DATA 2010 Term project

Mai Nguyen & Tuan Ngo

5/22/2022

Topic

Our team chose the *Top100 Billboard dataset* with Billboard data and Audio data.

```
billboard = readr::read_csv(
  'https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2021/2021-09-14/billboard.c
audio = readr::read_csv(
  'https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2021/2021-09-14/audio_featu
```

Introduction: Dataset Exploratory Analysis

For the Billboard data, there are 10 variables: - url

- week_id
- week_position
- song
- performer
- song_id
- instance (number of time the song appears on the chart: max = 10)
- previous_week_position
- peak_position
- weeks_on_chart (max = 87)

For the Audio data, there are 22 variables: - song_id

- performer
- song
- spotify_genre
- spotify_track_id
- spotify_track_preview_url

- spotify_track_duration_ms (min = 29688 ms = 0.49 minutes and max = 3079157 ms = 51.32 minutes)
- spotify_track_explicit
- spotify_track_album
- danceability (range: 0.0 to 1.0)
- energy
- key
- loudness
- mode
- speechiness
- acousticness
- instrumentalness
- liveness
- valence
- tempo
- time_signature
- spotify_track_popularity

There are some variables that appear in both datasets so we decided to combine them together to create 1 dataset using 'left_join' from the 'tidyverse' package.

We ranked the songs by the number of weeks on chart of every songs and we found that most of the top 10 are from after 2010.

song	week_id
Radioactive	5/10/2014
Sail	3/22/2014
I'm Yours	10/10/2009
Blinding Lights	5/29/2021
How Do I Live	10/10/1998
Counting Stars	10/18/2014
Party Rock Anthem	7/21/2012
Foolish Games/You Were Meant For Me	2/21/1998
Rolling In The Deep	4/14/2012
Before He Cheats	12/1/2007

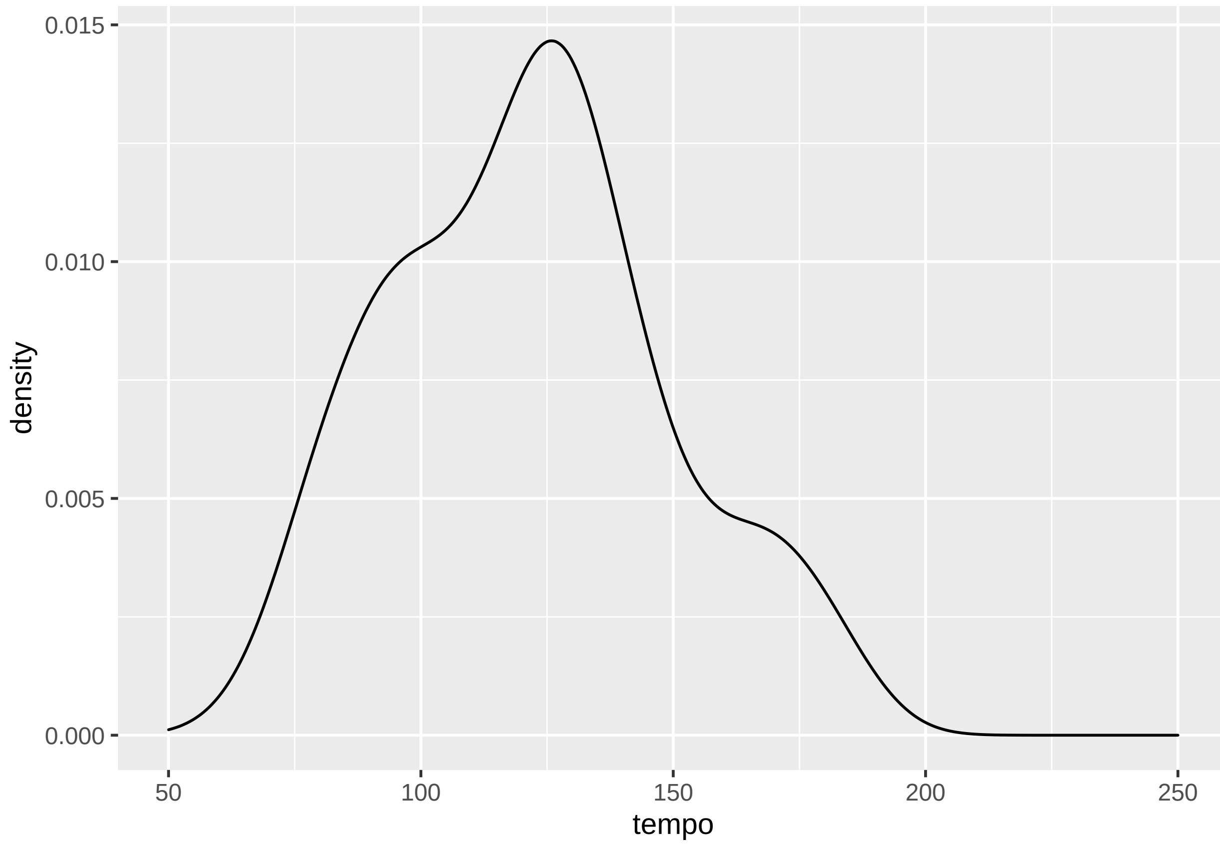
Therefore, if we want find out the latest trend, we will focus on analyzing the chart from 2010 up until now only. We used 'lubridate' package to change the format of the date of week_id and filtered out the songs with Week_id starting from 01-01-2010 up until today. This new dataset is called song_after_2010.

We also use visualization (use ggplot and ggplot2) to see the characteristics of song that had topped the chart (having peak position equals 1).

We observed the following features of top 1 songs on the chart:

- Most songs have tempo around 80 to 136 and the range of tempo is from 66 to 186

```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 66.00   99.98  122.02  121.84  136.05  186.00 
```



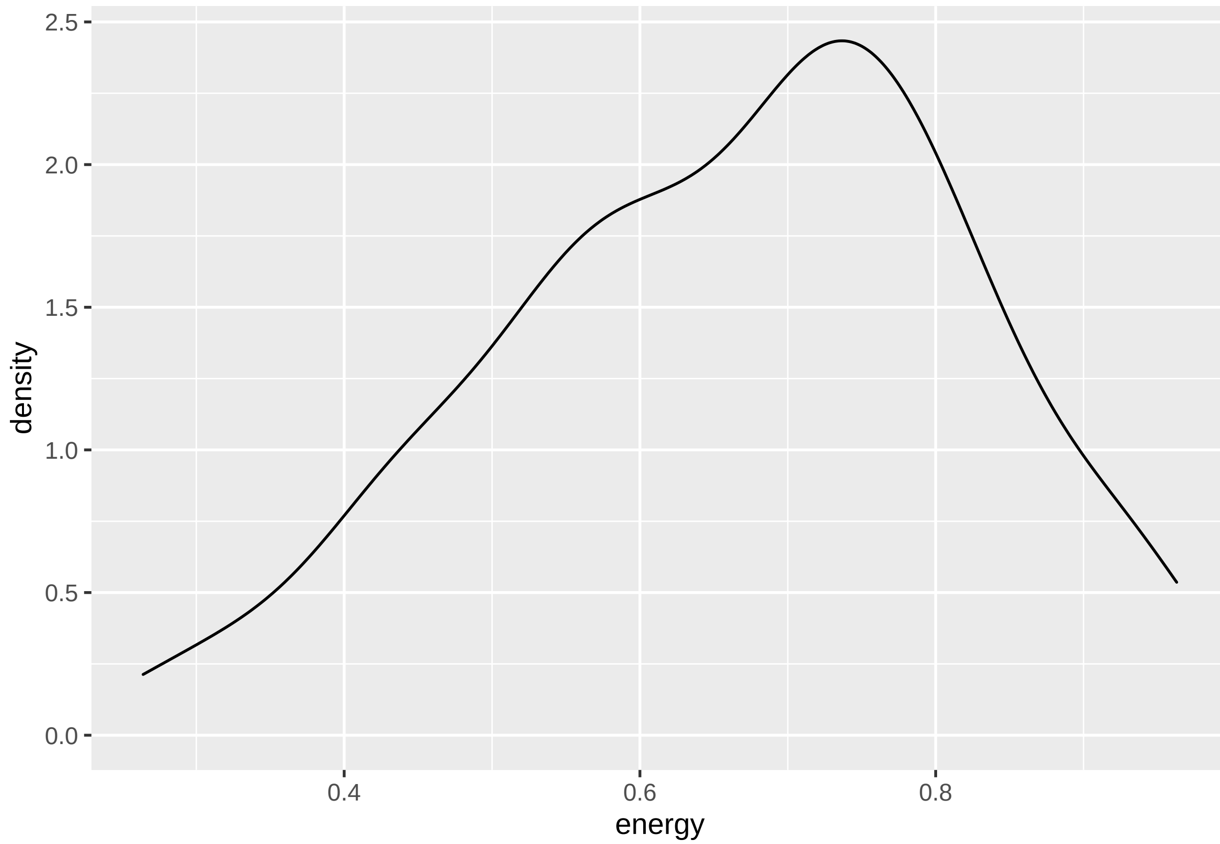
- Songs with genre of pop and hip hop

```
Selecting by n
```

spotify_genre	n
[‘dance pop’, ‘pop’, ‘post-teen pop’]	19
[‘barbadian pop’, ‘dance pop’, ‘pop’, ‘post-teen pop’, ‘r&b’, ‘urban contemporary’]	7
[‘pop’, ‘post-teen pop’]	7
[‘pop’]	7
[‘canadian hip hop’, ‘canadian pop’, ‘hip hop’, ‘pop rap’, ‘rap’, ‘toronto rap’]	6

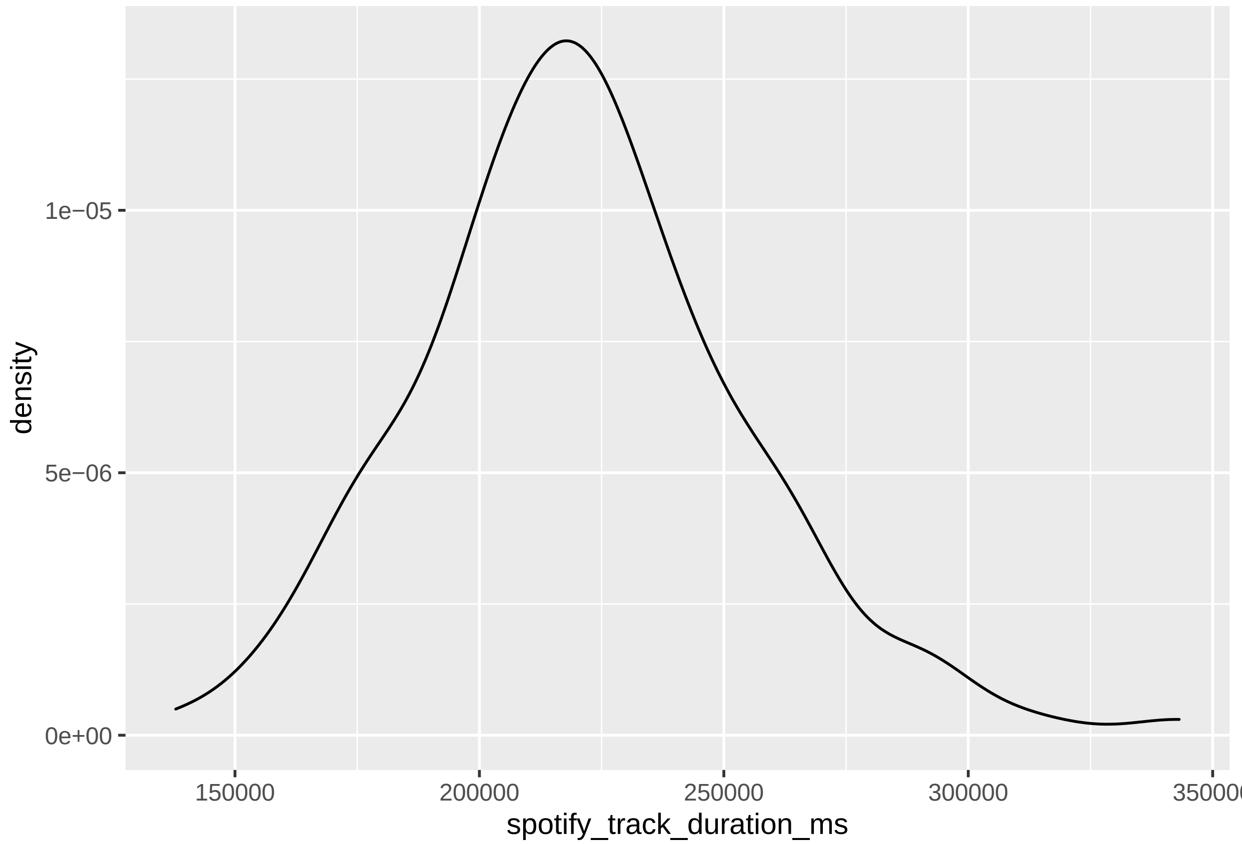
- Songs with energy from 0.52 to 0.83 (not too low or too high)

```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.2640  0.5580  0.6930  0.6638  0.7720  0.9630 
```



