conducted on the models could be improved, this is due to AvgTrackTempo has a high correlation with the three variables we defined as popular but haven't included it due to the tests conducted

Section 2.4.2) Model testing

We will now use the models devised above to predict popularity ratings for the artists and albums in our test set, comparing the predictions to the actual popularity values. Since the models were built using training data from the two most recent decades, 2000s and 2010s, we shall only test these. The error used here is the mean absolute error (MAE).

```
errArtFol <- 0
errArtPop <- 0
errAlbPop <- 0

curRow <- filter(SpotifyData1, Artist == "The Streets", AlbumName == "Original Pirate Material")
a <- exp(predict(artistNumfol, newdata = curRow))
errArtFol <- errArtFol + abs(curRow$ArtistNumFollowers - a)
b <- exp(predict(artistPop, newdata = curRow))
errArtPop <- errArtPop + abs(curRow$ArtistPopularity - b)
```

```
c <- exp(predict(albumPop, newdata = curRow))
errAlbPop <- errAlbPop + abs(curRow$AlbumPopularity - c)

curRow <- filter(SpotifyData1, Artist == "Missy Elliott", AlbumName == "Miss E...So Addictive")
a <- exp(predict(artistNumfol, newdata = curRow))
errArtFol <- errArtFol + abs(curRow$ArtistNumFollowers - a)
b <- exp(predict(artistPop, newdata = curRow))
errArtPop <- errArtPop + abs(curRow$ArtistPopularity - b)
c <- exp(predict(albumPop, newdata = curRow))
errAlbPop <- errAlbPop + abs(curRow$AlbumPopularity - c)

curRow <- filter(SpotifyData1, Artist == "Public Service Broadcasting",
                 AlbumName == "The Race For Space")
a <- exp(predict(artistNumfol, newdata = curRow))
errArtFol <- errArtFol + abs(curRow$ArtistNumFollowers - a)
b <- exp(predict(artistPop, newdata = curRow))
errArtPop <- errArtPop + abs(curRow$ArtistPopularity - b)
c <- exp(predict(albumPop, newdata = curRow))
errAlbPop <- errAlbPop + abs(curRow$AlbumPopularity - c)

curRow <- filter(SpotifyData1, Artist == "Daft Punk", AlbumName == "Random Access Memories")
a <- exp(predict(artistNumfol, newdata = curRow))
errArtFol <- errArtFol + abs(curRow$ArtistNumFollowers - a)
b <- exp(predict(artistPop, newdata = curRow))
errArtPop <- errArtPop + abs(curRow$ArtistPopularity - b)
c <- exp(predict(albumPop, newdata = curRow))
errAlbPop <- errAlbPop + abs(curRow$AlbumPopularity - c)

d <- errArtFol/4; e <- errArtPop/4; f <- errAlbPop/4
g <- as.data.frame(cbind(d,e,f))
row.names(g) <- c("MAE error")
g %>%
  kable(col.names = c("Artist Followers", "Artist Popularity", "Album Popularity"),
      align = c('c','c','c'), digits = 2) %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed")) %>%
  column_spec(1, width = "5cm", bold = TRUE,
              border_left = TRUE, border_right = TRUE) %>%
  column_spec(4, border_right = TRUE)
```

	Artist Followers	Artist Popularity	Album Popularity
MAE error	2458391	14.65	12.26

Section 3) Song Recommendation System (Task 2)

