Combined_Codes

Zonghong Yu, Yicheng Guo, Yilin Yang, Huiting Song, Shiyu Wang

2022-12-12

```
df <- read.csv("genres_v2.csv",stringsAsFactors=FALSE)
head(df)
```

```
##   danceability energy key loudness mode speechiness acousticness
## 1        0.831  0.814   2   -7.364    1      0.4200       0.0598
## 2        0.719  0.493   8   -7.230    1      0.0794       0.4010
## 3        0.850  0.893   5   -4.783    1      0.0623       0.0138
## 4        0.476  0.781   0   -4.710    1      0.1030       0.0237
## 5        0.798  0.624   2   -7.668    1      0.2930       0.2170
## 6        0.721  0.568   0  -11.295    1      0.4140       0.0452
##   instrumentalness liveness valence   tempo           type
## 1         1.34e-02   0.0556  0.3890 156.985 audio_features
## 2         0.00e+00   0.1180  0.1240 115.080 audio_features
## 3         4.14e-06   0.3720  0.0391 218.050 audio_features
## 4         0.00e+00   0.1140  0.1750 186.948 audio_features
## 5         0.00e+00   0.1660  0.5910 147.988 audio_features
## 6         2.12e-01   0.1280  0.1090 144.915 audio_features
##                       id                                  uri
## 1 2Vc6NJ9PW9gD9q343XFRKx spotify:track:2Vc6NJ9PW9gD9q343XFRKx
## 2 7pgJBLVz5VmnL7uGHmRj6p spotify:track:7pgJBLVz5VmnL7uGHmRj6p
## 3 0vSWgAlfpye0WCGeNmuNhy spotify:track:0vSWgAlfpye0WCGeNmuNhy
## 4 0VSXnJqQkwuH2ei1nOQ1nu spotify:track:0VSXnJqQkwuH2ei1nOQ1nu
## 5 4jCeguq9rMTlbMmPHuO7S3 spotify:track:4jCeguq9rMTlbMmPHuO7S3
## 6 6fsypiJHyWmeINsOLC1cos spotify:track:6fsypiJHyWmeINsOLC1cos
##                                            track_href
## 1 https://api.spotify.com/v1/tracks/2Vc6NJ9PW9gD9q343XFRKx
## 2 https://api.spotify.com/v1/tracks/7pgJBLVz5VmnL7uGHmRj6p
## 3 https://api.spotify.com/v1/tracks/0vSWgAlfpye0WCGeNmuNhy
## 4 https://api.spotify.com/v1/tracks/0VSXnJqQkwuH2ei1nOQ1nu
## 5 https://api.spotify.com/v1/tracks/4jCeguq9rMTlbMmPHuO7S3
## 6 https://api.spotify.com/v1/tracks/6fsypiJHyWmeINsOLC1cos
##                                              analysis_url duration_ms
## 1 https://api.spotify.com/v1/audio-analysis/2Vc6NJ9PW9gD9q343XFRKx      124539
## 2 https://api.spotify.com/v1/audio-analysis/7pgJBLVz5VmnL7uGHmRj6p      224427
## 3 https://api.spotify.com/v1/audio-analysis/0vSWgAlfpye0WCGeNmuNhy       98821
## 4 https://api.spotify.com/v1/audio-analysis/0VSXnJqQkwuH2ei1nOQ1nu      123661
## 5 https://api.spotify.com/v1/audio-analysis/4jCeguq9rMTlbMmPHuO7S3      123298
## 6 https://api.spotify.com/v1/audio-analysis/6fsypiJHyWmeINsOLC1cos      112511
##   time_signature     genre                                        song_name
## 1              4 Dark Trap                            Mercury: Retrograde
## 2              4 Dark Trap                                      Pathology
## 3              4 Dark Trap                                       Symbiote
## 4              3 Dark Trap ProductOfDrugs (Prod. The Virus and Antidote)
## 5              4 Dark Trap                                          Venom
## 6              4 Dark Trap                                        Gatteka
##   Unnamed..0 title
## 1         NA
## 2         NA
## 3         NA
## 4         NA
## 5         NA
## 6         NA
```

```
names(df)
```

```
##  [1] "danceability"     "energy"           "key"              "loudness"
##  [5] "mode"             "speechiness"      "acousticness"     "instrumentalness"
##  [9] "liveness"         "valence"          "tempo"            "type"
## [13] "id"               "uri"              "track_href"       "analysis_url"
## [17] "duration_ms"      "time_signature"   "genre"            "song_name"
## [21] "Unnamed..0"       "title"
```

```
df = subset(df, select = -c(uri, id, track_href, Unnamed..0, duration_ms, analysis_url, title, t
ype) )
```

```
head(df)
```

```
##   danceability energy key loudness mode speechiness acousticness
## 1        0.831  0.814   2   -7.364    1      0.4200       0.0598
## 2        0.719  0.493   8   -7.230    1      0.0794       0.4010
## 3        0.850  0.893   5   -4.783    1      0.0623       0.0138
## 4        0.476  0.781   0   -4.710    1      0.1030       0.0237
## 5        0.798  0.624   2   -7.668    1      0.2930       0.2170
## 6        0.721  0.568   0  -11.295    1      0.4140       0.0452
##   instrumentalness liveness valence   tempo time_signature     genre
## 1         1.34e-02   0.0556  0.3890 156.985              4 Dark Trap
## 2         0.00e+00   0.1180  0.1240 115.080              4 Dark Trap
## 3         4.14e-06   0.3720  0.0391 218.050              4 Dark Trap
## 4         0.00e+00   0.1140  0.1750 186.948              3 Dark Trap
## 5         0.00e+00   0.1660  0.5910 147.988              4 Dark Trap
## 6         2.12e-01   0.1280  0.1090 144.915              4 Dark Trap
##                                     song_name
## 1                           Mercury: Retrograde
## 2                                     Pathology
## 3                                      Symbiote
## 4 ProductOfDrugs (Prod. The Virus and Antidote)
## 5                                         Venom
## 6                                       Gatteka
```

```
df_dup = df[duplicated(df$song_name), ]
head(df_dup)
```

```
##     danceability energy key loudness mode speechiness acousticness
## 81         0.909  0.573   5   -6.856    1      0.1810      0.03320
## 294        0.756  0.746   1   -6.397    1      0.1420      0.23600
## 426        0.705  0.648   4  -10.467    0      0.1410      0.00476
## 489        0.675  0.628   4   -6.165    0      0.0838      0.00997
## 580        0.945  0.587   2   -5.104    1      0.2710      0.25300
## 669        0.733  0.513   6   -5.068    0      0.0389      0.55900
##     instrumentalness liveness valence   tempo time_signature     genre
## 81          1.63e-02   0.6570  0.3920 144.946              4 Dark Trap
## 294         0.00e+00   0.0999  0.0793 207.956              4 Dark Trap
## 426         5.95e-04   0.3730  0.3980 130.037              4 Dark Trap
## 489         0.00e+00   0.0868  0.5110 179.992              4 Dark Trap
## 580         1.01e-06   0.2640  0.5090 131.969              4 Dark Trap
## 669         1.48e-05   0.1740  0.5640 159.944              4 Dark Trap
##                song_name
## 81                 Venom
## 294      Story: No Title
## 426              No Teeth
## 489 Make It Through Fall
## 580                Killer
## 669             Nightmare
```

```
dup <- df[df$song_name == "Venom",]
```

```
df = df[!duplicated(df$song_name), ]
```

```
head(df)
```

```
##   danceability energy key loudness mode speechiness acousticness
## 1        0.831  0.814   2   -7.364    1      0.4200       0.0598
## 2        0.719  0.493   8   -7.230    1      0.0794       0.4010
## 3        0.850  0.893   5   -4.783    1      0.0623       0.0138
## 4        0.476  0.781   0   -4.710    1      0.1030       0.0237
## 5        0.798  0.624   2   -7.668    1      0.2930       0.2170
## 6        0.721  0.568   0  -11.295    1      0.4140       0.0452
##   instrumentalness liveness valence   tempo time_signature     genre
## 1         1.34e-02   0.0556  0.3890 156.985              4 Dark Trap
## 2         0.00e+00   0.1180  0.1240 115.080              4 Dark Trap
## 3         4.14e-06   0.3720  0.0391 218.050              4 Dark Trap
## 4         0.00e+00   0.1140  0.1750 186.948              3 Dark Trap
## 5         0.00e+00   0.1660  0.5910 147.988              4 Dark Trap
## 6         2.12e-01   0.1280  0.1090 144.915              4 Dark Trap
##                                     song_name
## 1                           Mercury: Retrograde
## 2                                     Pathology
## 3                                      Symbiote
## 4 ProductOfDrugs (Prod. The Virus and Antidote)
## 5                                         Venom
## 6                                       Gatteka
```

```
sum(duplicated(df$song_name))
```

```
## [1] 0
```

```
library(tidyverse) #for data cleaning and manipulation
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.3.6     ✓ purrr   0.3.4
## ✓ tibble  3.1.8     ✓ dplyr   1.0.9
## ✓ tidyr   1.2.0     ✓ stringr 1.4.0
## ✓ readr   2.1.2     ✓ forcats 0.5.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
#library(rccdates) #for converting date variables
library(wordcloud) #for creating word cloud visualizations
```

```
## Loading required package: RColorBrewer
```

```
library(ggplot2) #for data visualizations
library(tm) #used for text mining
```

```
## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(RColorBrewer) #color schemes for plots
library(SnowballC) #for text stemming
library(corrplot) #for correlation matrix visualization
```

```
## Warning: package 'corrplot' was built under R version 4.2.2
```

```
## corrplot 0.92 loaded
```

```
spotify <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/maste
r/data/2020/2020-01-21/spotify_songs.csv')
```

```
## Rows: 32833 Columns: 23
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (10): track_id, track_name, track_artist, track_album_id, track_album_na...
## dbl (13): track_popularity, danceability, energy, key, loudness, mode, speec...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
write.csv(spotify, "spotify.csv")
```

```
#removing columns 1,5,6,8,& 9
spotify <- spotify[,-c(1,5,6,8,9)]
#separating track_album_release_date
spotify <- spotify%>%separate(track_album_release_date,c("release_year", "release_month", "relea
se_day"), sep="-")
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 1886 rows [152,
## 750, 751, 752, 754, 756, 760, 766, 769, 780, 783, 786, 787, 788, 789, 790, 794,
## 799, 805, 806, ...].
```

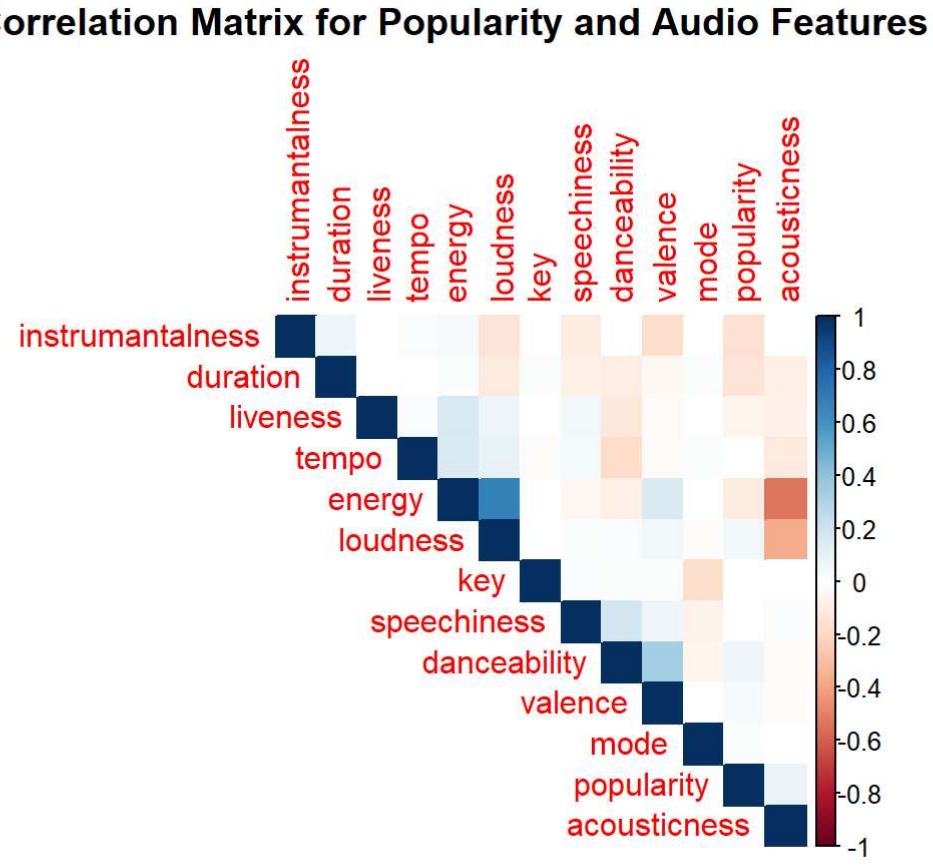
```
#deleting release_month and release_day
spotify <- spotify[,-c(5,6)]

#changing year to a factor
spotify$release_year <- as.factor(spotify$release_year)

#changing genre to a factor
spotify$playlist_genre <- as.factor(spotify$playlist_genre)

#changing subgenre to a factor
spotify$playlist_subgenre <- as.factor(spotify$playlist_subgenre)

#simplifying variable names
names(spotify) <- c("name", "artist", "popularity", "year", "genre", "subgenre", "danceability",
"energy", "key", "loudness", "mode", "speechiness", "acousticness", "instrumantalness", "livenes
s", "valence", "tempo", "duration")
```

```
spotify = na.omit(spotify)
sum(is.na(spotify))
```

```
## [1] 0
```

```
write.csv(spotify, "spotify_cleaned.csv")
```

```
spotify %>%
  select(popularity, danceability, energy, key, loudness, mode, speechiness, acousticness, instr
umantalness, liveness, valence, tempo, duration) %>%
  cor() %>%
  corrplot(method = 'color', order = 'hclust',  type = 'upper',
           diag = TRUE, main = 'Correlation Matrix for Popularity and Audio Features',
           mar = c(2,2,2,2))
```




```
title2 <- Corpus(VectorSource(spotify$name))
# Convert the text to lower case
title2 <- tm_map(title2, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(title2, content_transformer(tolower)):
## transformation drops documents
```

```
# Remove numbers
title2 <- tm_map(title2, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(title2, removeNumbers): transformation drops
## documents
```

```
# Remove english common stopwords
title2 <- tm_map(title2, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(title2, removeWords, stopwords("english")):
## transformation drops documents
```