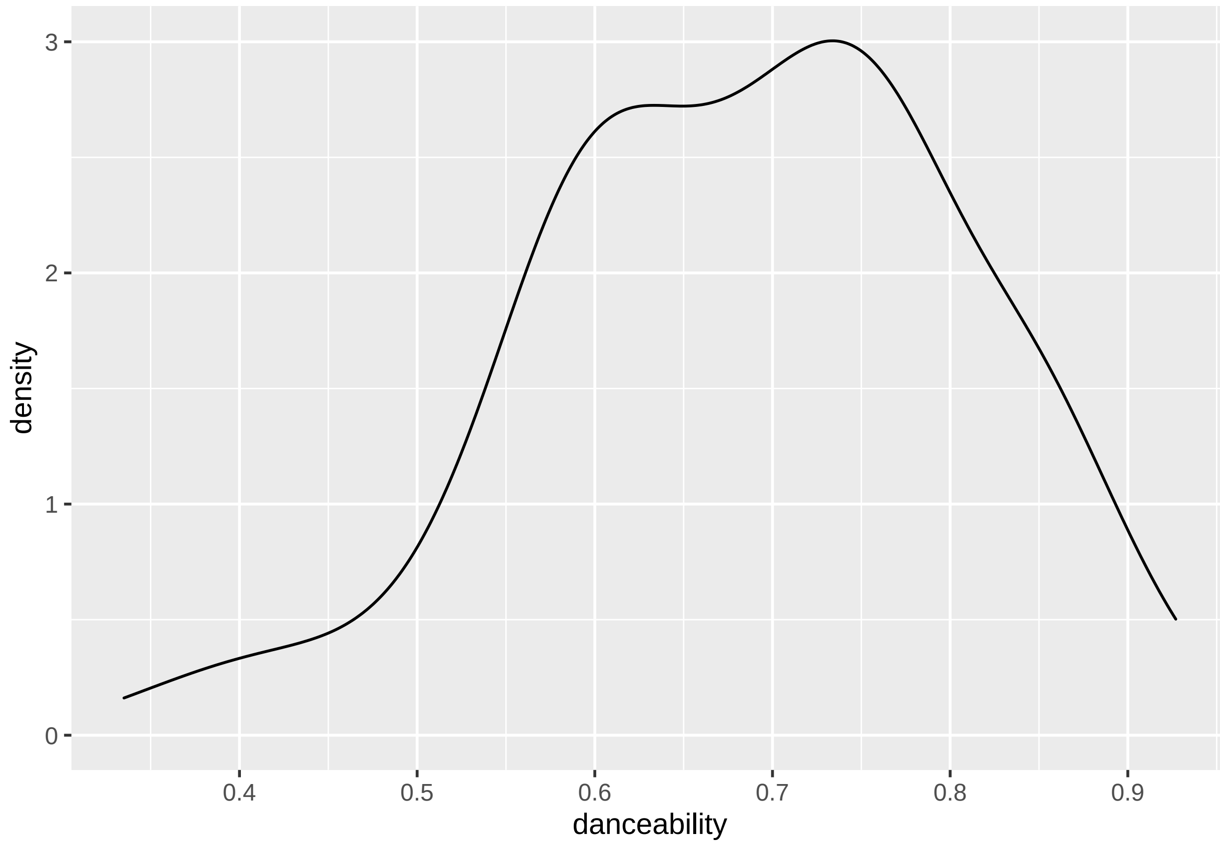
- Songs with duration from 200000ms (3 min) to 250000ms (4 min)

```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
137875  200080  219200  221112  241106  343150 
```



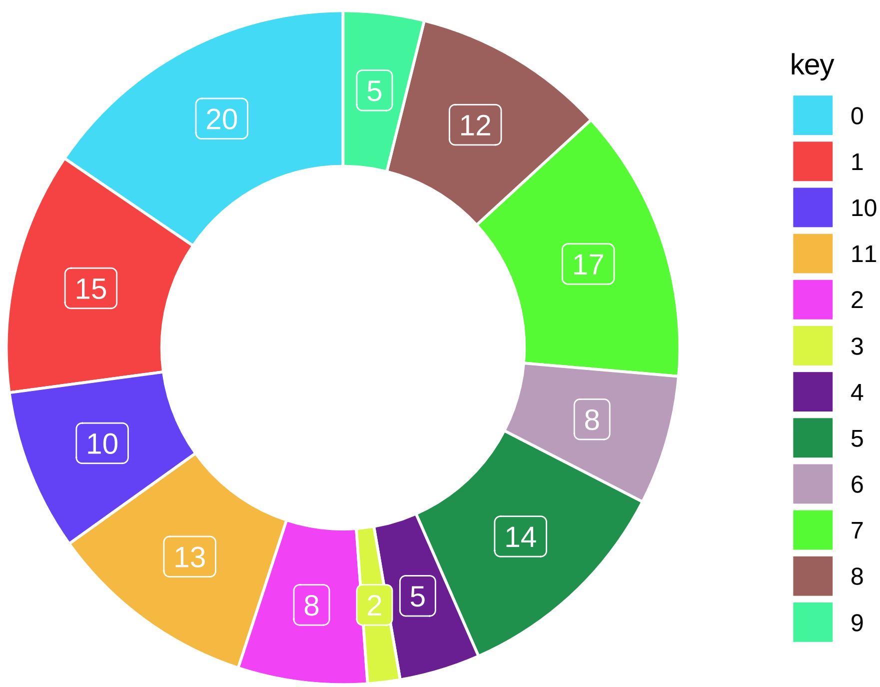
- Songs with average danceability (from 0.6 to 0.8)

```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.3350  0.6070  0.6970  0.6875  0.7780  0.9270
```



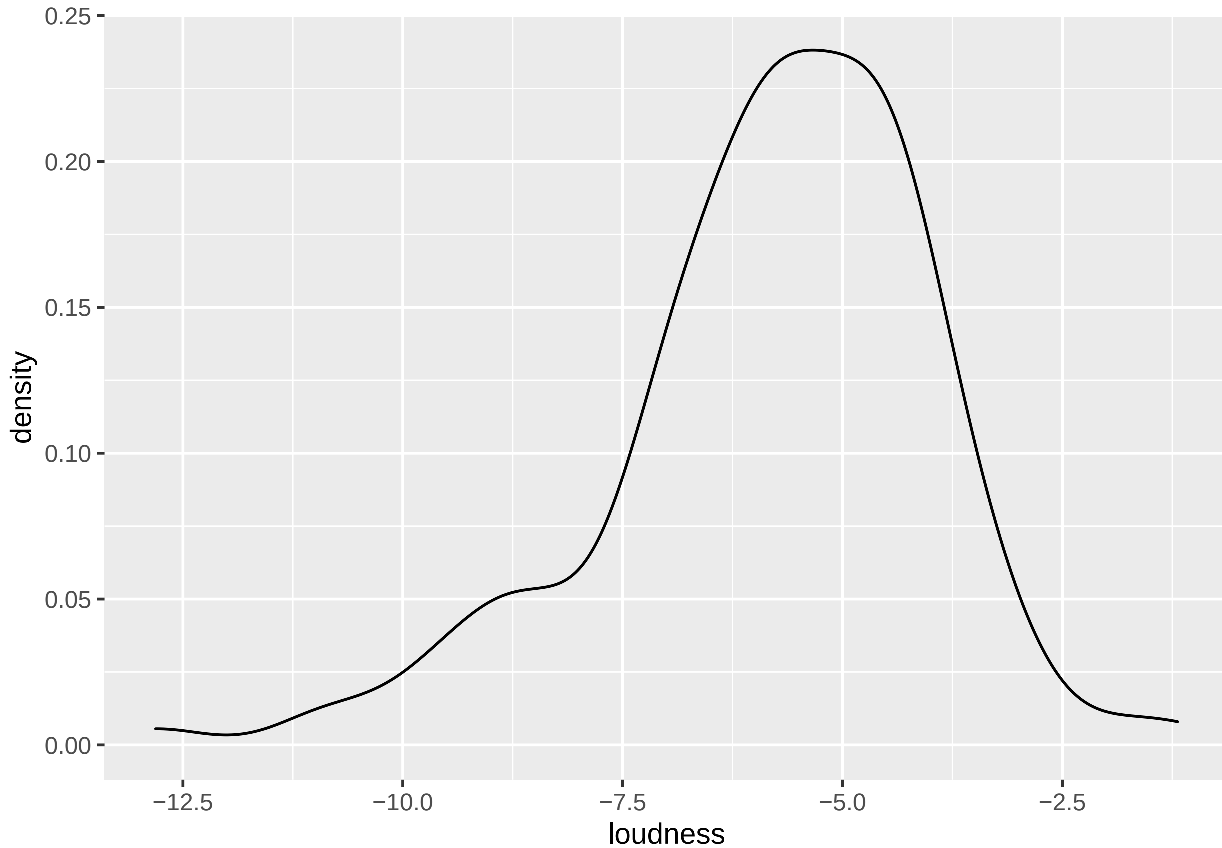
- Most popular key is C (key = 0) and least popular is D-sharp or E-minor (key = 3)