Section 3.1) Data cleaning

```
library(readxl)
Spotify <- read_excel("edited_spotify.xlsx")
library(lubridate)
Spotify$AlbumReleaseDate <- parse_date_time(Spotify$AlbumReleaseDate,
                                            orders=c("y","ym","ymd"))
AlbumReleaseYear <- year(Spotify$AlbumReleaseDate)
Spotify <- data.frame(Spotify,AlbumReleaseYear, check.names = TRUE)
AlbumReleaseDecade <- 10*floor(Spotify$AlbumReleaseYear/10)
Spotify <- data.frame(Spotify, AlbumReleaseDecade, check.names = TRUE)

library(dplyr)
SpotifyData <- {Spotify %>%
    group_by(Artist,ArtistID,ArtistGenres,ArtistPopularity,ArtistNumFollowers,
             AlbumName,AlbumID,AlbumBestChartPosition,AlbumWeeksOnChart,
             AlbumWeeksNumberOne,AlbumPopularity,AlbumReleaseDate,
             AlbumReleaseYear, AlbumReleaseDecade) %>%
    summarize(AlbumAcousticness = mean(TrackAcousticness),
              AlbumValence = mean(TrackValence),
              AvgTrackDuration = mean(TrackDuration),
              AlbumDanceability = mean(TrackDanceability),
              AlbumEnergy = mean(TrackEnergy),
              AlbumLoudness = mean(TrackLoudness),
              AvgTrackMode = mean(TrackMode),
              AlbumSpeechiness = mean(TrackSpeechiness),
              AlbumInstrumentalness = mean(TrackInstrumentalness),
              AlbumLiveness = mean(TrackLiveness),
              AvgTrackTempo = mean(TrackTempo),
              AvgTrackTimeSignature = mean(TrackTimeSignature))}
```

Section 3.2) Removing test albums

We need to be able to test the system of recommendations by having some test albums and then use the rest of the albums for the models training. The code below is removing all 12 of our test albums but storing each of them to a variable so we can test the model.

```
# Led Zeppelin: Led Zeppelin
x1 <- SpotifyData[84,]
# Madonna: Madonna
x2 <- SpotifyData[98,]
# Andy Stewart: The Scottish Soldier
x3 <- SpotifyData[8,]
# Nick Lowe: Jesus of Cool
x4 <- SpotifyData[114,]
# The Undertones: The Undertones
x5 <- SpotifyData[199,]
# Squeeze: Argybargy
x6 <- SpotifyData[155,]
# Portishead: Dummy
x7 <- SpotifyData[129,]
# The Beautiful South: Blue Is The Colour
x8 <- SpotifyData[169,]
# The Streets: Original Pirate Material
x9 <- SpotifyData[197,]
# Missy Elliott: Miss E...So Addictive
x10 <- SpotifyData[104,]
# Public Service Broadcasting: The Race For Space
x11 <- SpotifyData[133,]
# Daft Punk: Random Access Memories
x12 <- SpotifyData[30,]

SpotifyData <- SpotifyData[c(-84,-98,-8,-114,-199,-155,-129,-169, -197, -133, -104, -30),]
```

Section 3.3) The system code

```
Database <- function(AlbName) {
  x <- SpotifyData %>% filter(AlbumName == AlbName)
  if(length(x) == 0) {
    return("Album not in the database")
  }
  else {
    x <<- x
    predict(x)
  }
}
```

This database function has an input of the AlbumName, if there is an album in the database with the same name then we will use the function below to predict. If there is no album with the same name in the database then we return “Album not in the database”. Then if we get that message you have to use the “predict” function below to predict the next album and song.

```
predict <- function(x) {
# This is to find the album in the database
  a <- FALSE
  b <- x[1,7]
  for(i in 1:length(t(SpotifyData[,1]))) {
# This is remove the album from the database if it is already in it
    if(SpotifyData[i,7] == b) {
      a <- TRUE
      SpotifyData1 <- SpotifyData[-i,]
     }
  }
# This is to add the album to the database if it is not in it
  if(a == FALSE) {
    SpotifyData1 <- SpotifyData
  }
# Now we are going to cluster the data based of the 9 descriptive variables
# Some of these include: Valence, Danceability, Energy,...
  d <- c(15,16,18,19,20,25,24,23,22)
  SpotifyData2 <- SpotifyData1[,d]
# Now clustering the data and attaching the cluster to the database
  k <- kmeans(SpotifyData2, 20, nstart=25, iter.max = 1000)
  Cluster <- as.data.frame(k[["cluster"]])
  SpotifyData1 <- bind_cols(SpotifyData1,Cluster)
# Now finding the cluster the new data point is in
  n <- as.numeric(length(SpotifyData1[,1]))
  e <- as.numeric(SpotifyData1[n,27])
  SpotifyData1 <- as.data.frame(SpotifyData1)
# Now creating a new database with the albums from the same cluster as the current one
  SpotifyData3 <- SpotifyData1 %>% filter(Cluster == e)
# Now randomly selecting an album from that database
  x <- sample(1:length(SpotifyData3[,1]),1)
# Finally displaying the next album she should listen to, the genre and the artist
  print(paste0("Your album recommendation: ", SpotifyData3[as.numeric(x),c(1)],
               " - ", SpotifyData3[as.numeric(x),c(6)]))
  q <- SpotifyData3[as.numeric(x),7]
  SpotifyData4 <- Spotify %>% filter(AlbumID == q)
  y <- sample(1:length(SpotifyData4[,1]),1)
  print(paste0("Your song recommendation: ",SpotifyData4[as.numeric(y),c(13)]))
}
```