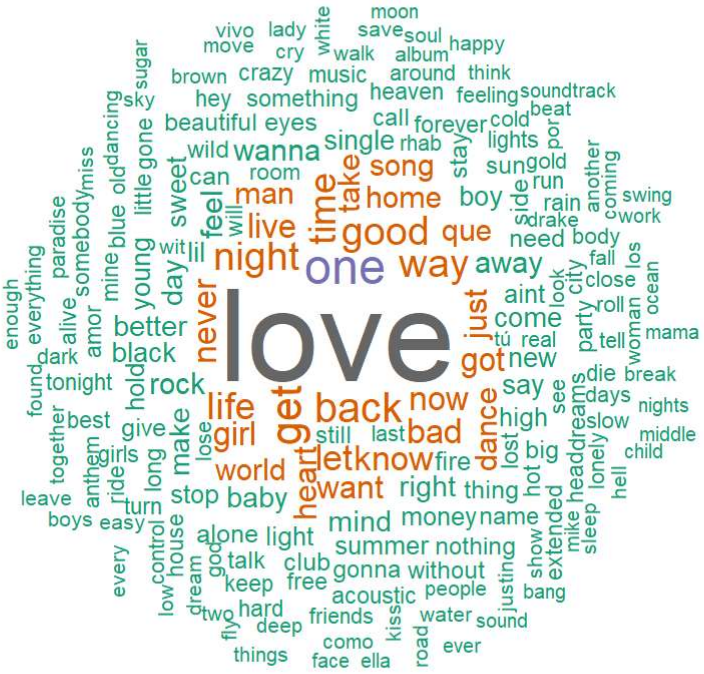
```
# Remove punctuations
title2 <- tm_map(title2, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(title2, removePunctuation): transformation drops
## documents
```

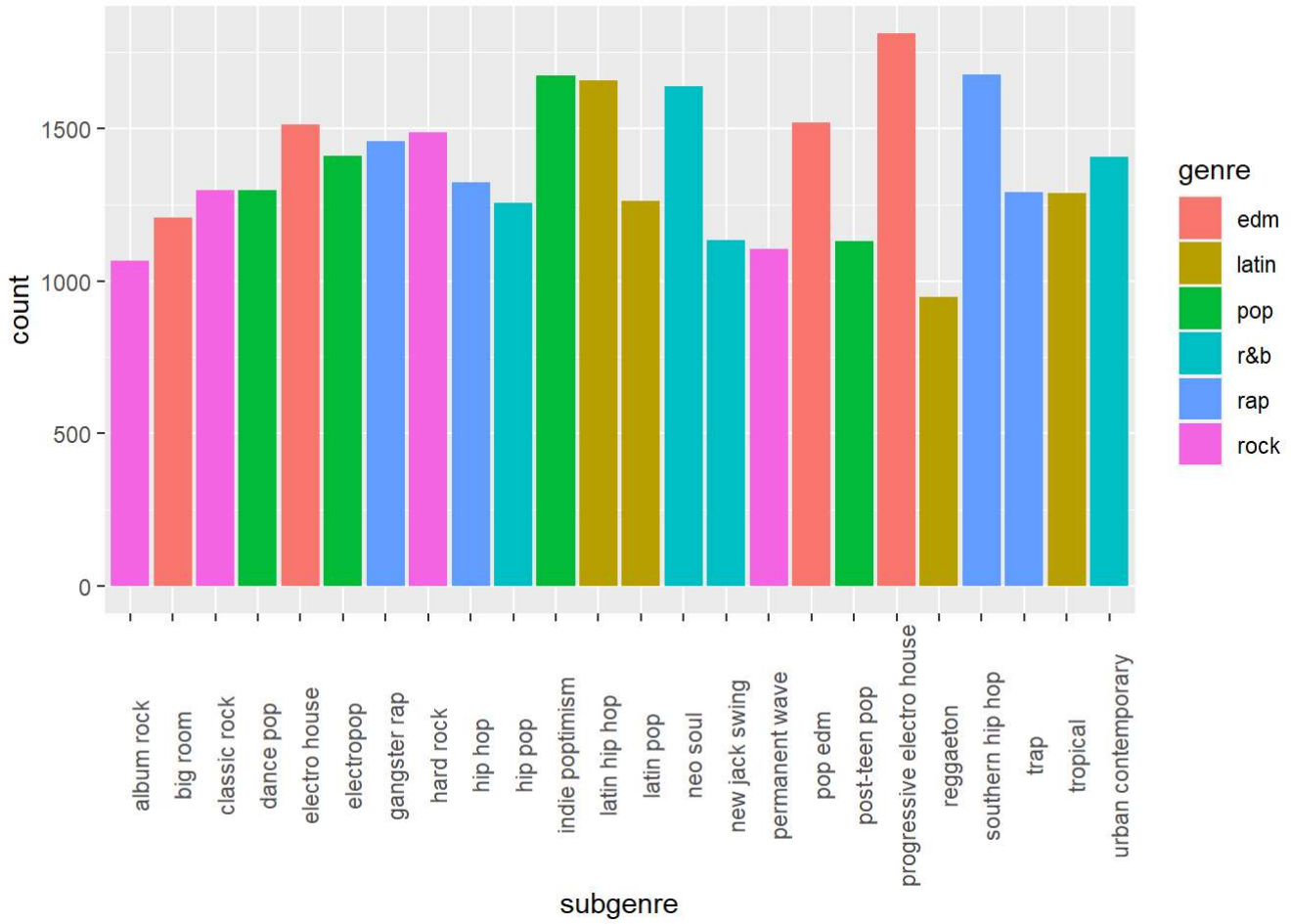
```
# Remove other data specific stop words
title2 <- tm_map(title2, removeWords, c("feat","edit", "version", "radio", "remix", "remastere
d", "mix","like", "original", "remaster"))
```

```
## Warning in tm_map.SimpleCorpus(title2, removeWords, c("feat", "edit",
## "version", : transformation drops documents
```

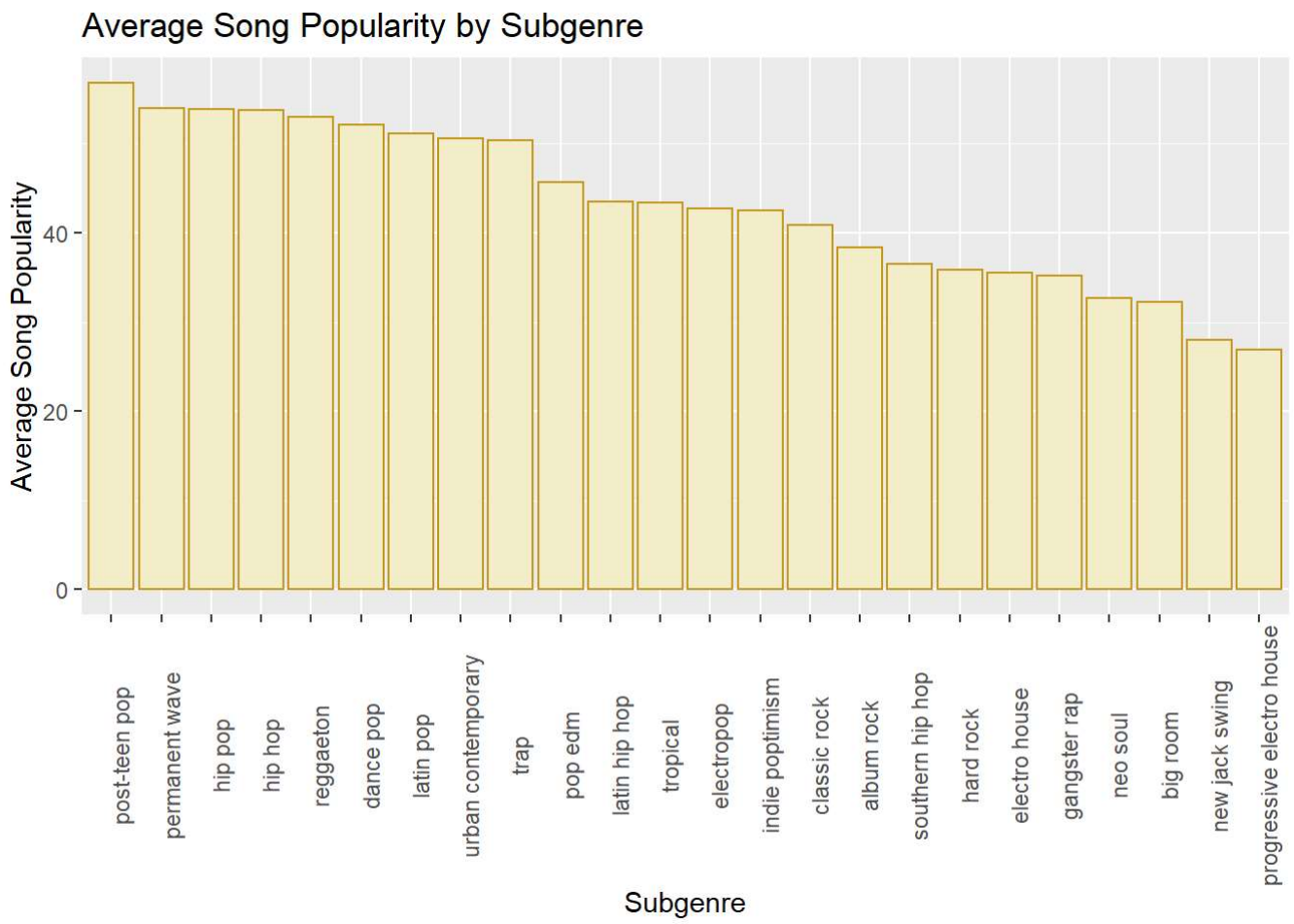
```
title2_dtm <- DocumentTermMatrix(title2)
title2_freq <- colSums(as.matrix(title2_dtm))
freq2 <- sort(colSums(as.matrix(title2_dtm)), decreasing=TRUE)
title2_wf <- data.frame(word=names(title2_freq), freq=title2_freq)

#create word cloud
set.seed(1234)
wordcloud(words = title2_wf$word, freq = title2_wf$freq, min.freq =1,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```



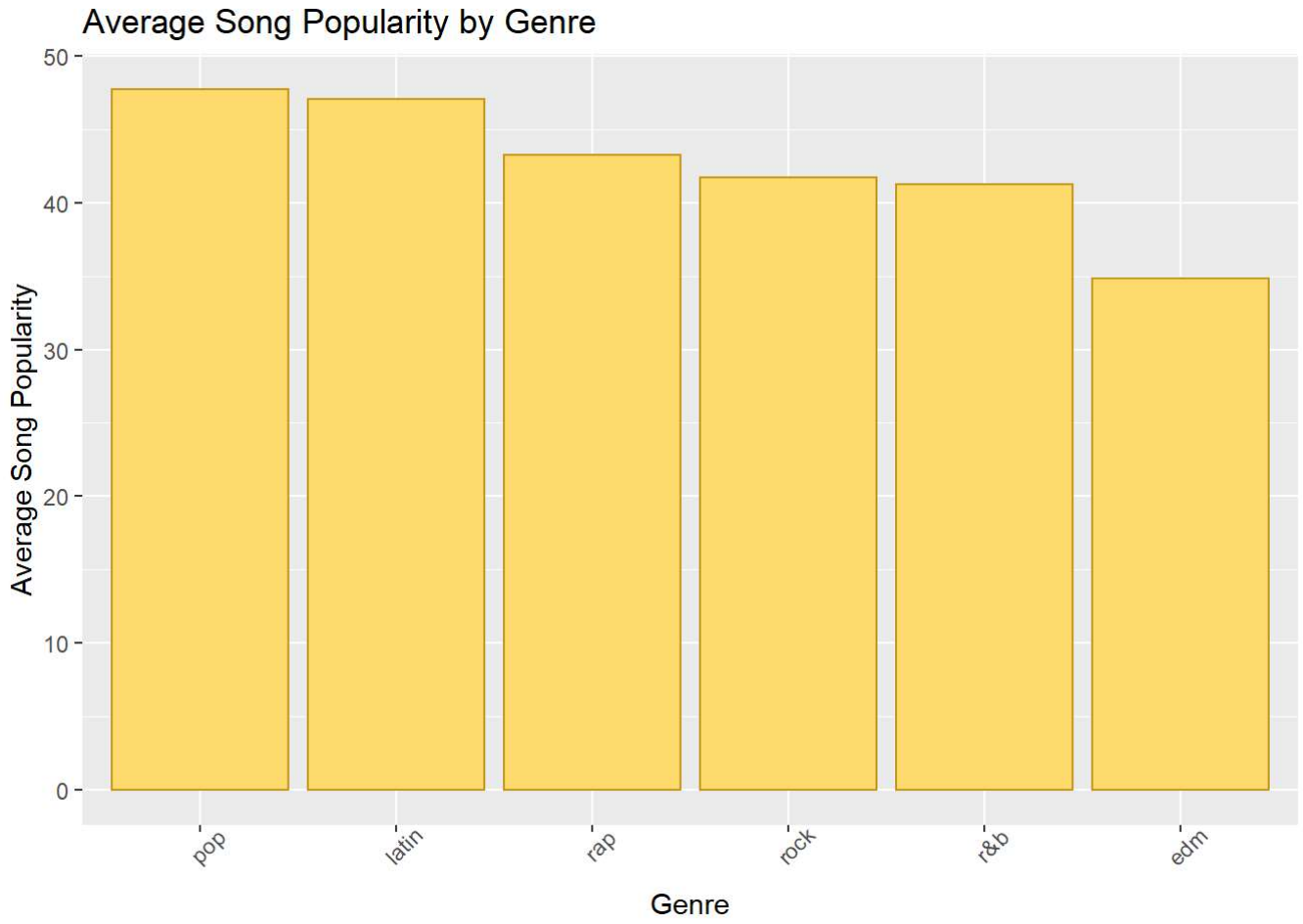
```
ggplot(data = spotify, aes(x = subgenre, fill = genre)) +
  geom_bar()+
  theme(axis.text.x = element_text(angle = 90))
```