key	count
0	20
7	17
1	15
5	14
11	13
8	12
10	10
2	8
6	8
4	5
9	5
3	2

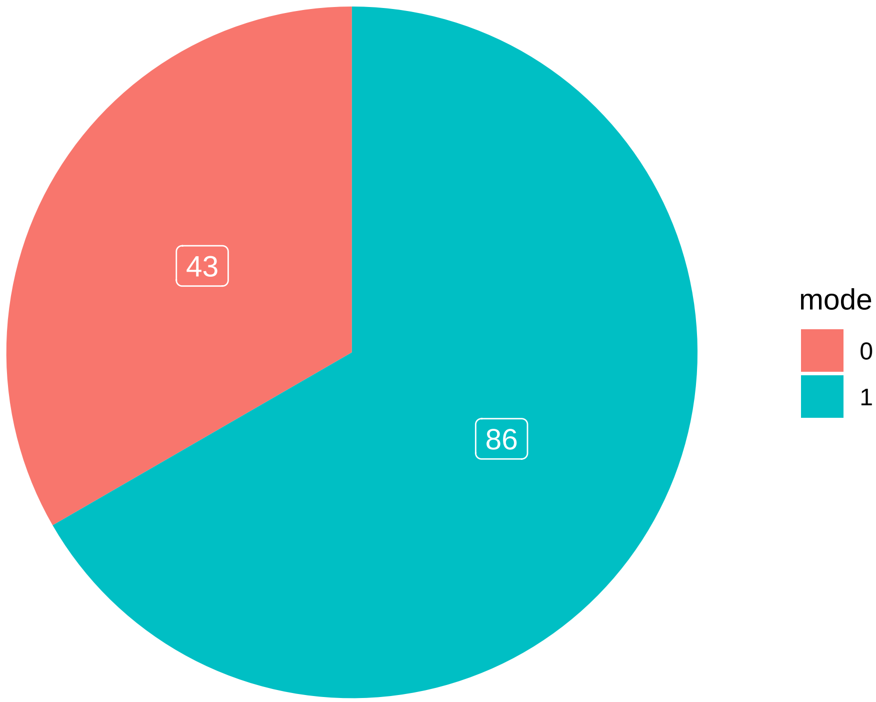


- Songs with overall loudness from around -7.5 to around -4

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-12.810  -6.720  -5.608  -5.815  -4.505  -1.190 
```

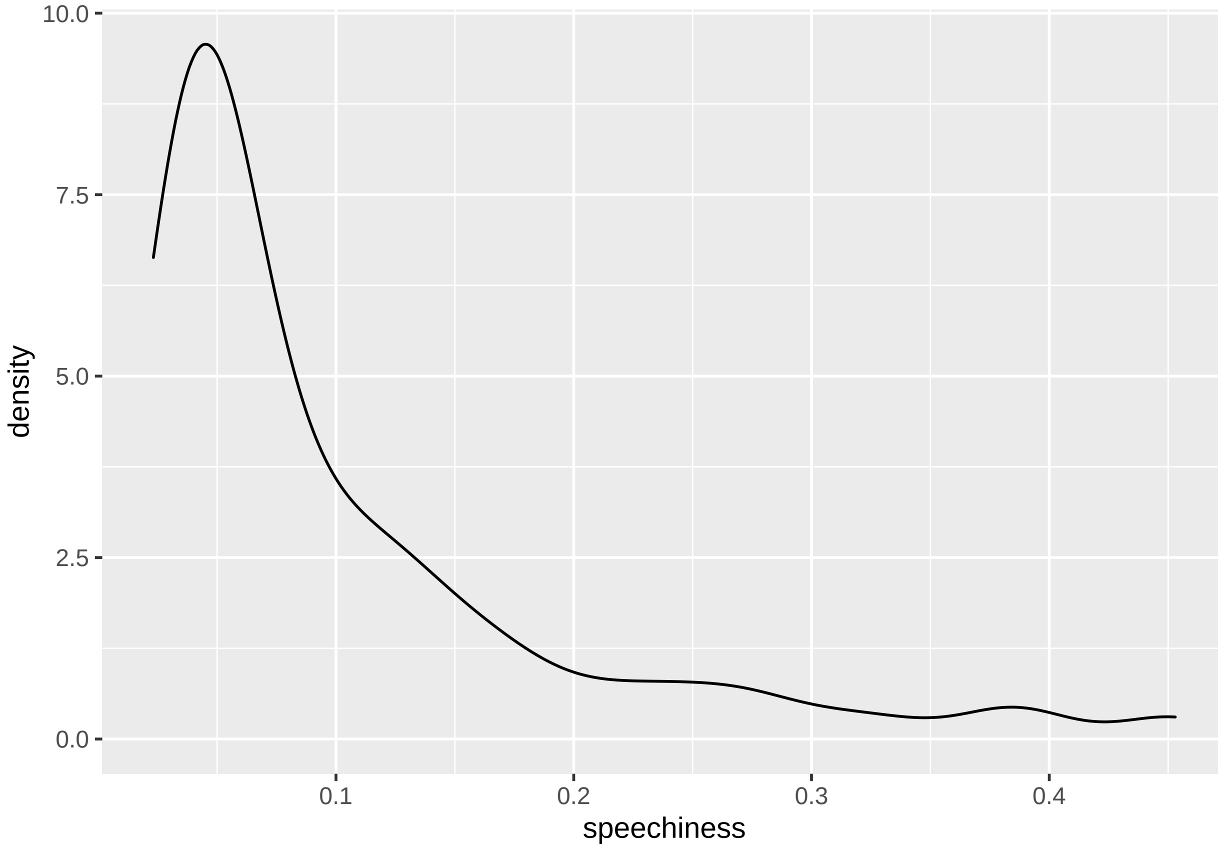


- Most songs are in major modality (mode = 1) which suggest a popularity in songs that sound more cheerful

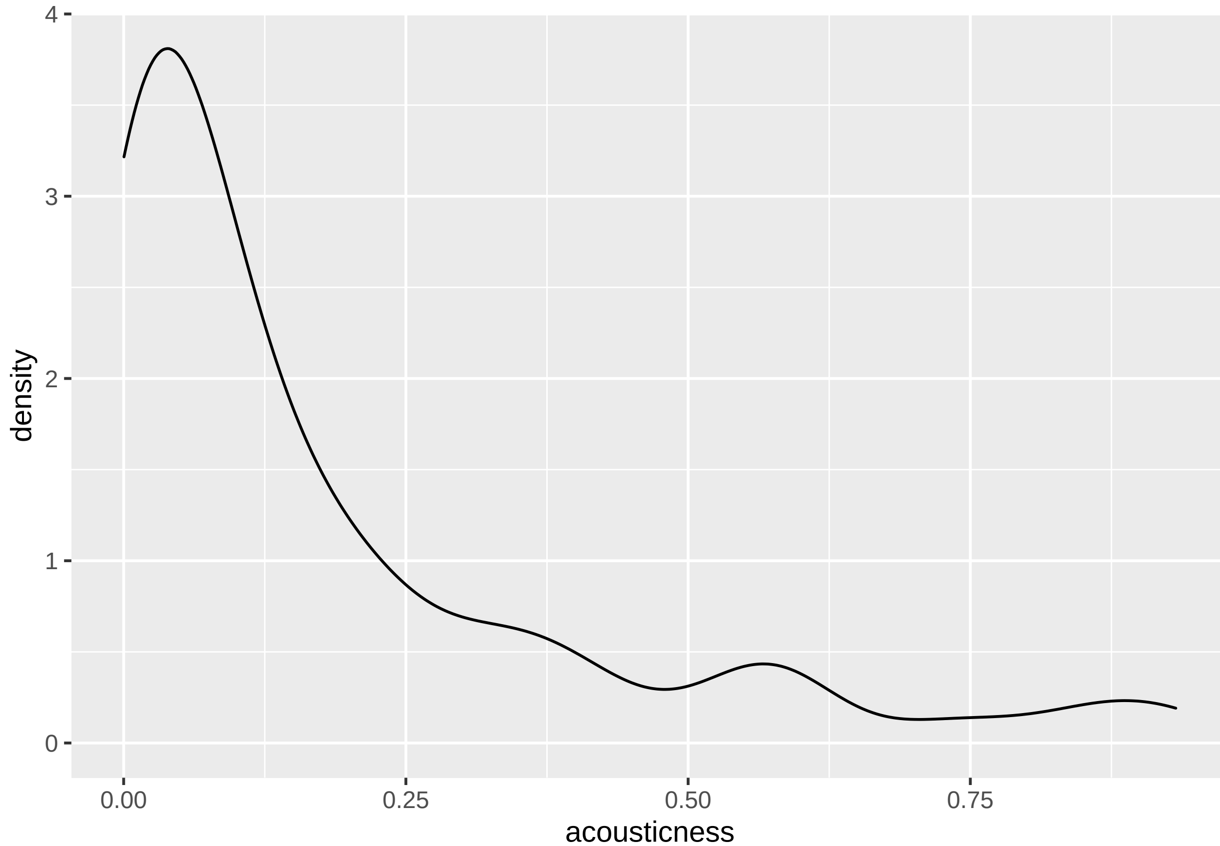


- Preference in songs with speechiness below 0.1 which suggest songs with less spoken words and more music are more likely to top the chart

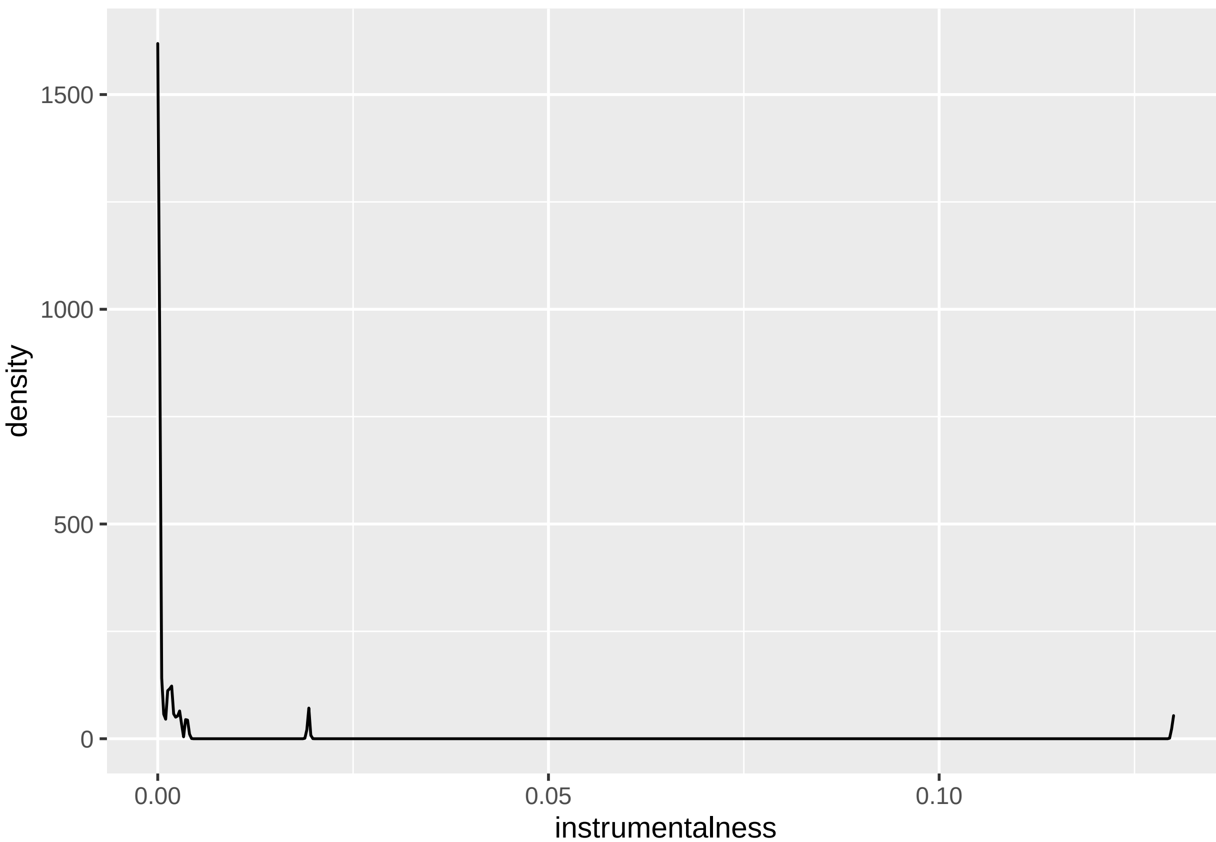
```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.0232  0.0421  0.0601  0.1019  0.1260  0.4530
```



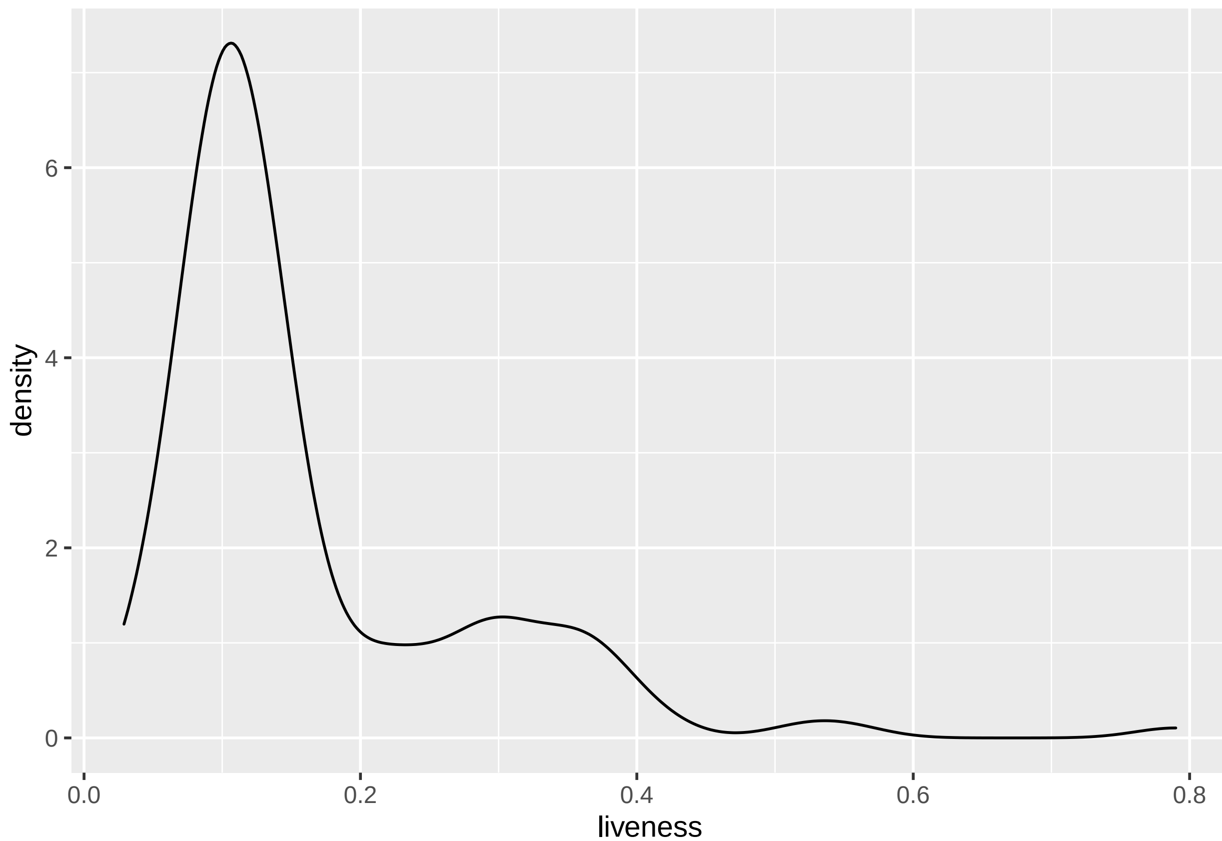
- Preference in songs with acousticness below 0.25 which suggest songs that are not acoustic are more likely to top the chart.



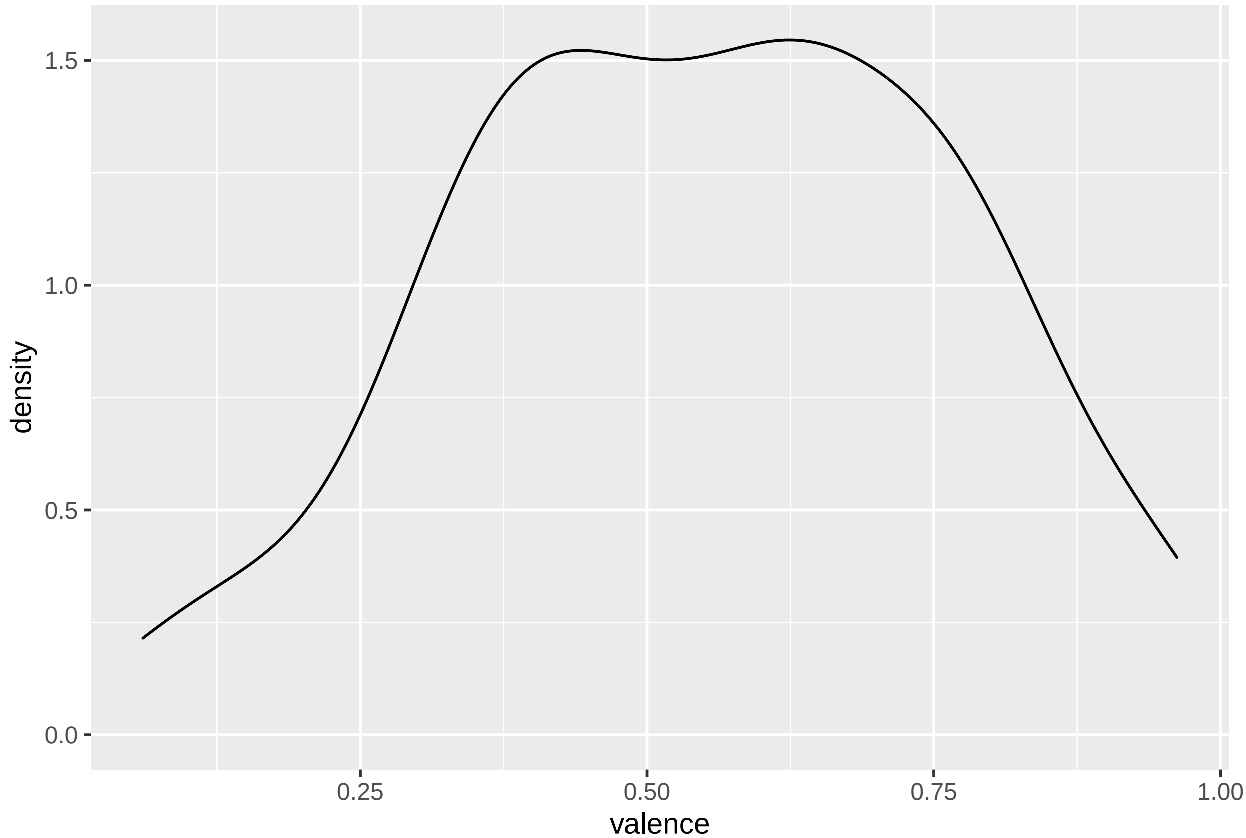
- Preference in songs with instrumentalness of 0 which suggests preference in songs with vocal contents but there are some exceptions (eg: there is 1 song with instrumentalness of 0.13 which suggest a likelihood of containing no vocal content)



- Preference in songs with liveness around 0.1 and moreover, none of the songs has liveness above 0.8. Therefore, it suggests a preference in tracks that are not live (studio recorded tracks).



- Most popular valance is from around 0.3 to 0.75



Another feature we want to look at is the number of weeks on chart of the songs. First, we re-order the songs by weeks_on_chart then use unique() so that each song only appears once with its highest number of weeks on chart (we can do this since all other variables that we need for our analysis will remain the same).

We created 2 new datasets from songs_after_2010: uniquie-all (contains all unique songs after 2010) and unique_top1 (contains songs that have reached number 1 on the chart which means having peak_position = 1).

We then pick the top 10 songs in each dataset that have the highest number of weeks on chart.

For songs that have reached number 1 (peak_position = 1), the top 10 songs are:

song	performer
Party Rock Anthem	LMFAO Featuring Lauren Bennett & GoonRock
Rolling In The Deep	Adele
Circles	Post Malone
Somebody That I Used To Know	Gotye Featuring Kimbra
All Of Me	John Legend
Shape Of You	Ed Sheeran
Dark Horse	Katy Perry Featuring Juicy J
I Gotta Feeling	The Black Eyed Peas
Perfect	Ed Sheeran
Uptown Funk!	Mark Ronson Featuring Bruno Mars

However, we see that a lot of songs that have never reached number 1 (peak_position != 1) can stay on the chart for a long time, even longer than songs that reached number 1. So we found that only 3 songs from the list above made into the top 10:

song	performer
Radioactive	Imagine Dragons
Sail	AWOLNATION
Blinding Lights	The Weeknd
Counting Stars	OneRepublic
Party Rock Anthem	LMFAO Featuring Lauren Bennett & GoonRock
Rolling In The Deep	Adele
I Hope	Gabby Barrett Featuring Charlie Puth
Ho Hey	The Lumineers
Circles	Post Malone
Demons	Imagine Dragons