This function will recommend an album to the user to listen to next then we predict them a song to listen to first from that album. The function does this by using K-means clustering, so it is grouping the database based of the 9 predictive variables. Then it is creating a new database with only the albums with the same cluster and then removing the album inputted (this is to stop predicting the album the user is currently listening to). Then it randomly chooses an album from within that cluster, then randomly chooses a song from that album to start with.

Section 3.4) Predictions for test albums

Prediction for Led Zeppelin: Led Zeppelin

```
predict(x=x1)
```

```
## [1] "Your album recommendation: Alicia Keys - Songs In A Minor"
## [1] "Your song recommendation: Piano & I"
```

Prediction for Madonna: Madonna

```
predict(x=x2)
```

```
## [1] "Your album recommendation: The Beatles - Help!"
## [1] "Your song recommendation: Act Naturally"
```

Prediction for Andy Stewart: The Scottish Soldier

```
predict(x=x3)
```

```
## [1] "Your album recommendation: The Beatles - Help!"
## [1] "Your song recommendation: Tell Me What You See"
```

Prediction for Nick Lowe: Jesus of Cool

```
predict(x=x4)
```

```
## [1] "Your album recommendation: Gorillaz - Plastic Beach"
## [1] "Your song recommendation: On Melancholy Hill"
```

Prediction for The Undertones: The Undertones

```
predict(x=x5)
```

```
## [1] "Your album recommendation: Alicia Keys - Songs In A Minor"
## [1] "Your song recommendation: Girlfriend"
```

Prediction for Squeeze: Argybargy

```
predict(x=x6)
```

```
## [1] "Your album recommendation: The Stone Roses - The Stone Roses"
## [1] "Your song recommendation: This Is the One"
```

Prediction for Portishead: Dummy

```
predict(x=x7)
```

```
## [1] "Your album recommendation: Ben Folds - Rockin' The Suburbs"
## [1] "Your song recommendation: Annie Waits"
```

Prediction for The Beautiful South: Blue Is The Colour

```
predict(x=x8)
```

```
## [1] "Your album recommendation: The Boo Radleys - C'Mon Kids"
## [1] "Your song recommendation: Everything Is Sorrow"
```

Prediction for The Streets: Original Pirate Material

```
predict(x=x9)
```

```
## [1] "Your album recommendation: The Pogues - If I Should Fall From Grace With God (Expanded)"
## [1] "Your song recommendation: Sketches of Spain"
```

Prediction for Missy Elliot: Miss E...So Addictive

```
predict(x=x10)
```

```
## [1] "Your album recommendation: The Pogues - If I Should Fall From Grace With God (Expanded)"
## [1] "Your song recommendation: Sketches of Spain"
```

Prediction for Public Service Broadcasting: The Race For Space

```
predict(x=x11)
```

```
## [1] "Your album recommendation: Al Green - Let's Stay Together"
## [1] "Your song recommendation: How Can You Mend a Broken Heart"
```

Prediction for Daft Punk: Random Access Memories

```
predict(x=x12)
```

```
## [1] "Your album recommendation: The Stone Roses - The Stone Roses"
## [1] "Your song recommendation: Don't Stop"
```

Section 3.5) Limitations

1) If we input an album that has nothing like it in the database then the recommendation will not be accurate. An example is Daft Punk, there is not much like Daft Punk in the database so the recommendation may not be accurate.

2) We couldn't implement a way for the algorithm to remove the previous album listened to. This means if you listened to Daft Punk then if we recommended ABBA then you may get predicted Daft Punk again which is not optimal.

3) If the album inputted does not have the same rows in the same order as the Spotify database then the algorithm will not work.

4) We have chosen to use 20 clusters which was chosen through trial and error, we couldn't find a way to implement this in the algorithm.