```
spotify %>% group_by(subgenre) %>%
  summarize(average_popularity=mean(popularity)) %>%
  ggplot(aes(x=reorder(subgenre,-average_popularity), y=average_popularity))+
  geom_col(fill = "#F4EDCA", color = "#C4961A")+
  theme(axis.text.x = element_text(angle = 90))+
  ggtitle("Average Song Popularity by Subgenre")+
  labs(y="Average Song Popularity", x = "Subgenre")
```




```
spotify %>% group_by(genre) %>%
  summarize(average_popularity=mean(popularity)) %>%
  ggplot(aes(x=reorder(genre,-average_popularity), y=average_popularity))+
  geom_col(fill = "#FFDB6D", color = "#C4961A")+
  theme(axis.text.x = element_text(angle = 45))+
  ggtitle("Average Song Popularity by Genre")+
  labs(y="Average Song Popularity", x = "Genre")
```




```
# Compute the analysis of variance
res.aov <- aov(popularity ~ genre, data = spotify)
# Tukey's multiple comparison of means
TukeyHSD(res.aov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = popularity ~ genre, data = spotify)
##
## $genre
##                   diff        lwr        upr     p adj
## latin-edm  12.2113019 10.8822333 13.5403705 0.0000000
## pop-edm    12.9113438 11.6055353 14.2171522 0.0000000
## r&b-edm     6.3900052  5.0794252  7.7005852 0.0000000
## rap-edm     8.4045025  7.1128079  9.6961971 0.0000000
## rock-edm    6.8948113  5.5511883  8.2384343 0.0000000
## pop-latin   0.7000419 -0.6584696  2.0585534 0.6845195
## r&b-latin  -5.8212967 -7.1843952 -4.4581981 0.0000000
## rap-latin  -3.8067994 -5.1517501 -2.4618486 0.0000000
## rock-latin -5.3164905 -6.7113885 -3.9215925 0.0000000
## r&b-pop    -6.5213386 -7.8617677 -5.1809095 0.0000000
## rap-pop    -4.5068413 -5.8288114 -3.1848712 0.0000000
## rock-pop   -6.0165325 -7.3892862 -4.6437787 0.0000000
## rap-r&b     2.0144973  0.6878138  3.3411809 0.0002188
## rock-r&b    0.5048061 -0.8724873  1.8820995 0.9028530
## rock-rap   -1.5096912 -2.8690263 -0.1503561 0.0193464
```

```
spotify$popular_level<-rep(0, nrow(spotify))

spotify <- within(spotify, {

  popular_level[popularity>=66] <- "very_popular"
  popular_level[popularity>=33 & popularity< 66] <- "Moderate"
  popular_level[popularity <= 33] <- "not_popular"
} )
```

```
popular_table <- spotify %>%
  group_by(artist) %>%

  summarize(total_popular=sum(popular_level == "very_popular"),
            total_not_popular=sum(popular_level == "not_popular"),
            moderate = sum(popular_level == "Moderate"),
            popularity_ratio=ifelse(
              total_not_popular>0,total_popular/total_not_popular,total_popular)) %>%

  top_n(10,total_popular) %>%
  select(artist, total_popular,moderate,total_not_popular, popularity_ratio) %>%
  arrange(desc(total_popular))

popular_table
```

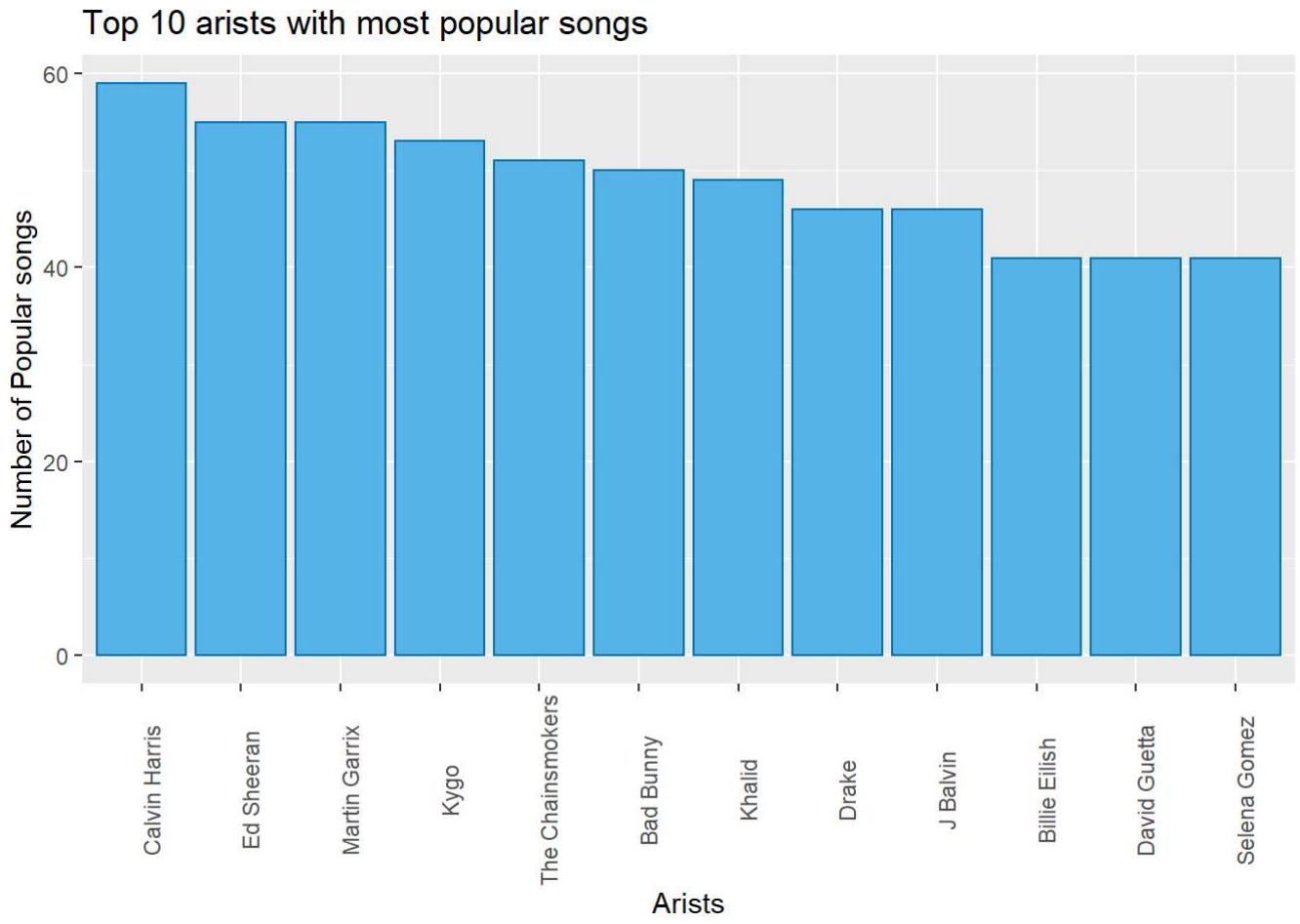
```
## # A tibble: 12 × 5
##    artist           total_popular moderate total_not_popular popularity_ratio
##    <chr>                    <int>    <int>             <int>            <dbl>
##  1 Calvin Harris               59       20                12             4.92
##  2 Ed Sheeran                  55       12                 2            27.5
##  3 Martin Garrix               55       63                43             1.28
##  4 Kygo                        53       23                 7             7.57
##  5 The Chainsmokers            51       57                15             3.4
##  6 Bad Bunny                   50        1                10             5
##  7 Khalid                      49        8                 0            49
##  8 Drake                       46       13                41             1.12
##  9 J Balvin                    46       10                16             2.88
## 10 Billie Eilish               41        0                 2            20.5
## 11 David Guetta                41       49                20             2.05
## 12 Selena Gomez                41        7                 4            10.2
```

```
knitr::kable(popular_table, align = "lccc", format="markdown",col.names = c('Artist', 'Popula
r','Moderate', ' Not popular', 'popularity ratio'), caption="Top 10 Artists by Number of Popular
Songs")
```

Top 10 Artists by Number of Popular Songs

Artist	Popular	Moderate	Not popular	popularity ratio
Calvin Harris	59	20	12	4.916667
Ed Sheeran	55	12	2	27.500000
Martin Garrix	55	63	43	1.279070
Kygo	53	23	7	7.571429
The Chainsmokers	51	57	15	3.400000
Bad Bunny	50	1	10	5.000000
Khalid	49	8	0	49.000000
Drake	46	13	41	1.121951
J Balvin	46	10	16	2.875000
Billie Eilish	41	0	2	20.500000
David Guetta	41	49	20	2.050000
Selena Gomez	41	7	4	10.250000