We wanted to look into the songs that have never reached number 1 so we filtered them out and here are some features different from our observation of the dataset above:

- The genres are mostly rock and pop
- The length of the song is around 215000 ms to 289133 ms which is 3.6 min to 4.8 min
- The keys are mostly 1 (C sharp and D flat)
- The modes are all 1 (major) except Counting Star with 0 (minor)
- The speechiness are low, under 0.1 -> this represents preference in music and other non-speech-like tracks
- The acousticness is low overall with the exception of Ho Hey -> preference in non-acoustic songs (this supports the observation in popular genre: pop and rock)
- The liveness is low, under 0.1 except from Radioactive, Counting Star, Demon (by pop rock bands Imagine Dragons and OneRepublic) -> mostly pre-recorded tracks without audience voice
- The valence are all under 0.5 -> preference in negative sound songs (e.g. sad, depressed, angry) -> contradicts the popular valence of songs with long time on charts that reaches top 1

Tentative Analysis Question

From the top 100 Billboard dataset, our team want to determine:

- the effects of different variables have on a top billboard song.
- which variables have the most effect on bringing a song to the top (having peak position equals 1) or keeping a song in the top from time to time(having high number of weeks on chart).

Based on our observations of the dataset with 10 songs that have long time on chart and reached top 1, we have a hypothesis: For a song to reach top 1 and stay on the chart for a high number of weeks, it needs the following:

- Genre: Pop, Rock or Dance

- Duration: around 4.15 minutes
- Danceability: high danceability, around 0.7119
- Energy: high energy, around 0.6068
- Keys: 0
- Loudness: low loudness, around -5.6001
- Mode: 1 (major)
- Speechiness: low, around 0.0545
- Acousticness: low, around 0.266635
- Instrumentalness: low, around 0.00027165
- Liveness: low, around 0.13775
- Valence: around 0.5499 -> songs that sounds a bit negative
- Tempo: around 116.9884 -> fast tempo

Method

We are planning to try to use different regression models and nearest neighbor to analyze those questions.

For the regression model, we will use it to predict peak_position and weeks_on_chart using numerical variables (spotify_track_duration_ms, danceability, key, loudness, energy, speechiness, mode, acousticness, instrumentalness,liveness, valence, tempo). For the nearest neighbor, we will try to classify the song genre by their audio features.

Building model

First, we split data into train and test using createDataPartition from "caret" library. We choose to put 90% of the data into training data and the remaining 10% into test data. We only select peak_position, spotify_track_duration_ms, danceability, key, loudness, energy, speechiness, mode, acousticness, instrumentalness,liveness, valence, tempo, weeks_on_chart columns because they include the information needed.

Linear regression

Then we start building linear regression model, starting with model predicting weeks_on_chart:

```
[1] 9.096675
```

We found that this model predicting weeks_on_chart from loudness, spotify_track_duration_ms, liveness and tempo will result in the lowest RMSE of 9.117999.

Next is the model predicting peak_position:

```
[1] 29.50402
```

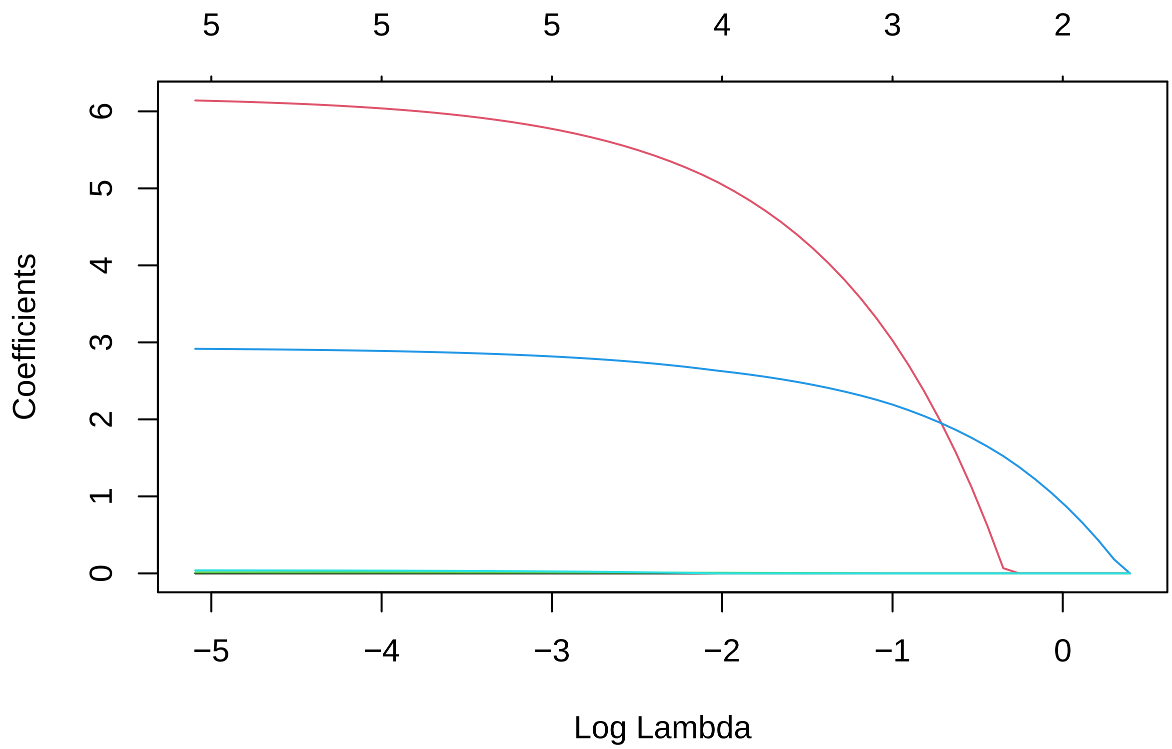
We found that this model predicting peak_position also from spotify_track_duration_ms, instrumentalness, tempo, mode and key will result in the lowest RMSE of 29.50429.

These two models have some similarities. They both use spotify_track_duration_ms and tempo to predict.

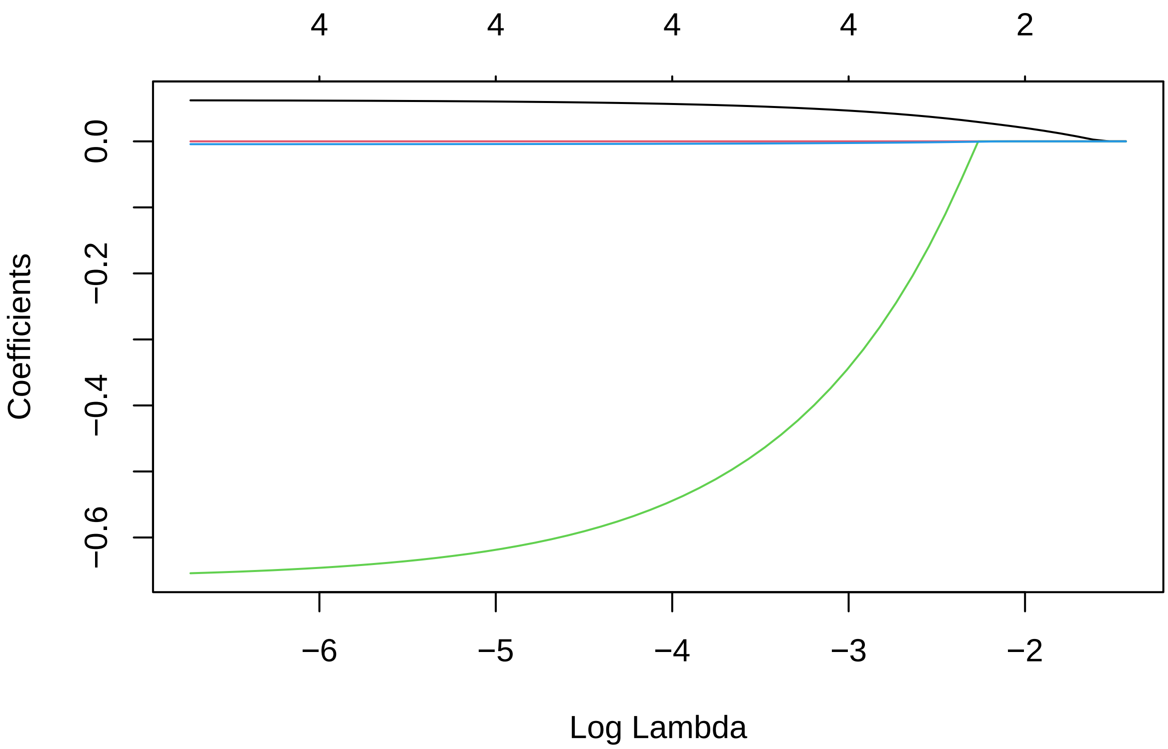
Lasso regression

We also try using Lasso regression to see if there is any improvement.

First we try to predict peak_position:




```
[1] 29.52553
```




```
[1] 9.137633
```

The RMSE's of the 2 regression model using Lasso regression do not improve with 9.15705 and 29.53312 for weeks_on_chart and peak_position models respectively.

Nearest neighbor

We try to classify the songs after 2010's genre using nearest neighbor method with k neighbors = 9.

Execute the python code (Nearest_Neighbor_Analysis.ipynb) with input: songs_after_2010_only_audio_features_genre.cs
from the code above

Output:

- predict genre: array([“[‘melodic rap’, ‘rap’, ‘trap’]”, “[‘dance pop’, ‘edm’, ‘electro house’, ‘house’, ‘pop’, ‘progressive house’, ‘tropical house’, ‘uk dance’]”,“[‘electropop’, ‘pop’, ‘tropical house’]”, . . . , “[‘dance pop’, ‘hip hop’, ‘miami hip hop’, ‘pop’, ‘pop rap’, ‘rap’, ‘southern hip hop’, ‘trap’]”,“[‘dance pop’, ‘pop’, ‘post-teen pop’]”, “[‘complextro’, ‘dance pop’, ‘edm’, ‘electro house’, ‘german techno’, ‘pop’, ‘post-teen pop’, ‘tropical house’]”], dtype=object)
- accuracy = 0.9126416739319965

Using 14 numeric variables about song's audio features: “spotify_track_duration_ms”,key“,”loudness“,”mode“,”speechiness",

“acousticness”,“instrumentalness”,“liveness”,“valence”,“time_signature”, “spotify_track_popularity”, “danceability”,“energy”,“tempo”, we are able to classify a song’s genre up to 91.26% accuracy with k-neighbor = 9.

#Discussion Comparing RMSE’s of the models and since the smaller the RMSE means the better the model, we found that the model with the lowest RMSE is the linear regression model.

First, we take a look at the coefficients of the linear regression model for weeks_on_chart.

	x
(Intercept)	5.0721103
loudness	0.0624929
spotify_track_duration_ms	0.0000034
liveness	-0.6617975
tempo	-0.0043399

This model has the lowest RMSE when using the following 4 variables: loudness, spotify_track_duration_ms, liveness and tempo.

According to this model, to stay on the chart for a long time, song needs to have the following characteristics: - loudness around 0.05 dB

- duration of the song around 0.0000032 ms
- liveness of 0.8261512 suggests a studio recorded track (not live)

Then we take a look at the coefficients of the model for peak_position.

	x
(Intercept)	45.4664541
spotify_track_duration_ms	-0.0000228
instrumentalness	6.1929532
tempo	0.0117395
mode	2.9300849
key	0.0414272

This model has the lowest RMSE when using the following 5 variables: spotify_track_duration_ms, instrumentalness, tempo, mode and key

According to this model, to stay on the chart for a long time, song needs to have the following characteristics: - tempo around 0.0067127

- key of the song is 0 (C)

Result

We will fit the values in the hypothesis into linear regression models to see if the predicted peak_position and weeks_on_chart.

```r
hypothesis_woc <- c(-5.6001, 249186.2, 0.13775, 116.9884)
x1 <- c(1, hypothesis_woc)
hypothesis_pp <- c(249186.2, 0.00027165, 116.9884, 1, 0)
x2 <- c(1, hypothesis_pp)

sum(x1*fit_woc$coefficients)
```

```
## [1] 4.979225
```

```r
sum(x2*fit_position$coefficients)
```

```
## [1] 44.09616
```

We see that the predicted peak_position and weeks_on_chart are not what we desired (large weeks_on_chart and low peak_position).

Conclusion

From the results above, we think that our hypothesis needs to consider more data than just looking at top 10 songs with high peak_position and large number of weeks_on_chart. If we classify the songs genre by their audio features, using nearest neighbor approach we have a pretty good accuracy around 90%.

Appendix

Code that are not shown in the report:

```r
billboard <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/

audio <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/202

summary(billboard)
```

```
##      url               week_id          week_position       song
##  Length:327895      Length:327895      Min.   :  1.0   Length:327895
##  Class :character   Class :character   1st Qu.: 25.5   Class :character
##  Mode  :character   Mode  :character   Median : 50.0   Mode  :character
##                                        Mean   : 50.5
##                                        3rd Qu.: 75.0
##                                        Max.   :100.0
##
##   performer           song_id             instance      previous_week_position
##  Length:327895      Length:327895      Min.   : 1.000   Min.   :  1.0
##  Class :character   Class :character   1st Qu.: 1.000   1st Qu.: 23.0
##  Mode  :character   Mode  :character   Median : 1.000   Median : 47.0
##                                        Mean   : 1.073   Mean   : 47.6
##                                        3rd Qu.: 1.000   3rd Qu.: 72.0
##                                        Max.   :10.000   Max.   :100.0
##                                                         NA's   :31954
```

```
##  peak_position    weeks_on_chart
##  Min.   :  1.00   Min.   : 1.000
##  1st Qu.: 14.00   1st Qu.: 4.000
##  Median : 39.00   Median : 7.000
##  Mean   : 41.36   Mean   : 9.154
##  3rd Qu.: 66.00   3rd Qu.:13.000
##  Max.   :100.00   Max.   :87.000
##
```

```
summary(audio)
```

```
##    song_id            performer              song            spotify_genre
##  Length:29503       Length:29503       Length:29503       Length:29503
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  spotify_track_id   spotify_track_preview_url spotify_track_duration_ms
##  Length:29503       Length:29503              Min.   :  29688
##  Class :character   Class :character          1st Qu.: 175053
##  Mode  :character   Mode  :character          Median : 214850
##                                               Mean   : 220684
##                                               3rd Qu.: 253253
##                                               Max.   :3079157
##                                               NA's   :5106
##  spotify_track_explicit spotify_track_album  danceability      energy
##  Mode :logical          Length:29503        Min.   :0.000   Min.   :0.001
##  FALSE:21449            Class :character    1st Qu.:0.499   1st Qu.:0.476
##  TRUE :2948             Mode  :character    Median :0.608   Median :0.634
##  NA's :5106                                 Mean   :0.600   Mean   :0.618
##                                             3rd Qu.:0.708   3rd Qu.:0.778
##                                             Max.   :0.988   Max.   :0.997
##                                             NA's   :5169    NA's   :5169
##       key            loudness             mode         speechiness
##  Min.   : 0.000   Min.   :-28.030   Min.   :0.000   Min.   :0.000
##  1st Qu.: 2.000   1st Qu.:-11.034   1st Qu.:0.000   1st Qu.:0.032
##  Median : 5.000   Median : -8.205   Median :1.000   Median :0.041
##  Mean   : 5.232   Mean   : -8.665   Mean   :0.727   Mean   :0.074
##  3rd Qu.: 8.000   3rd Qu.: -5.856   3rd Qu.:1.000   3rd Qu.:0.068
##  Max.   :11.000   Max.   :  2.291   Max.   :1.000   Max.   :0.951
##  NA's   :5169     NA's   :5169      NA's   :5169    NA's   :5169
##   acousticness   instrumentalness    liveness        valence
##  Min.   :0.000   Min.   :0.000    Min.   :0.010   Min.   :0.000
##  1st Qu.:0.047   1st Qu.:0.000    1st Qu.:0.091   1st Qu.:0.415
##  Median :0.195   Median :0.000    Median :0.131   Median :0.622
##  Mean   :0.295   Mean   :0.033    Mean   :0.192   Mean   :0.602
##  3rd Qu.:0.508   3rd Qu.:0.000    3rd Qu.:0.249   3rd Qu.:0.802
##  Max.   :0.991   Max.   :0.982    Max.   :0.999   Max.   :0.991
##  NA's   :5169    NA's   :5169     NA's   :5169    NA's   :5169
##      tempo        time_signature  spotify_track_popularity
##  Min.   :  0.00   Min.   :0.000   Min.   :  0.00
##  1st Qu.: 99.06   1st Qu.:4.000   1st Qu.: 23.00
```

```
##  Median :118.91   Median :4.000   Median : 43.00
##  Mean   :120.28   Mean   :3.932   Mean   : 41.22
##  3rd Qu.:136.48   3rd Qu.:4.000   3rd Qu.: 59.00
##  Max.   :241.01   Max.   :5.000   Max.   :100.00
##  NA's   :5169     NA's   :5169    NA's   :5106
```