```
ggplot(popular_table,
       aes(x=reorder(artist, -total_popular), y=total_popular)) + geom_bar(stat = "identity",fil
l = "#56B4E9", color = "#0072B2")+
  theme(axis.text.x = element_text(angle = 90))+
  ggtitle("Top 10 arists with most popular songs")+
  labs(y="Number of Popular songs", x = "Arists")
```

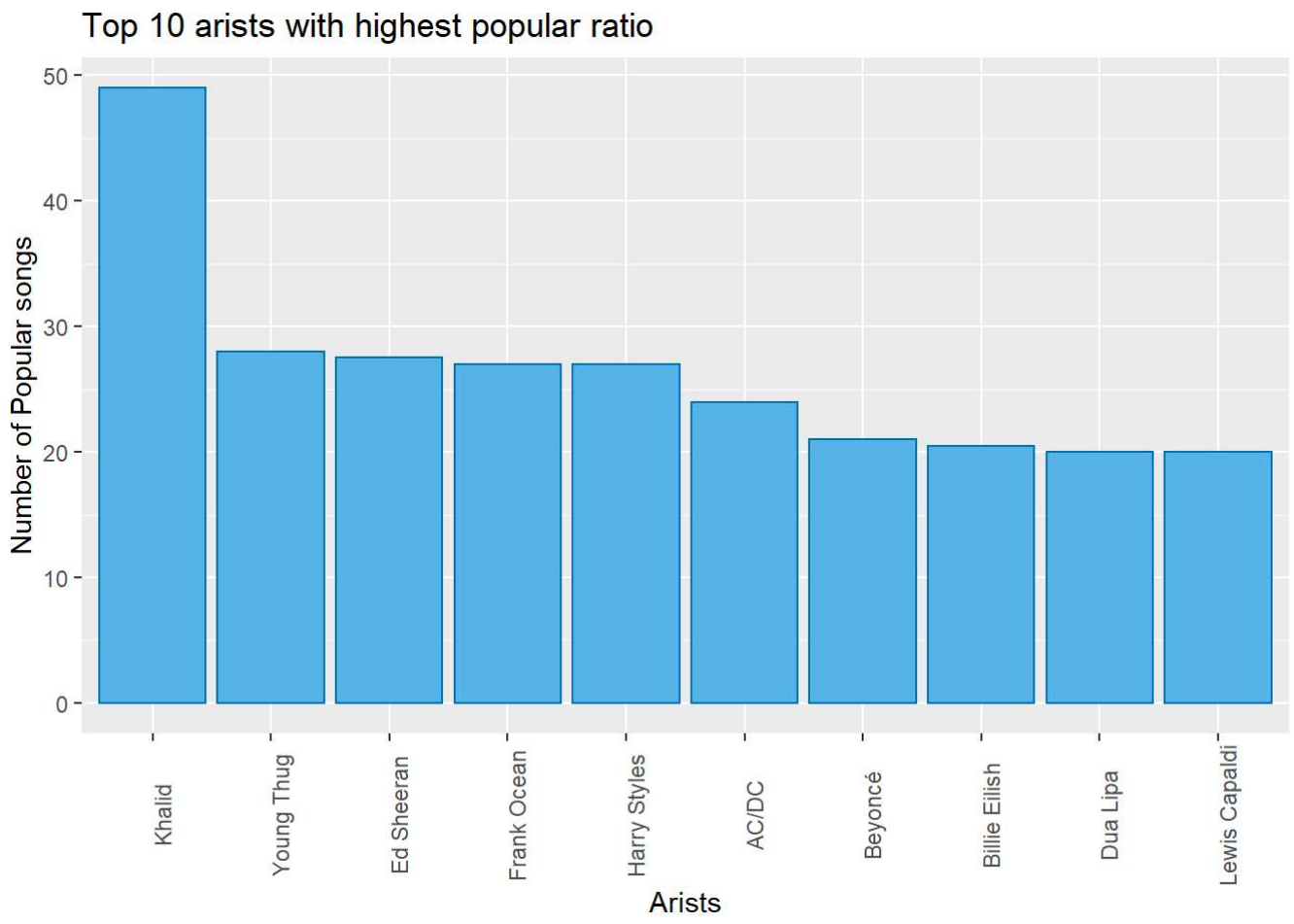



```
ratio_table <- spotify %>%
  group_by(artist) %>%

  summarize(total_popular=sum(popular_level == "very_popular"),
            total_not_popular=sum(popular_level == "not_popular"),
            moderate = sum(popular_level == "Moderate"),
            popularity_ratio=ifelse(
              total_not_popular>0,total_popular/total_not_popular,total_popular)) %>%

  top_n(10,popularity_ratio) %>%
  select(artist, total_popular,moderate,total_not_popular, popularity_ratio) %>%
  arrange(desc(popularity_ratio))


knitr::kable(ratio_table, align = "lccc", format="markdown",col.names = c('Artist', 'Popular','M
oderate', ' Not popular', 'Popular ratio'), caption="Top 10 Artists by Popular Ratio")
```

Top 10 Artists by Popular Ratio

Artist	Popular	Moderate	Not popular	Popular ratio
Khalid	49	8	0	49.0
Young Thug	28	15	1	28.0
Ed Sheeran	55	12	2	27.5
Frank Ocean	27	13	1	27.0
Harry Styles	27	0	0	27.0
AC/DC	24	4	0	24.0
Beyoncé	21	4	0	21.0
Billie Eilish	41	0	2	20.5
Dua Lipa	20	8	1	20.0
Lewis Capaldi	20	1	0	20.0

```
ggplot(ratio_table,
       aes(x=reorder(artist, -popularity_ratio), y=popularity_ratio)) + geom_bar(stat = "identit
y",fill = "#56B4E9", color = "#0072B2")+
  theme(axis.text.x = element_text(angle = 90))+
  ggtitle("Top 10 arists with highest popular ratio")+
  labs(y="Number of Popular songs", x = "Arists")
```




```
# parse out the keywords from the pipe-delimited string and determine keyword frequency
parse_key <- data.frame(table(unlist(strsplit(as.character(spotify$artist), split = "|",
                                              fixed = TRUE))))
# list the 20 most frequent keywords
head(parse_key[order(parse_key$Freq, decreasing = TRUE), ], 10)
```

```
##                          Var1 Freq
## 6165             Martin Garrix  161
## 7735                     Queen  136
## 9351          The Chainsmokers  123
## 2284              David Guetta  110
## 2634                  Don Omar  102
## 2683                     Drake  100
## 2487 Dimitri Vegas & Like Mike   93
## 1492             Calvin Harris   91
## 3873                  Hardwell   84
## 5290                      Kygo   83
```

```
# Packages
library(tidyverse)
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(ggplot2)
library(boot)
```

Write your Null and Alternative Hypothesis.

```
df <- read.csv("spotify_cleaned.csv",stringsAsFactors=FALSE)
head(df)
```

```
##   X                                                 name           artist
## 1 1 I Don't Care (with Justin Bieber) - Loud Luxury Remix       Ed Sheeran
## 2 2                       Memories - Dillon Francis Remix         Maroon 5
## 3 3                       All the Time - Don Diablo Remix     Zara Larsson
## 4 4                     Call You Mine - Keanu Silva Remix The Chainsmokers
## 5 5                Someone You Loved - Future Humans Remix    Lewis Capaldi
## 6 6     Beautiful People (feat. Khalid) - Jack Wins Remix       Ed Sheeran
##   popularity year genre  subgenre danceability energy key loudness mode
## 1         66 2019   pop dance pop        0.748  0.916   6   -2.634    1
## 2         67 2019   pop dance pop        0.726  0.815  11   -4.969    1
## 3         70 2019   pop dance pop        0.675  0.931   1   -3.432    0
## 4         60 2019   pop dance pop        0.718  0.930   7   -3.778    1
## 5         69 2019   pop dance pop        0.650  0.833   1   -4.672    1
## 6         67 2019   pop dance pop        0.675  0.919   8   -5.385    1
##   speechiness acousticness instrumantalness liveness valence   tempo duration
## 1      0.0583       0.1020         0.00e+00   0.0653   0.518 122.036   194754
## 2      0.0373       0.0724         4.21e-03   0.3570   0.693  99.972   162600
## 3      0.0742       0.0794         2.33e-05   0.1100   0.613 124.008   176616
## 4      0.1020       0.0287         9.43e-06   0.2040   0.277 121.956   169093
## 5      0.0359       0.0803         0.00e+00   0.0833   0.725 123.976   189052
## 6      0.1270       0.0799         0.00e+00   0.1430   0.585 124.982   163049
```

```
# My Data Science Question:
# Is the population mean of popularity of rock music greater than that of rap music?


# Hypothesis:
# H0: The mean popularity of rock music is the same as the mean popularity of rap music
# Ha: The mean popularity of rock music is higher than the mean popularity of rap music
```

Do a t-test (if your Data science question is about population averages. If your question is about comparing proportions then use a Z-test), and write your conclusion at 5% significance level.

```
# Since my question is to compare average, therefore, I use t-test for my hypothesis:

dt_pop <- df$popularity[df$genre == "rock"]


ur_pop <- df$popularity[df$genre == "rap"]




# Since the hypothesis is to compare energy of KC and AG,  alternative is higher, use greater in
t-test.
t.test(ur_pop, dt_pop, alt="greater",conf.level = 0.95)  #at 5% significance level.
```

```
##
##  Welch Two Sample t-test
##
## data:  ur_pop and dt_pop
## t = 3.2267, df = 10232, p-value = 0.0006281
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.7400385       Inf
## sample estimates:
## mean of x mean of y
##  43.23803  41.72834
```

```
# Conclusion:
# Based on the t-test, we can see that the p-value is less than 0.05 which is the 5% significanc
e level.
# Therefore, we reject the Null hypothesis at this significant level.
# We can conclude that we have enough evidence to prove that the mean energy of Kelly Clarkson i
s not the same as the mean energy of Ariana Grande. In fact, we can say that the mean energy of
Kelly Clarkson is larger than the mean energy of Ariana Grande

# this conclusion also aligns with my EDA analysis, that the mean energy of Kelly Clarkson is la
rger than the mean energy of Ariana Grande
```

Do a bootstrap test(here you will be using bootstrap sampling and 95% bootstrap percentile interval) to answer the same question and write your conclusion at 5% significance level.

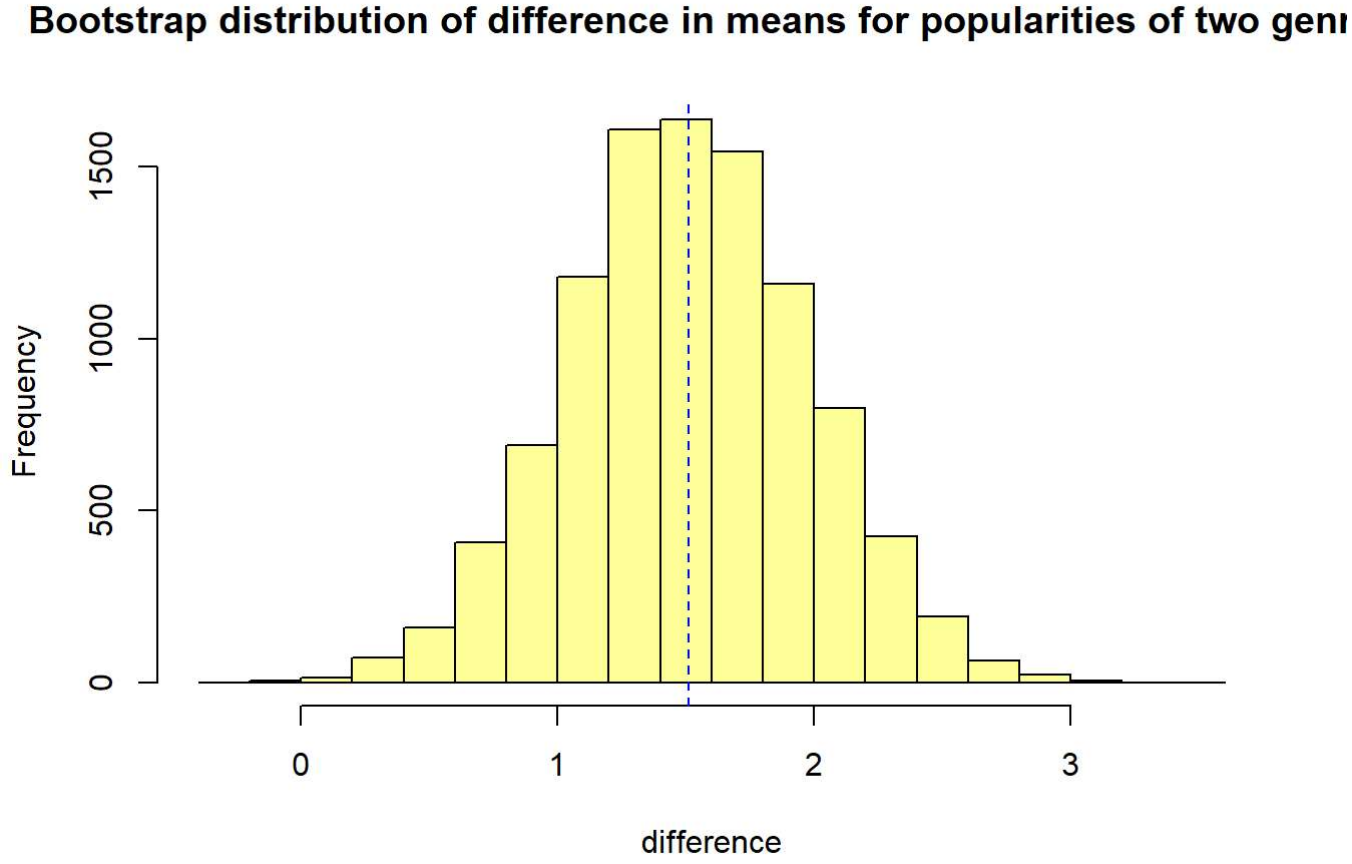
```
# Use bootstrap