```
library(tidyverse)

chart_audio = left_join(billboard, audio, by = c("song_id", "performer", "song"))

rank_num_week = chart_audio[order(chart_audio$weeks_on_chart, decreasing = TRUE),] %>%
  distinct(song, .keep_all = TRUE) %>%
  select(song, week_id) %>%
  head(10)
library(knitr)
kable(rank_num_week)
```

song	week_id
Radioactive	5/10/2014
Sail	3/22/2014
I'm Yours	10/10/2009
Blinding Lights	5/29/2021
How Do I Live	10/10/1998
Counting Stars	10/18/2014
Party Rock Anthem	7/21/2012
Foolish Games/You Were Meant For Me	2/21/1998
Rolling In The Deep	4/14/2012
Before He Cheats	12/1/2007

```
library(lubridate)

chart_audio$week_id = mdy(chart_audio$week_id)

songs_after_2010 = chart_audio %>% filter(week_id %in% c(ymd("2010-01-01"):today()))

descending_woc = songs_after_2010[order(
songs_after_2010$weeks_on_chart, decreasing = TRUE),]

top1_after_2010 = descending_woc %>% filter(peak_position == 1)

unique_all = descending_woc%>% distinct(song, .keep_all = TRUE)

unique_top1 = top1_after_2010 %>% distinct(song, .keep_all = TRUE)
```

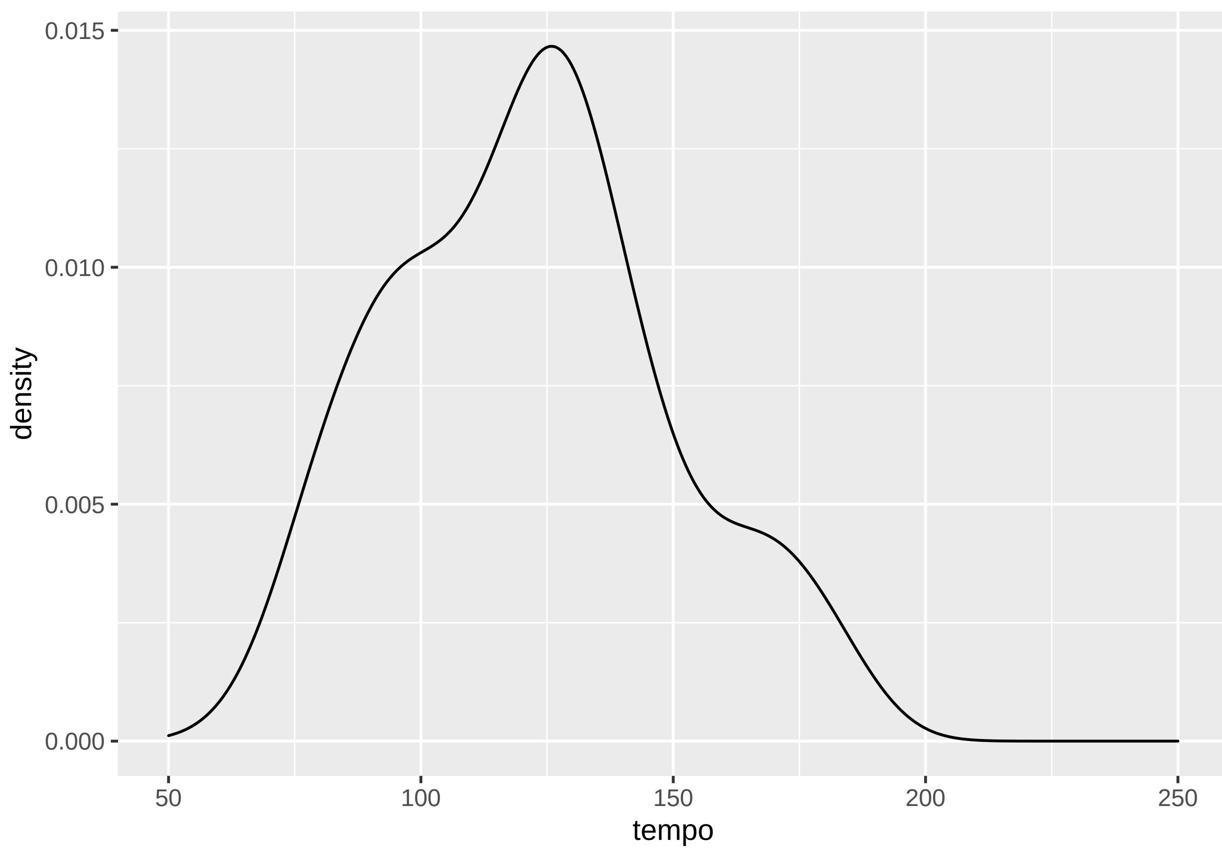
Code for visualization and table

tempo

```r
library(ggplot2)

count_tempo = unique_top1 %>%
  filter(!is.na(tempo))

summary(count_tempo$tempo)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   66.00   99.98  122.02  121.84  136.05  186.00
```

```r
count_tempo %>%
  ggplot(aes(x = tempo)) +
  geom_density()+
  xlim(50,250)
```



genre

```r
count_genre = unique_top1 %>% filter(!is.na(spotify_genre)) %>%
  count(spotify_genre) %>%
  arrange(desc(n))

kable(count_genre %>% top_n(5))
```

```
## Selecting by n
```

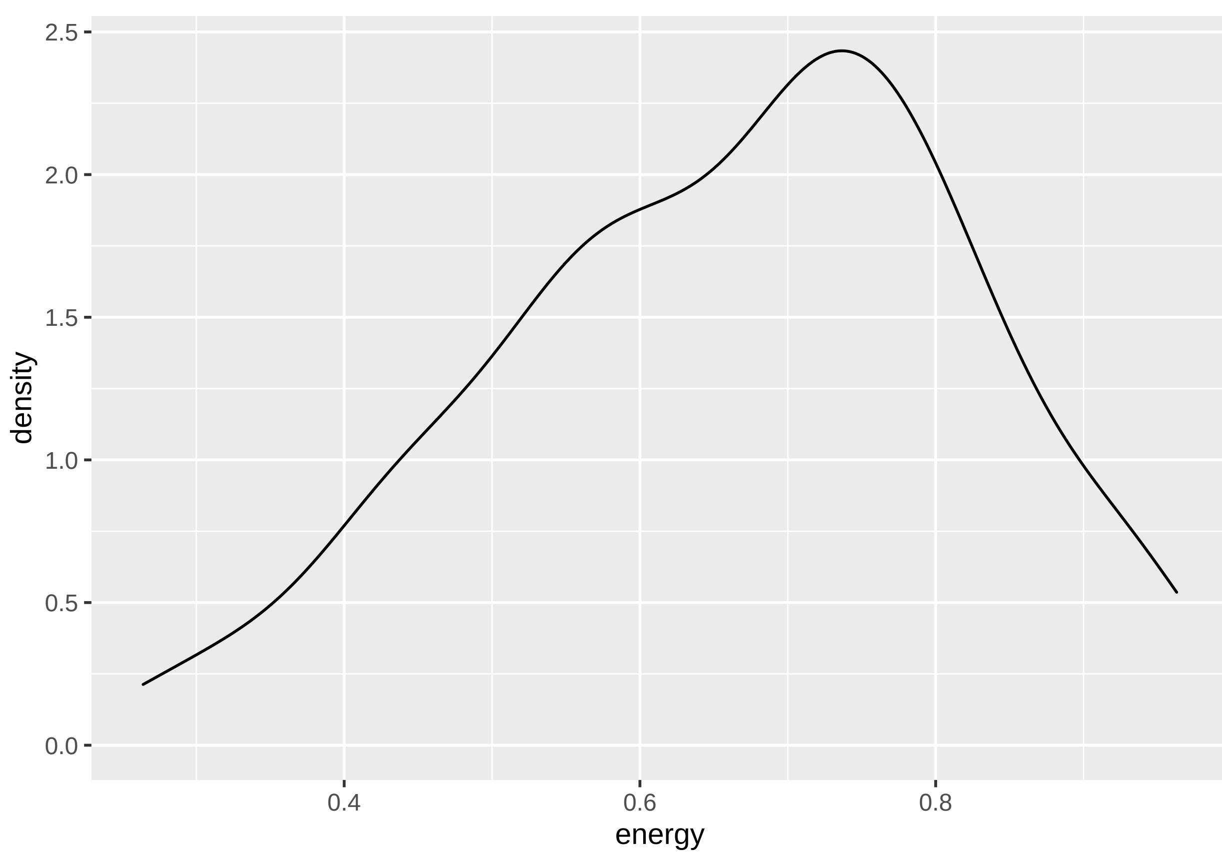
spotify_genre	n
['dance pop', 'pop', 'post-teen pop']	19
['barbadian pop', 'dance pop', 'pop', 'post-teen pop', 'r&b', 'urban contemporary']	7
['pop', 'post-teen pop']	7
['pop']	7
['canadian hip hop', 'canadian pop', 'hip hop', 'pop rap', 'rap', 'toronto rap']	6

energy

```
count_energy = unique_top1 %>%
  filter(!is.na(energy))

summary(count_energy$energy)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2640  0.5580  0.6930  0.6638  0.7720  0.9630
```

```
count_energy %>%
  ggplot(aes(x = energy)) +
  geom_density()
```

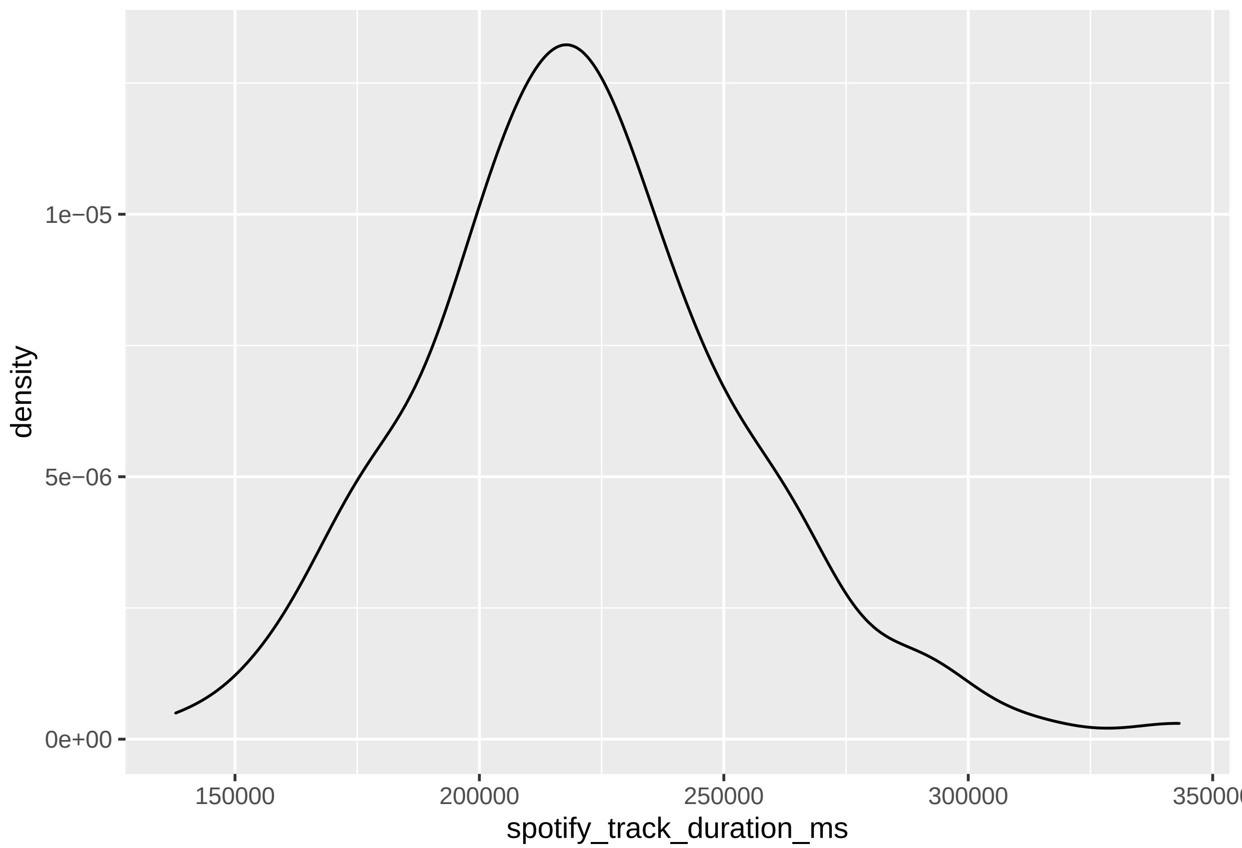


duration of the song

```
count_duration = unique_top1 %>%
  filter(!is.na(spotify_track_duration_ms))

summary(count_duration$spotify_track_duration_ms)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  137875  200080  219200  221112  241106  343150
```

```
count_duration %>%
  ggplot(aes(x = spotify_track_duration_ms)) +
  geom_density()
```

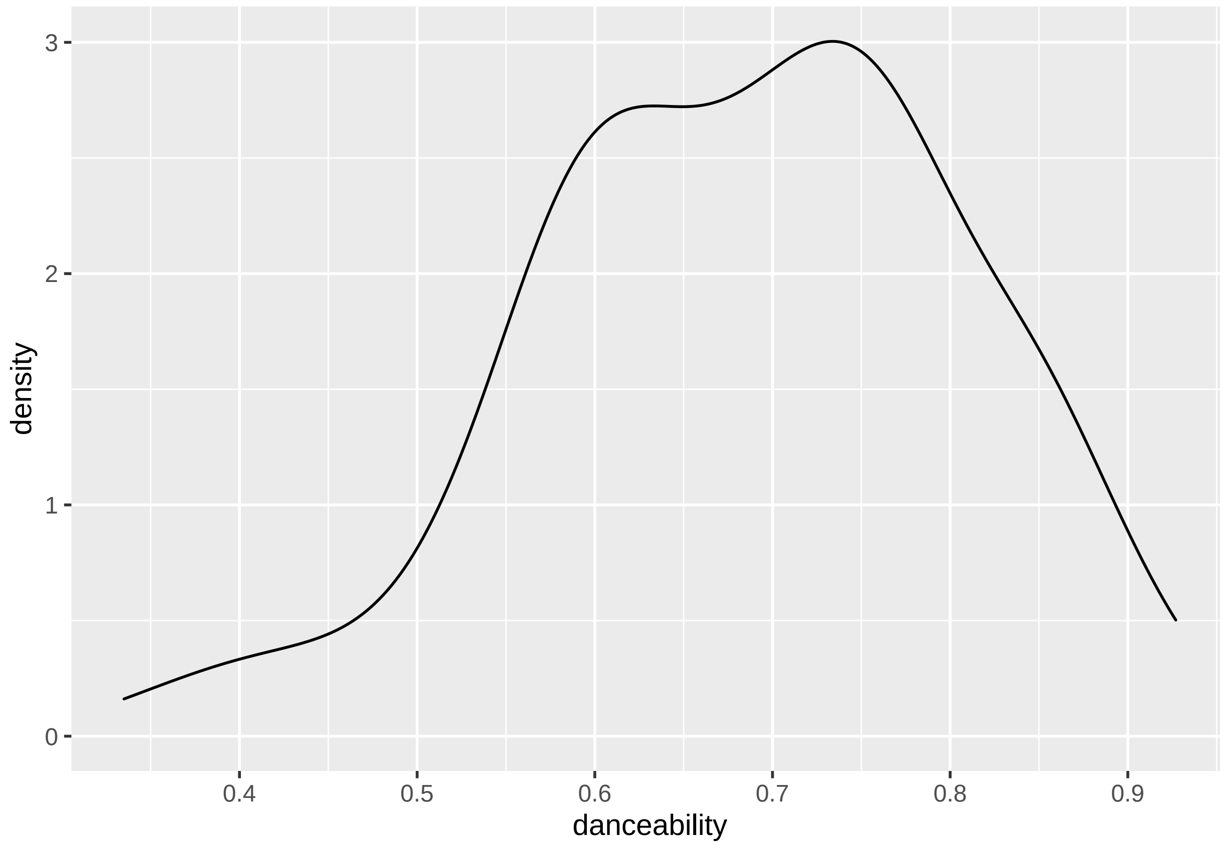


danceability

```
count_danceability = unique_top1 %>%
  filter(!is.na(danceability))

summary(count_danceability$danceability)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3350  0.6070  0.6970  0.6875  0.7780  0.9270
```

```
count_danceability %>%
  ggplot(aes(x = danceability)) +
  geom_density()
```



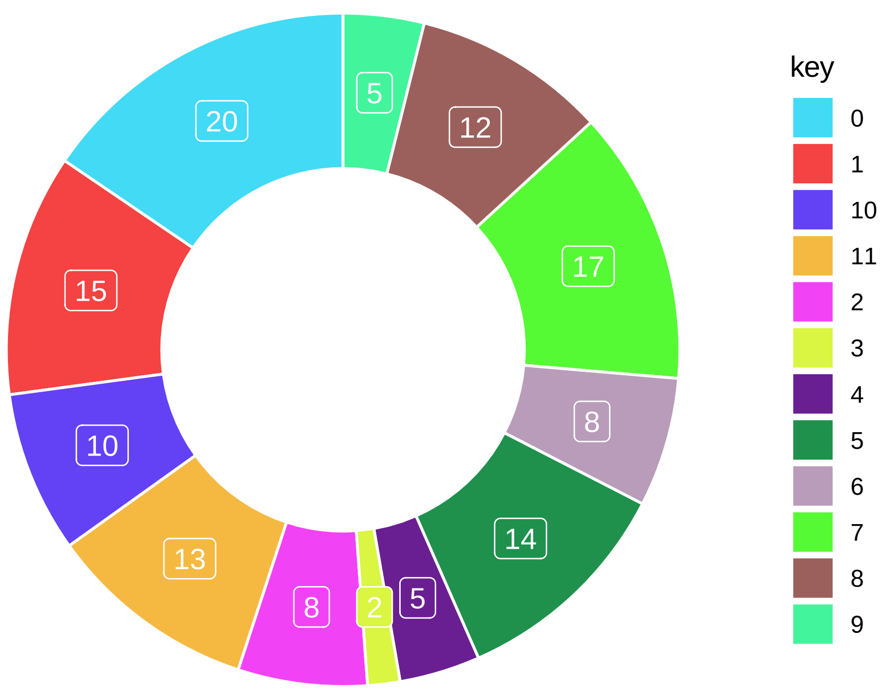
key

```
count_key = unique_top1 %>%
  filter(!is.na(key)) %>%
  count(key)%>%
  rename(count = n) %>%
  arrange(desc(count))

kable(count_key)
```

key	count
0	20
7	17
1	15
5	14
11	13
8	12
10	10

key	count
2	8
6	8
4	5
9	5
3	2

```
count_key$key = as.character(count_key$key)

mycols = c("#42daf5", "#f54242", "#6342f5", "#f5b942", "#f242f5", "#daf542", "#691f91", "#1f914d", "#b9
count_key %>%
  ggplot(aes(x = 2, y = count, fill = key)) +
  geom_bar(stat="identity", color = "white") +
  geom_label(aes(label = count),
             color = "white",
             position = position_stack(vjust = 0.5),
             show.legend = FALSE) +
  coord_polar(theta = "y", start = 0)+
  scale_fill_manual(values = mycols) +
  theme_void() +
  xlim(0.5, 2.5)
```

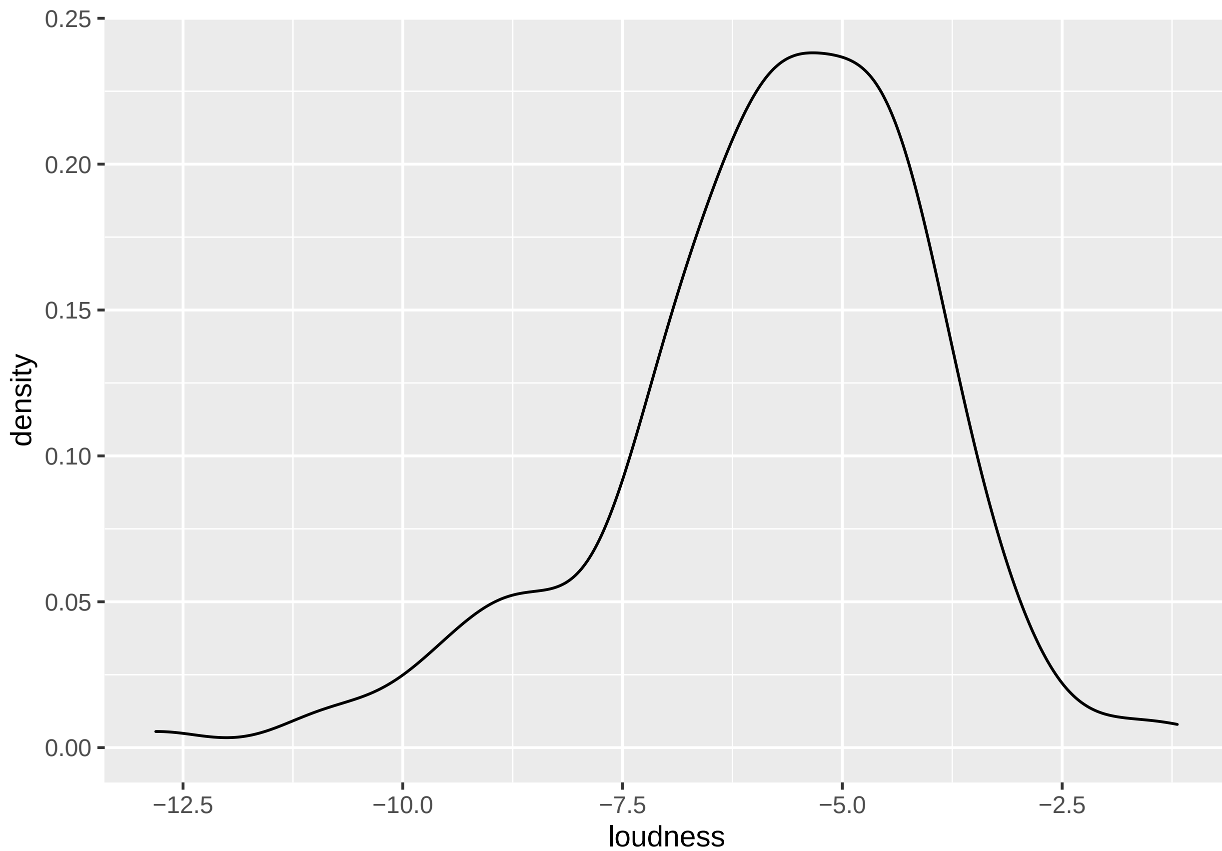


loudness

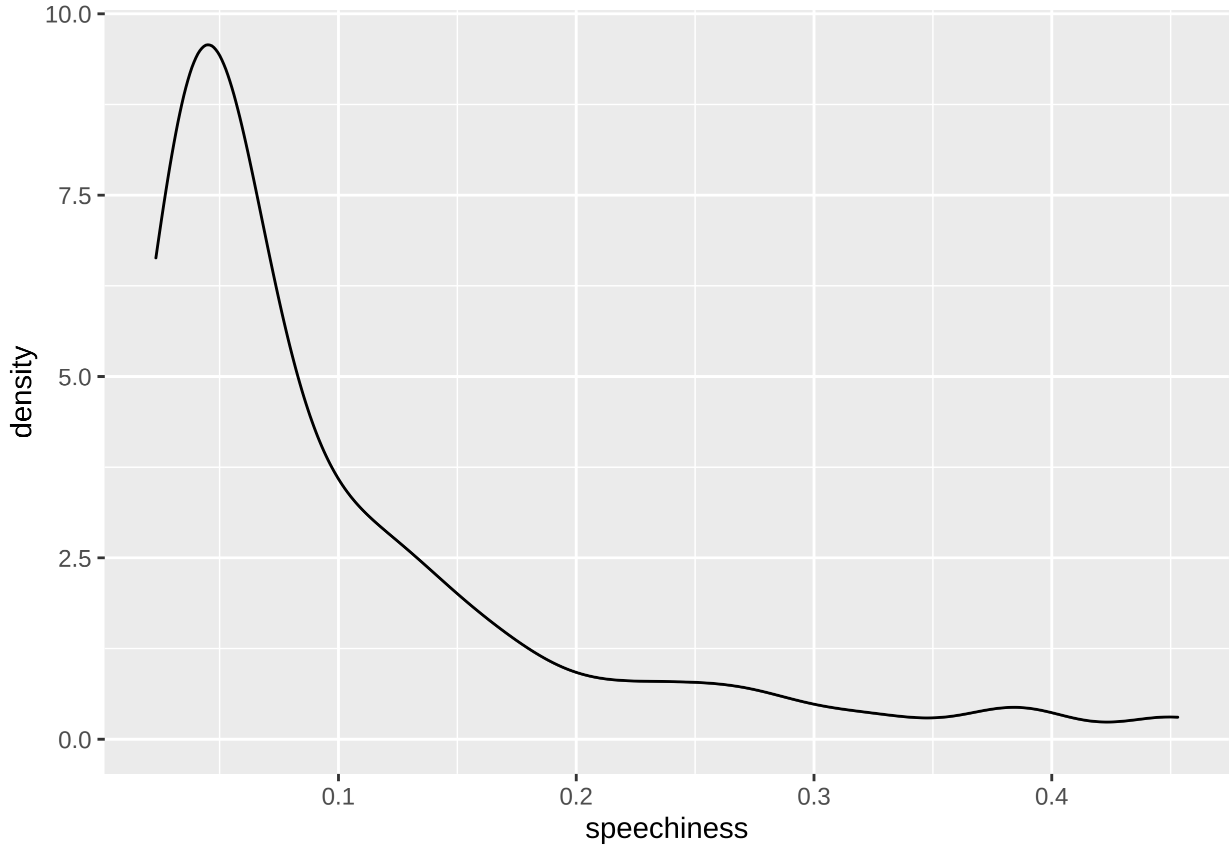
```
count_loudness = unique_top1 %>%
  filter(!is.na(loudness))