set.seed(2)
difference <- rep(NA,10000)
boot_ratio <- rep(NA, 10000)

for (j in 1:10000){
  boot_dt <- mean(sample(dt_pop, length(dt_pop), replace = T))
  boot_ur <- mean(sample(ur_pop, length(ur_pop), replace = T))
  difference[j] <- boot_ur - boot_dt   #the difference
  boot_ratio[j] <- boot_ur / boot_dt # The ratio
}

mean(difference) #bootstrap mean difference
```

```
## [1] 1.509232
```

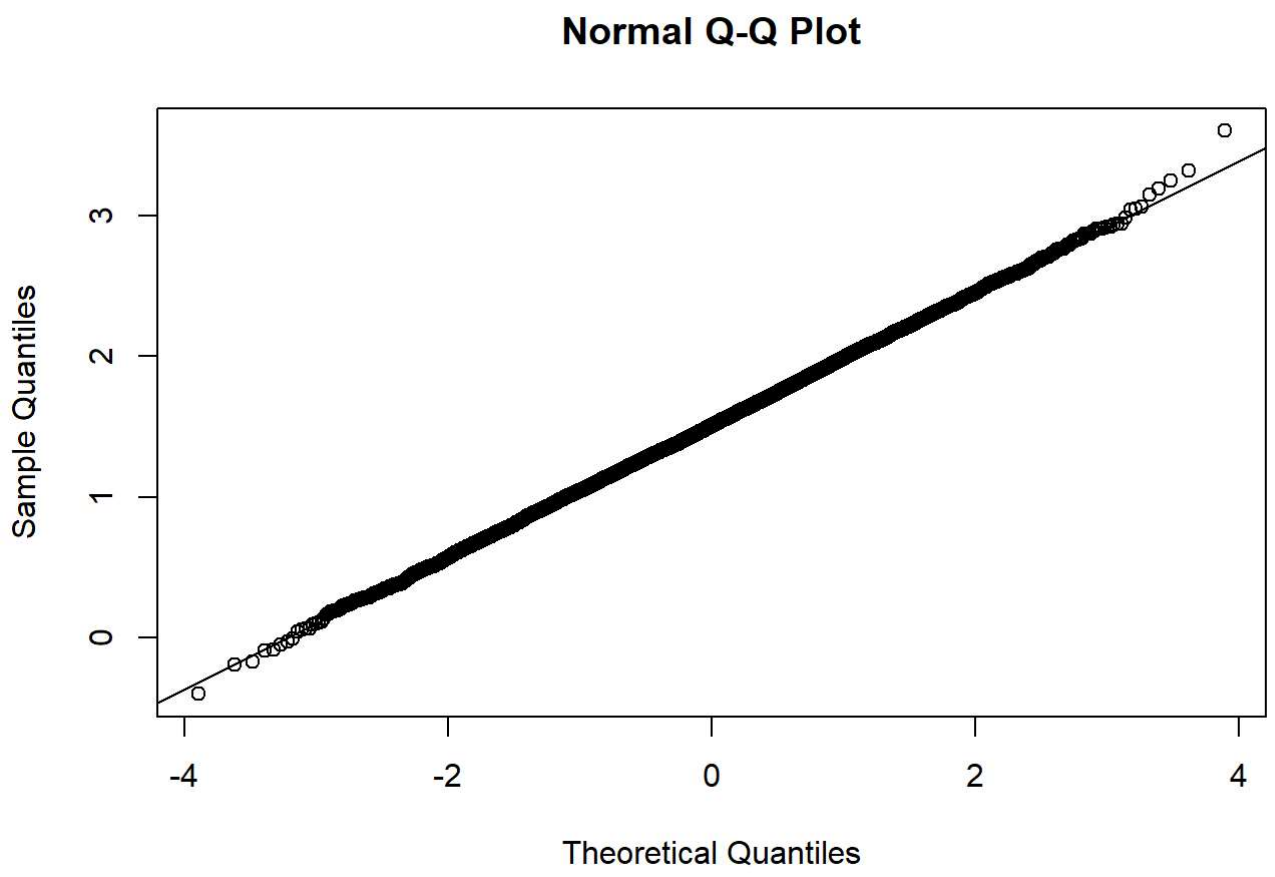
```
mean(boot_ratio) #bootstrap mean rAtio
```

```
## [1] 1.036242
```

```
hist(difference, main = "Bootstrap distribution of difference in means for popularities of two g
enres",col = '#FFFF99')
abline(v = mean(ur_pop) - mean(dt_pop), col = "blue", lty = 2)
```



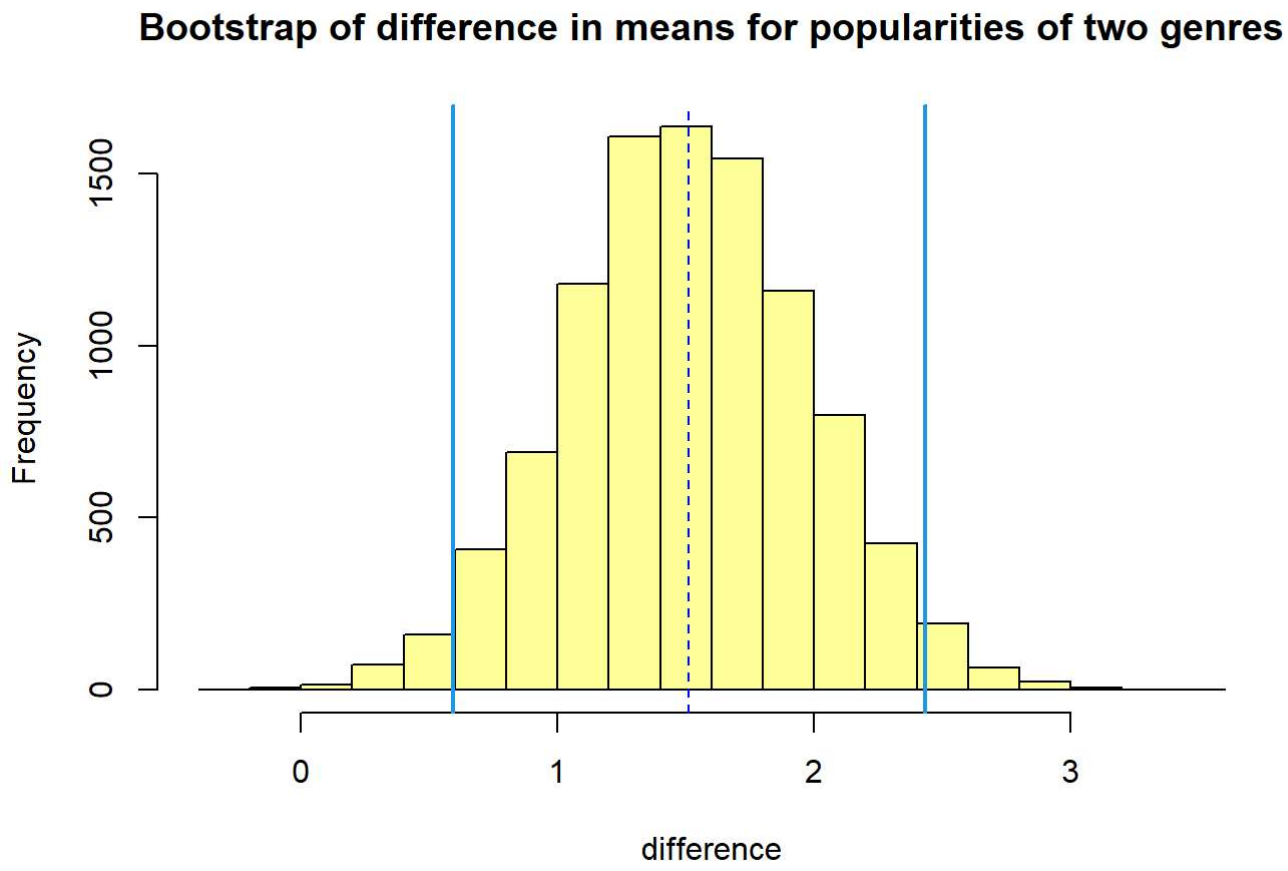

```
qqnorm(difference)
qqline(difference)
```




```
# 95% CI
CI <- quantile(difference, c(0.025, 0.975))
CI
```

```
##      2.5%     97.5%
## 0.5916207 2.4320700
```

```
hist(difference, main = "Bootstrap of difference in means for popularities of two genres",col =
'#FFFF99')
abline(v = mean(ur_pop) - mean(dt_pop), col = "blue", lty = 2)
abline(v = CI, col = 4, lwd = 2)
```



Import the cleaned dataset

```
df <- read.csv("spotify_cleaned.csv")
head(df,5)
```

```
##   X                                                 name           artist
## 1 1 I Don't Care (with Justin Bieber) - Loud Luxury Remix       Ed Sheeran
## 2 2                       Memories - Dillon Francis Remix         Maroon 5
## 3 3                       All the Time - Don Diablo Remix     Zara Larsson
## 4 4                    Call You Mine - Keanu Silva Remix The Chainsmokers
## 5 5              Someone You Loved - Future Humans Remix    Lewis Capaldi
##   popularity year genre  subgenre danceability energy key loudness mode
## 1         66 2019   pop dance pop        0.748  0.916   6   -2.634    1
## 2         67 2019   pop dance pop        0.726  0.815  11   -4.969    1
## 3         70 2019   pop dance pop        0.675  0.931   1   -3.432    0
## 4         60 2019   pop dance pop        0.718  0.930   7   -3.778    1
## 5         69 2019   pop dance pop        0.650  0.833   1   -4.672    1
##   speechiness acousticness instrumantalness liveness valence   tempo duration
## 1      0.0583       0.1020         0.00e+00   0.0653   0.518 122.036   194754
## 2      0.0373       0.0724         4.21e-03   0.3570   0.693  99.972   162600
## 3      0.0742       0.0794         2.33e-05   0.1100   0.613 124.008   176616
## 4      0.1020       0.0287         9.43e-06   0.2040   0.277 121.956   169093
## 5      0.0359       0.0803         0.00e+00   0.0833   0.725 123.976   189052
```

From the daily observations, some music genres do dominate the front rank of the listener's music charts, such as pop music. There are many reasons for this phenomenon, such as the fact that these pop music are sung by favorite singers. Hence, with this complex relationship, whether the popularity of music is related to the genre of music itself needs to be studied. In the previous hypothesis test, the relationship between the average populative of rock music and of rap music has been addressed. However, the general relationship between the genres and the popularity cannot be explained by a test between two specific genres of musics.

In the second hypothsis test, whether the popularity is related to the genre will be examinated. Before implementing the test method, categorizing the popularity into three different groups - high,medium,low - will be helpful in understanding the relationship. Based on the 0 to 100 popularity score rules on spotify, three categories can be presented as: - High: popularity >= 66 - medium: 33 > popularity > 66 - low: popularity <= 33

The two-way table is generated to show the frequency of songs in the group of genre and popularity.

```
df$popularity <- as.factor(ifelse(df$popularity>=66, 'High',
                    ifelse(df$popularity<66 & df$popularity>33, 'Medium','Low')))
```

```
library(knitr)
t1 = table(df$popularity,df$genre)
kable(t1,align = "lccrr")
```

	edm	latin	pop	r&b	rap	rock
High	613	1378	1603	1096	921	1000
Low	2733	1343	1533	2043	1654	1748
Medium	2697	2432	2371	2292	3168	2203

In statistics, Chi-square test is commonly used in testing the independence of two variables. If two variables are independent, it means there is no relationship between two factors. Based on the question of whether the popularity is related to the genre, the null hypothesis (H0) can be set as the popularity and the genre is independent, and then correspondingly, the alternative hypothesis (Ha) will be the popularity and the genre is dependent.In this case, the specific hypotheses are:

- H0: There is no relationship between the popularity and the genre of musics.
- Ha: There is relationship between the popularity and the genre of musics.

```
t2 <- chisq.test(t1)
t2
```

```
##
##  Pearson's Chi-squared test
##
## data:  t1
## X-squared = 1270.6, df = 10, p-value < 2.2e-16
```

```
chisq.test(t1)$expected
```

```
##
##               edm    latin      pop      r&b      rap      rock
##   High   1216.957 1037.726 1109.016 1093.711 1156.542  997.0471
##   Low    2034.828 1735.143 1854.343 1828.752 1933.810 1667.1242
##   Medium 2791.215 2380.131 2543.641 2508.537 2652.647 2286.8287
```

From the output, the p-value is less than the significance level of 5%, which means the rejection of null hypothesis. In this context, rejecting the null hypothesis for the Chi-square test of independence means there is a significant relationship between the popularity and the genre of musics.

Load the required dataset

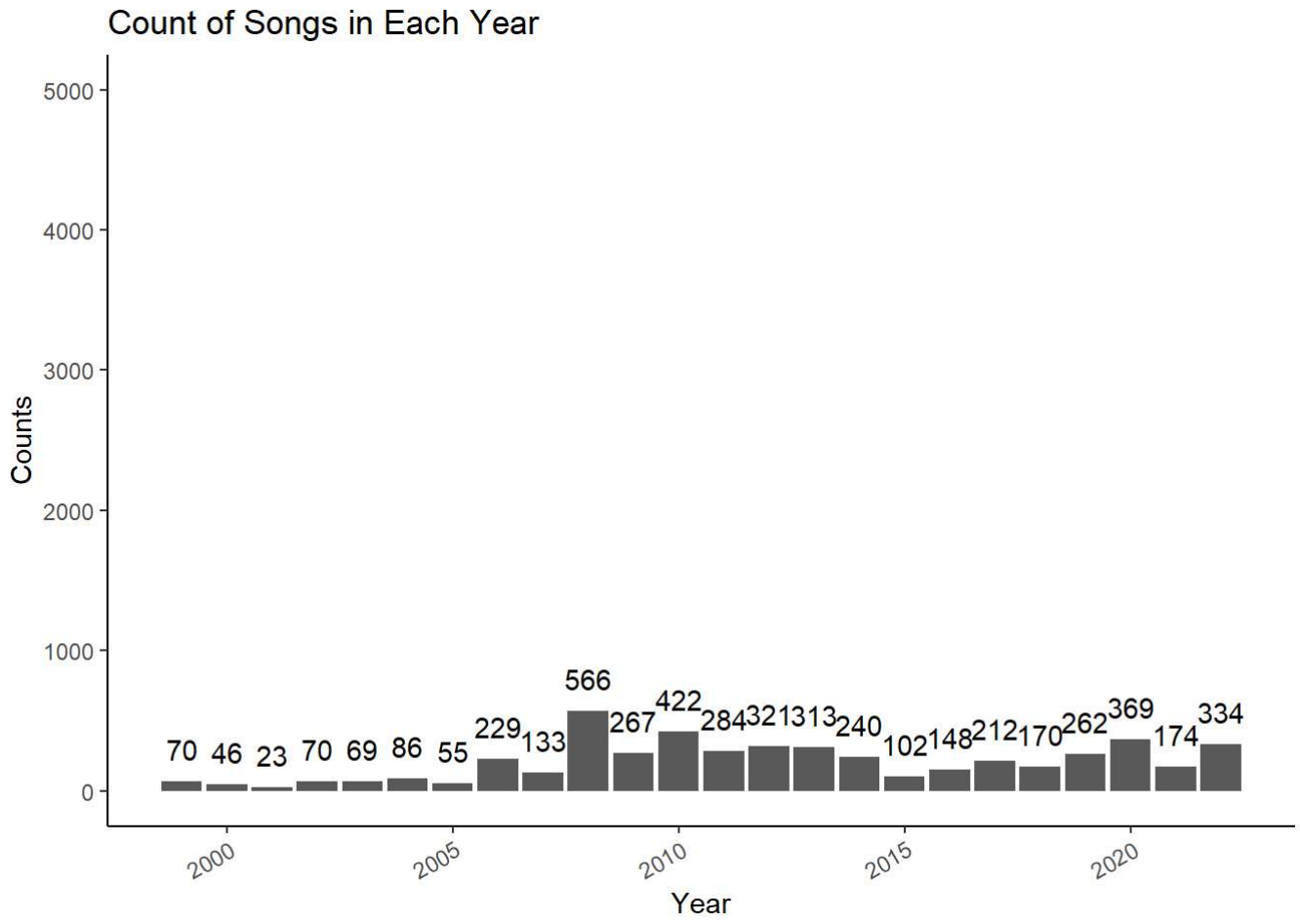
```
artists <- read.csv("./Tracks_Artists.csv")
head(artists)
```

```
##   X  artist_name Valence danceability energy loudness speechiness acousticness
## 1 1 Taylor Swift  0.0984        0.735  0.444  -10.519      0.0684       0.2040
## 2 2 Taylor Swift  0.0382        0.658  0.378   -8.300      0.0379       0.0593
## 3 3 Taylor Swift  0.5190        0.638  0.634   -6.582      0.0457       0.1330
## 4 4 Taylor Swift  0.1540        0.659  0.323  -13.425      0.0436       0.7350
## 5 5 Taylor Swift  0.3760        0.694  0.380  -10.307      0.0614       0.4160
## 6 6 Taylor Swift  0.2300        0.636  0.377  -11.721      0.0708       0.7100
##   liveness   tempo                            track_name
## 1   0.1700  97.038                         Lavender Haze
## 2   0.0976 108.034                                Maroon
## 3   0.1520  96.953                           Anti-Hero
## 4   0.1160 110.007 Snow On The Beach (feat. Lana Del Rey)
## 5   0.1260 120.044               You're On Your Own, Kid
## 6   0.1150 139.966                         Midnight Rain
##               album_name album_release_year
## 1 Midnights (3am Edition)              2022
## 2 Midnights (3am Edition)              2022
## 3 Midnights (3am Edition)              2022
## 4 Midnights (3am Edition)              2022
## 5 Midnights (3am Edition)              2022
## 6 Midnights (3am Edition)              2022
```

```
summary(artists$album_release_year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1999    2008    2012    2013    2018    2022
```

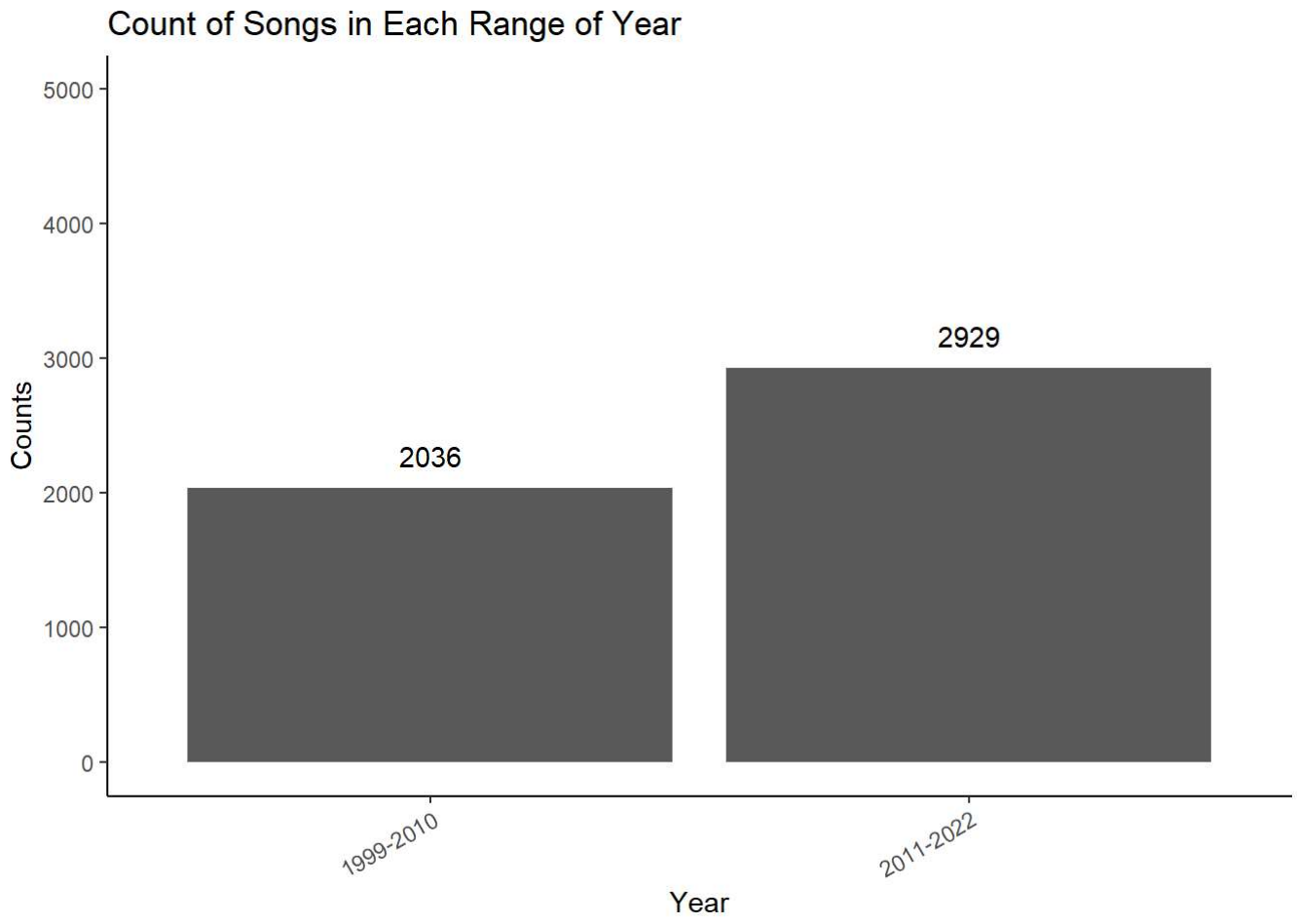
```
library(ggplot2)
ggplot(artists, aes(x=album_release_year)) +
  geom_bar()+
  geom_text(stat='count', aes(label=..count..), vjust=-1)+
  ylim(0,5000)+
  theme_classic()+
  theme(axis.text.x=element_text(angle=30,hjust=1))+
  labs(title="Count of Songs in Each Year",
       x ="Year", y = "Counts")
```




```
artists$year_range <- ifelse(artists$album_release_year < 2011, "1999-2010","2011-2022")
head(artists)
```

```
##   X  artist_name Valence danceability energy loudness speechiness acousticness
## 1 1 Taylor Swift  0.0984        0.735  0.444  -10.519      0.0684       0.2040
## 2 2 Taylor Swift  0.0382        0.658  0.378   -8.300      0.0379       0.0593
## 3 3 Taylor Swift  0.5190        0.638  0.634   -6.582      0.0457       0.1330
## 4 4 Taylor Swift  0.1540        0.659  0.323  -13.425      0.0436       0.7350
## 5 5 Taylor Swift  0.3760        0.694  0.380  -10.307      0.0614       0.4160
## 6 6 Taylor Swift  0.2300        0.636  0.377  -11.721      0.0708       0.7100
##   liveness   tempo                              track_name
## 1   0.1700  97.038                           Lavender Haze
## 2   0.0976 108.034                                  Maroon
## 3   0.1520  96.953                             Anti-Hero
## 4   0.1160 110.007 Snow On The Beach (feat. Lana Del Rey)
## 5   0.1260 120.044                  You're On Your Own, Kid
## 6   0.1150 139.966                           Midnight Rain
##                album_name album_release_year year_range
## 1 Midnights (3am Edition)               2022  2011-2022
## 2 Midnights (3am Edition)               2022  2011-2022
## 3 Midnights (3am Edition)               2022  2011-2022
## 4 Midnights (3am Edition)               2022  2011-2022
## 5 Midnights (3am Edition)               2022  2011-2022
## 6 Midnights (3am Edition)               2022  2011-2022
```

```
library(ggplot2)
ggplot(artists, aes(x=year_range)) +
  geom_bar()+
  geom_text(stat='count', aes(label=..count..), vjust=-1)+
  ylim(0,5000)+
  theme_classic()+
  theme(axis.text.x=element_text(angle=30,hjust=1))+
  labs(title="Count of Songs in Each Range of Year",
       x ="Year", y = "Counts")
```



Hypothesis 2:

Null Hypothesis: I will make null hypothesis as the average Valence of songs in 1999-2010 is higher than songs in 2011-2011.

Alternative Hypothesis: the average Valence of songs in 1999-2010 is lower than songs in 2011-2011.

```
# group the dataset by Yearrange
Year2010 <- subset(artists, artists$year_range == "1999-2010",select = c("Valence"))
Year2022 <- subset(artists, artists$year_range == "2011-2022",select = c("Valence") )
# t.test
t.test(Year2010,Year2022,alternative = "less")
```

```
##
##  Welch Two Sample t-test
##
## data:  Year2010 and Year2022
## t = 9.9977, df = 4084.5, p-value = 1
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##        -Inf 0.07579864
## sample estimates:
## mean of x mean of y
## 0.4736412 0.4085535
```

```
set.seed(1)
n1 = length(Year2010)
n2 = length(Year2022)
N <- 10000
diff_mean <- numeric(N)

for (i in 1:N)
{
Year2010.sample <- sample(Year2010$Valence, n1, replace = TRUE)
Year2022.sample <- sample(Year2022$Valence, n2, replace = TRUE)
diff_mean[i] <- mean(Year2010.sample) - mean(Year2022.sample)
}

mean(diff_mean)
```

```
## [1] 0.06635376
```

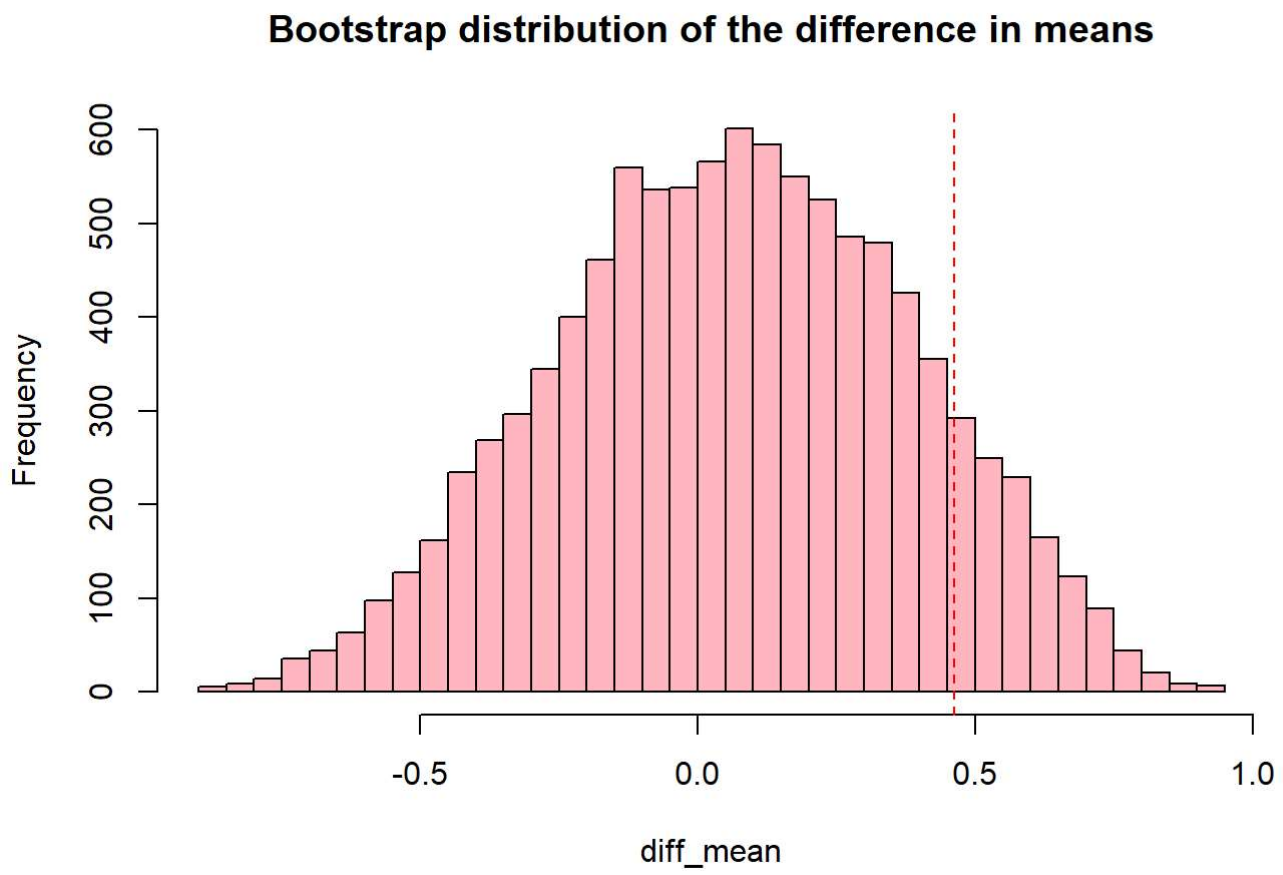
```
quantile(diff_mean, c(.025, .975))
```

```
##   2.5%  97.5%
## -0.557  0.661
```

```
mydiff = function(mydf){
  index1 = artists$year_range == "1999-2010"
  index2 = artists$year_range == "2011-2022"
  return(mean(artists$Valence[index1]) - mean(artists$Valence[index2]))
}

mydiff(genre.clean) #actual mean difference from the original sample
```

```
## [1] 0.06508774
```

```
hist(diff_mean,breaks=50,main = "Bootstrap distribution of the difference in means",col = 'light
pink')
abline(v = mean(Year2010.sample) - mean(Year2022.sample), col = "red", lty = 2)
```




```
data <-  read.csv("spotify_cleaned.csv")
```

Data Science Questions: Are the variables contributing for predicting “popularity” of the songs is same for different genres?

Create a new variable named “Valence_C”.

```
data$Valence_C <- rep(0,nrow(data))

data1 <- within(data, {
  Valence_C[valence>=0.8 & valence<=1] <- "more positive"
  Valence_C[valence>=0.5 & valence<0.8] <- "moderate"
  Valence_C[valence<=0.499] <- "more negative"
})
head(data1)
```

```
##   X                                                 name           artist
## 1 1 I Don't Care (with Justin Bieber) - Loud Luxury Remix       Ed Sheeran
## 2 2                       Memories - Dillon Francis Remix         Maroon 5
## 3 3                       All the Time - Don Diablo Remix     Zara Larsson
## 4 4                     Call You Mine - Keanu Silva Remix The Chainsmokers
## 5 5                Someone You Loved - Future Humans Remix    Lewis Capaldi
## 6 6     Beautiful People (feat. Khalid) - Jack Wins Remix       Ed Sheeran
##   popularity year genre  subgenre danceability energy key loudness mode
## 1         66 2019   pop dance pop        0.748  0.916   6   -2.634    1
## 2         67 2019   pop dance pop        0.726  0.815  11   -4.969    1
## 3         70 2019   pop dance pop        0.675  0.931   1   -3.432    0
## 4         60 2019   pop dance pop        0.718  0.930   7   -3.778    1
## 5         69 2019   pop dance pop        0.650  0.833   1   -4.672    1
## 6         67 2019   pop dance pop        0.675  0.919   8   -5.385    1
##   speechiness acousticness instrumantalness liveness valence   tempo duration
## 1      0.0583       0.1020         0.00e+00   0.0653   0.518 122.036   194754
## 2      0.0373       0.0724         4.21e-03   0.3570   0.693  99.972   162600
## 3      0.0742       0.0794         2.33e-05   0.1100   0.613 124.008   176616
## 4      0.1020       0.0287         9.43e-06   0.2040   0.277 121.956   169093
## 5      0.0359       0.0803         0.00e+00   0.0833   0.725 123.976   189052
## 6      0.1270       0.0799         0.00e+00   0.1430   0.585 124.982   163049
##       Valence_C
## 1      moderate
## 2      moderate
## 3      moderate
## 4 more negative
## 5      moderate
## 6      moderate
```

Fit multiple linear regression models separately for different genres.

```
set.seed(12)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(tidyverse)
```

```
pop <- data1[data1$genre=="pop",]
edm <- data1[data1$genre=="edm",]
```

```
names(pop)
```

```
##  [1] "X"                "name"             "artist"           "popularity"
##  [5] "year"             "genre"            "subgenre"         "danceability"
##  [9] "energy"           "key"              "loudness"         "mode"
## [13] "speechiness"      "acousticness"     "instrumantalness" "liveness"
## [17] "valence"          "tempo"            "duration"         "Valence_C"
```

```
training_samples <- pop$popularity %>%
  createDataPartition(p=0.8, list=FALSE)

train <- pop[training_samples, ]
test <- pop[-training_samples, ]
dim(train)
```

```
## [1] 4407   20
```

Fit the FULL linear regression model.

```
fit1 <- lm(popularity ~ danceability + energy + loudness + speechiness + acousticness + instruma
ntalness + liveness + valence + tempo +Valence_C, data = train)
summary(fit1)
```

```
##
## Call:
## lm(formula = popularity ~ danceability + energy + loudness +
##     speechiness + acousticness + instrumantalness + liveness +
##     valence + tempo + Valence_C, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -64.194 -17.056   4.545  19.186  58.333
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             74.966782   4.959693  15.115  < 2e-16 ***
## danceability            14.311075   3.227621   4.434 9.48e-06 ***
## energy                 -33.654892   3.451728  -9.750  < 2e-16 ***
## loudness                 2.548691   0.198625  12.832  < 2e-16 ***
## speechiness             22.378111   5.546528   4.035 5.56e-05 ***
## acousticness             1.132331   2.091348   0.541   0.5882
## instrumantalness       -10.358720   2.168184  -4.778 1.83e-06 ***
## liveness                 0.372393   2.730148   0.136   0.8915
## valence                  4.192508   3.821088   1.097   0.2726
## tempo                    0.002425   0.015471   0.157   0.8754
## Valence_Cmore negative   0.246728   1.379914   0.179   0.8581
## Valence_Cmore positive  -4.052253   1.548440  -2.617   0.0089 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.18 on 4395 degrees of freedom
## Multiple R-squared:  0.07447,    Adjusted R-squared:  0.07215
## F-statistic: 32.15 on 11 and 4395 DF,  p-value: < 2.2e-16
```

Remove insignificant variables.

```
fit2 <- lm(popularity ~ danceability + energy + loudness + speechiness + instrumantalness + Vale
nce_C, data = train)
summary(fit2)
```

```
##
## Call:
## lm(formula = popularity ~ danceability + energy + loudness +
##     speechiness + instrumantalness + Valence_C, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -64.044 -17.094   4.524  19.181  58.725
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              77.8923     3.9773  19.584  < 2e-16 ***
## danceability             14.7807     3.0557   4.837 1.36e-06 ***
## energy                  -33.8176     3.0221 -11.190  < 2e-16 ***
## loudness                  2.5462     0.1982  12.844  < 2e-16 ***
## speechiness              23.0326     5.4536   4.223 2.46e-05 ***
## instrumantalness        -10.5435     2.1567  -4.889 1.05e-06 ***
## Valence_Cmore negative   -0.9738     0.8297  -1.174   0.2406
## Valence_Cmore positive   -3.0531     1.2655  -2.413   0.0159 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.18 on 4399 degrees of freedom
## Multiple R-squared:  0.07414,    Adjusted R-squared:  0.07267
## F-statistic: 50.32 on 7 and 4399 DF,  p-value: < 2.2e-16
```

Check interactions.

```
fit12 <- lm(popularity ~ (danceability+energy+loudness+speechiness+instrumantalness)^2, data=tra
in)
summary(fit12)
```

```
##
## Call:
## lm(formula = popularity ~ (danceability + energy + loudness +
##     speechiness + instrumantalness)^2, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.687 -16.864   4.522  19.050  53.836
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   103.1230    14.9440   6.901 5.92e-12 ***
## danceability                   -5.9375    22.2770  -0.267 0.789844
## energy                        -72.3426    14.8384  -4.875 1.12e-06 ***
## loudness                        3.8407     0.9789   3.924 8.85e-05 ***
## speechiness                    28.1324    60.8885   0.462 0.644082
## instrumantalness              -70.0682    18.7063  -3.746 0.000182 ***
## danceability:energy            41.5464    21.7212   1.913 0.055850 .
## danceability:loudness           0.3138     1.4425   0.218 0.827802
## danceability:speechiness      -74.1727    42.3132  -1.753 0.079681 .
## danceability:instrumantalness  -1.8488    14.8859  -0.124 0.901163
## energy:loudness                -1.2025     0.6406  -1.877 0.060553 .
## energy:speechiness             21.8262    47.0904   0.463 0.643032
## energy:instrumantalness        29.7824    13.9461   2.136 0.032772 *
## loudness:speechiness           -4.4678     3.3171  -1.347 0.178077
## loudness:instrumantalness      -4.5224     0.8919  -5.071 4.12e-07 ***
## speechiness:instrumantalness   57.2961    62.7741   0.913 0.361432
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.09 on 4391 degrees of freedom
## Multiple R-squared:  0.08224,    Adjusted R-squared:  0.0791
## F-statistic: 26.23 on 15 and 4391 DF,  p-value: < 2.2e-16
```

```
fit3 <- lm(popularity~danceability+energy+loudness+speechiness+instrumantalness+energy*loudness+
loudness*instrumantalness,data=train)
summary(fit3)
```

```
##
## Call:
## lm(formula = popularity ~ danceability + energy + loudness +
##     speechiness + instrumantalness + energy * loudness + loudness *
##     instrumantalness, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.679 -16.902   4.628  19.004  51.384
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                85.3323     4.8193  17.706  < 2e-16 ***
## danceability               13.6163     2.8745   4.737 2.24e-06 ***
## energy                    -42.2346     5.2307  -8.074 8.68e-16 ***
## loudness                    3.5558     0.4316   8.239 2.26e-16 ***
## speechiness                23.9101     5.4301   4.403 1.09e-05 ***
## instrumantalness          -36.5057     5.5841  -6.537 6.97e-11 ***
## energy:loudness            -1.1351     0.6160  -1.843   0.0654 .
## loudness:instrumantalness  -3.1077     0.6226  -4.991 6.22e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.12 on 4399 degrees of freedom
## Multiple R-squared:  0.07821,    Adjusted R-squared:  0.07674
## F-statistic: 53.32 on 7 and 4399 DF,  p-value: < 2.2e-16
```

Make Predictions

```
pred1 <- fit1 %>% predict(test)
p1 = data.frame(
  RMSE=RMSE(pred1,test$popularity),
  R2=R2(pred1,test$popularity)
)

pred2 <- fit2 %>% predict(test)
p2 <- data.frame(
  RMSE=RMSE(pred2,test$popularity),
  R2=R2(pred2,test$popularity)
)

pred3 <- fit3 %>% predict(test)
p3 <- data.frame(
  RMSE=RMSE(pred3,test$popularity),
  R2=R2(pred3,test$popularity)
)
```

```
summary(fit1)$fstatistic[1]
```

```
##   value
## 32.1482
```

```
summary(fit1)$adj.r.squared
```

```
## [1] 0.07215349
```

```
summary(fit1)$sigma #RSE
```

```
## [1] 24.18283
```

```
all=rbind(p1,p2,p3)
all=cbind(all,c(summary(fit1)$fstatistic[1],summary(fit2)$fstatistic[1],summary(fit3)$fstatistic
[1]))
all=cbind(all,c(summary(fit1)$adj.r.squared,summary(fit2)$adj.r.squared,summary(fit3)$adj.r.squa
red))
all=cbind(all,c(summary(fit1)$sigma,summary(fit2)$sigma,summary(fit3)$sigma))

all=cbind(all,c("fit1","fit2","fit3"))
colnames(all)[c(3,4,5,6)]<-c("F stat","Adj R 2","RSE","models")
all
```

```
##       RMSE         R2   F stat    Adj R 2      RSE models
## 1 24.56187 0.06293309 32.14820 0.07215349 24.18283   fit1
## 2 24.56240 0.06287795 50.32470 0.07266956 24.17610   fit2
## 3 24.48017 0.06916223 53.31969 0.07674341 24.12294   fit3
```

It turns out that fit3 is the best model.

Next we check the predictors for genres “EDM” and compared with “Pop”.

```
training_samples <- edm$popularity %>%
  createDataPartition(p=0.8,list = FALSE)

train <- edm[training_samples,]
test <- edm[-training_samples,]
dim(train)
```

```
## [1] 4836   20
```

```
names(train)
```

```
##  [1] "X"                "name"             "artist"           "popularity"
##  [5] "year"             "genre"            "subgenre"         "danceability"
##  [9] "energy"           "key"              "loudness"         "mode"
## [13] "speechiness"      "acousticness"     "instrumantalness" "liveness"
## [17] "valence"          "tempo"            "duration"         "Valence_C"
```

Fit FULL linear regression model for EDM.

```
fit11 <- lm(popularity ~ danceability+energy+loudness+speechiness+acousticness+instrumantalness+
liveness+valence+tempo+Valence_C, data = train)
summary(fit11)
```

```
##
## Call:
## lm(formula = popularity ~ danceability + energy + loudness +
##     speechiness + acousticness + instrumantalness + liveness +
##     valence + tempo + Valence_C, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.222 -17.139   1.452  16.119  60.599
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              69.80804    5.19775  13.430  < 2e-16 ***
## danceability             -1.61857    2.98997  -0.541   0.5883
## energy                  -22.54855    3.39266  -6.646 3.34e-11 ***
## loudness                  1.05649    0.19034   5.551 3.00e-08 ***
## speechiness              -5.85242    4.54543  -1.288   0.1980
## acousticness             18.75716    2.42862   7.723 1.37e-14 ***
## instrumantalness        -11.67195    1.11720 -10.448  < 2e-16 ***
## liveness                 -0.50596    1.91175  -0.265   0.7913
## valence                   3.42872    2.88262   1.189   0.2343
## tempo                    -0.06207    0.02144  -2.895   0.0038 **
## Valence_Cmore negative   -3.01397    1.23370  -2.443   0.0146 *
## Valence_Cmore positive   -0.09883    1.62645  -0.061   0.9515
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22 on 4824 degrees of freedom
## Multiple R-squared:  0.09797,    Adjusted R-squared:  0.09592
## F-statistic: 47.63 on 11 and 4824 DF,  p-value: < 2.2e-16
```

Remove insignificant variables.

```
fit22 <- lm(popularity ~ energy+loudness+acousticness+instrumantalness+tempo+Valence_C,data=trai
n)
summary(fit22)
```

```
##
## Call:
## lm(formula = popularity ~ energy + loudness + acousticness +
##     instrumantalness + tempo + Valence_C, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -59.994 -17.040   1.451  16.197  60.350
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              70.63965    4.14229  17.053  < 2e-16 ***
## energy                  -22.92205    3.34518  -6.852 8.18e-12 ***
## loudness                  1.08253    0.18969   5.707 1.22e-08 ***
## acousticness             18.71702    2.40287   7.789 8.18e-15 ***
## instrumantalness        -11.83083    1.07050 -11.052  < 2e-16 ***
## tempo                    -0.06221    0.02095  -2.969  0.00301 **
## Valence_Cmore negative   -4.03112    0.72655  -5.548 3.04e-08 ***
## Valence_Cmore positive    0.72013    1.45498   0.495  0.62066
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22 on 4828 degrees of freedom
## Multiple R-squared:  0.09733,    Adjusted R-squared:  0.09602
## F-statistic: 74.37 on 7 and 4828 DF,  p-value: < 2.2e-16
```

Check interactions.

```
fit12 <- lm(popularity~(energy+loudness+acousticness+instrumantalness+tempo)^2,data = train)
summary(fit12)
```

```
##
## Call:
## lm(formula = popularity ~ (energy + loudness + acousticness +
##     instrumantalness + tempo)^2, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -61.181 -17.354   1.618  15.903  61.269
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   174.99957   29.15749   6.002 2.09e-09 ***
## energy                       -126.28142   28.83189  -4.380 1.21e-05 ***
## loudness                        5.00799    1.74073   2.877 0.004033 **
## acousticness                  -15.48766   22.37814  -0.692 0.488915
## instrumantalness              -36.98138   14.61259  -2.531 0.011412 *
## tempo                          -0.89089    0.22725  -3.920 8.97e-05 ***
## energy:loudness                 0.02925    0.87817   0.033 0.973426
## energy:acousticness            -4.79823   18.10678  -0.265 0.791023
## energy:instrumantalness        26.52900    9.93878   2.669 0.007628 **
## energy:tempo                    0.79113    0.22253   3.555 0.000381 ***
## loudness:acousticness           1.82139    1.16484   1.564 0.117969
## loudness:instrumantalness      -1.08406    0.51276  -2.114 0.034553 *
## loudness:tempo                 -0.03137    0.01284  -2.443 0.014613 *
## acousticness:instrumantalness   5.23843    9.01513   0.581 0.561220
## acousticness:tempo              0.41017    0.11723   3.499 0.000472 ***
## instrumantalness:tempo         -0.02880    0.08831  -0.326 0.744372
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22.01 on 4820 degrees of freedom
## Multiple R-squared:  0.09745,    Adjusted R-squared:  0.09464
## F-statistic:  34.7 on 15 and 4820 DF,  p-value: < 2.2e-16
```

```
fit33 <- lm(popularity~energy+loudness+acousticness+instrumantalness+tempo+energy*instrumantalne
ss+energy*tempo+loudness*acousticness+loudness*instrumantalness+loudness*tempo+acousticness*temp
o,data = train)
summary(fit33)
```

```
##
## Call:
## lm(formula = popularity ~ energy + loudness + acousticness +
##     instrumantalness + tempo + energy * instrumantalness + energy *
##     tempo + loudness * acousticness + loudness * instrumantalness +
##     loudness * tempo + acousticness * tempo, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.852 -17.348   1.649  15.852  61.511
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                173.93172   27.04614   6.431 1.39e-10 ***
## energy                    -124.98282   26.31185  -4.750 2.09e-06 ***
## loudness                     4.94943    1.52289   3.250 0.001162 **
## acousticness               -20.87972   15.32116  -1.363 0.173008
## instrumantalness           -39.70270    9.87629  -4.020 5.91e-05 ***
## tempo                       -0.87870    0.21524  -4.082 4.53e-05 ***
## energy:instrumantalness     25.38726    9.21056   2.756 0.005868 **
## energy:tempo                 0.77807    0.20934   3.717 0.000204 ***
## loudness:acousticness        1.40103    0.64173   2.183 0.029070 *
## loudness:instrumantalness   -1.13734    0.50221  -2.265 0.023578 *
## loudness:tempo              -0.03033    0.01197  -2.535 0.011286 *
## acousticness:tempo           0.41191    0.11624   3.544 0.000399 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 22 on 4824 degrees of freedom
## Multiple R-squared:  0.09737,    Adjusted R-squared:  0.09531
## F-statistic: 47.31 on 11 and 4824 DF,  p-value: < 2.2e-16
```

Make predictions

```
pred11 <- fit11 %>% predict(test)
p11=data.frame(
  RMSE=RMSE(pred11,test$popularity),
  R2=R2(pred11,test$popularity)
)

pred22 <- fit22 %>% predict(test)
p22=data.frame(
  RMSE=RMSE(pred22,test$popularity),
  R2=R2(pred22,test$popularity)
)

pred33 <- fit33 %>% predict(test)
p33=data.frame(
  RMSE=RMSE(pred33,test$popularity),
  R2=R2(pred33,test$popularity)
)
```

```
all2=rbind(p11,p22,p33)
all2=cbind(all2,c(summary(fit11)$fstatistic[1],summary(fit22)$fstatistic[1],summary(fit33)$fstat
istic[1]))
all2=cbind(all2,c(summary(fit11)$adj.r.squared,summary(fit22)$adj.r.squared,summary(fit33)$adj.
r.squared))
all2=cbind(all2,c(summary(fit11)$sigma,summary(fit22)$sigma,summary(fit33)$sigma))

all2=cbind(all2,c("fit11","fit22","fit33"))
colnames(all2)[c(3,4,5,6)] <- c("F stat","Adj R 2","RSE","models")
all2
```

```
##       RMSE         R2   F stat    Adj R 2      RSE models
## 1 22.32606 0.07767128 47.63290 0.09591728 21.99751  fit11
## 2 22.32488 0.07775583 74.37116 0.09602481 21.99620  fit22
## 3 22.26614 0.08191485 47.30527 0.09530761 22.00492  fit33
```

It turns out that fit33j is the best model.