summary(count_loudness$loudness)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -12.810  -6.720  -5.608  -5.815  -4.505  -1.190
```

```
count_loudness %>%
  ggplot(aes(x = loudness)) +
  geom_density()
```




```
count_speechiness %>%
  ggplot(aes(x = speechiness)) +
  geom_density()
```



speechiness

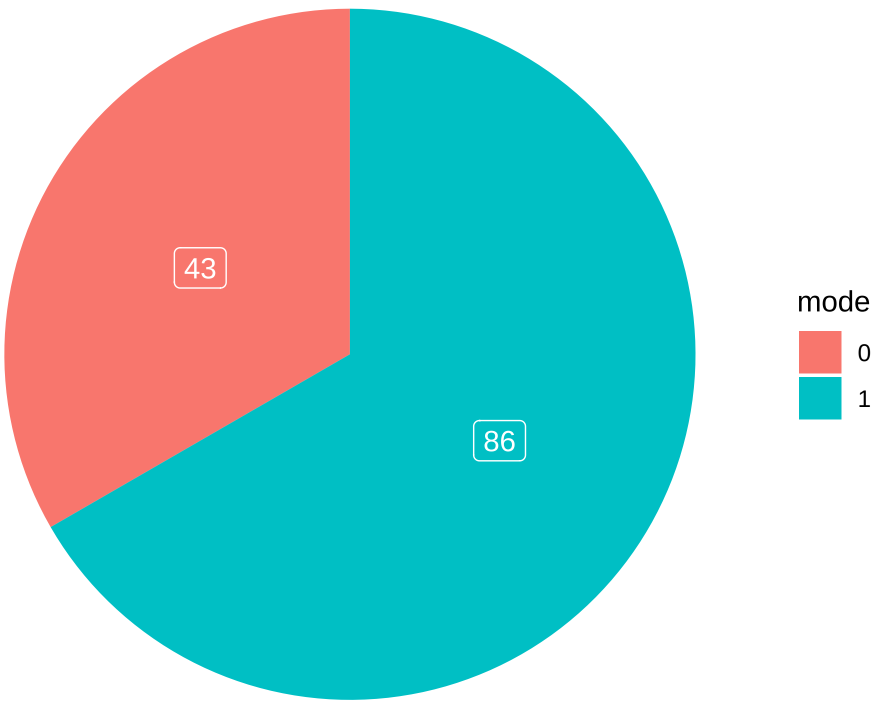
```
count_speechiness = unique_top1 %>%
  filter(!is.na(speechiness)) %>%
  count(speechiness)
summary(count_speechiness$speechiness)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0232  0.0421  0.0601  0.1019  0.1260  0.4530
```

mode

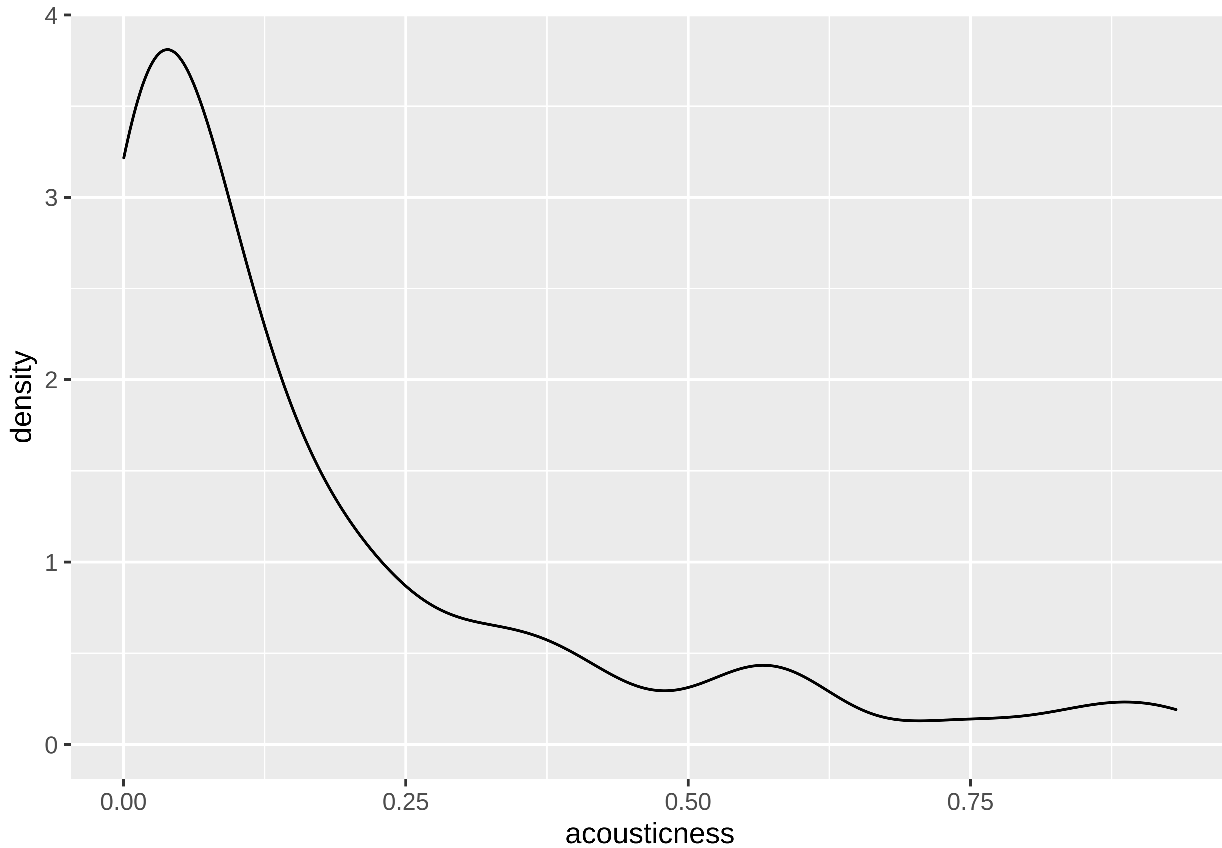
```
count_mode = unique_top1 %>%
  filter(!is.na(mode)) %>%
  count(mode) %>%
  rename(freq = n)
count_mode$mode = as.character(count_mode$mode)

count_mode %>%
  ggplot(aes(x = "", y = freq, fill = mode)) +
  geom_bar(stat="identity", width=1) +
  geom_label(aes(label = freq),
```

```
              color = "white",
              position = position_stack(vjust = 0.5),
              show.legend = FALSE) +
  coord_polar(theta = "y")+
  theme_void()
```



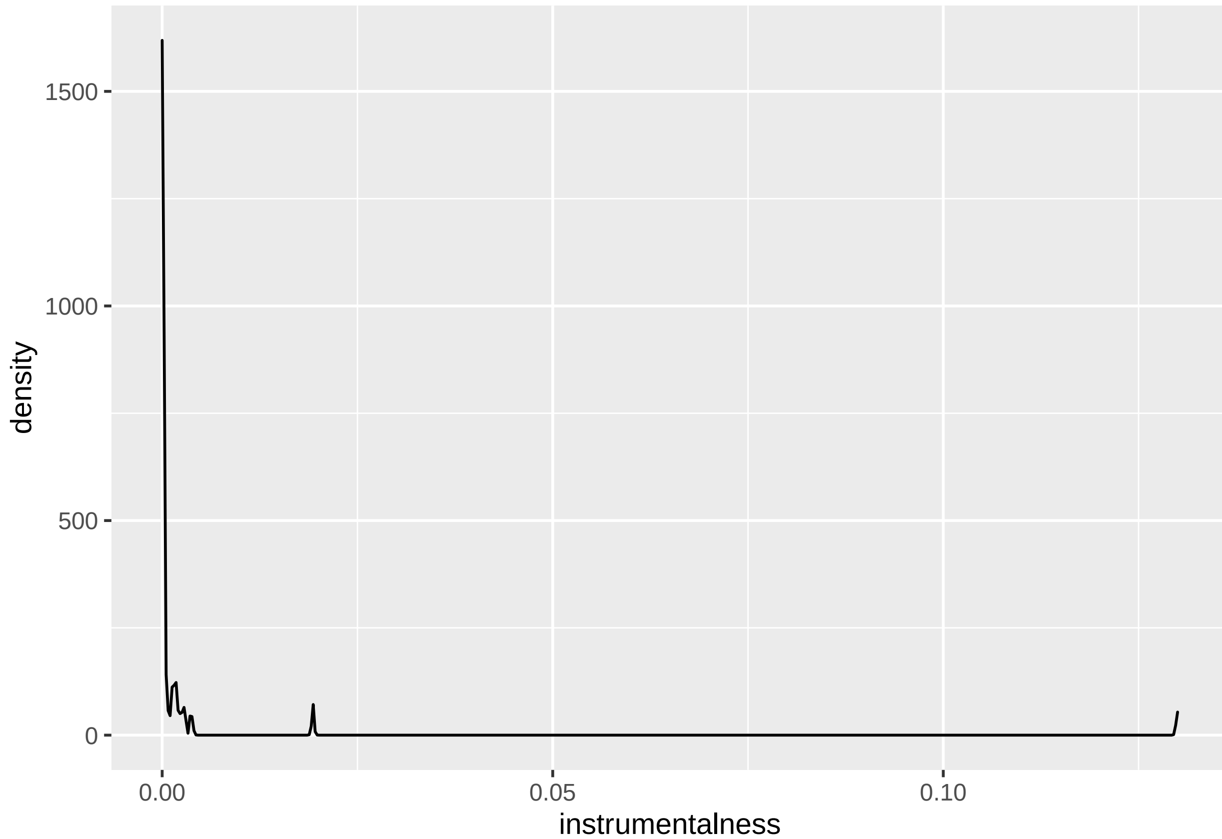
acousticness

```
count_acousticness = unique_top1 %>%
  filter(!is.na(acousticness))

count_acousticness %>%
  ggplot(aes(x = acousticness)) +
  geom_density()
```



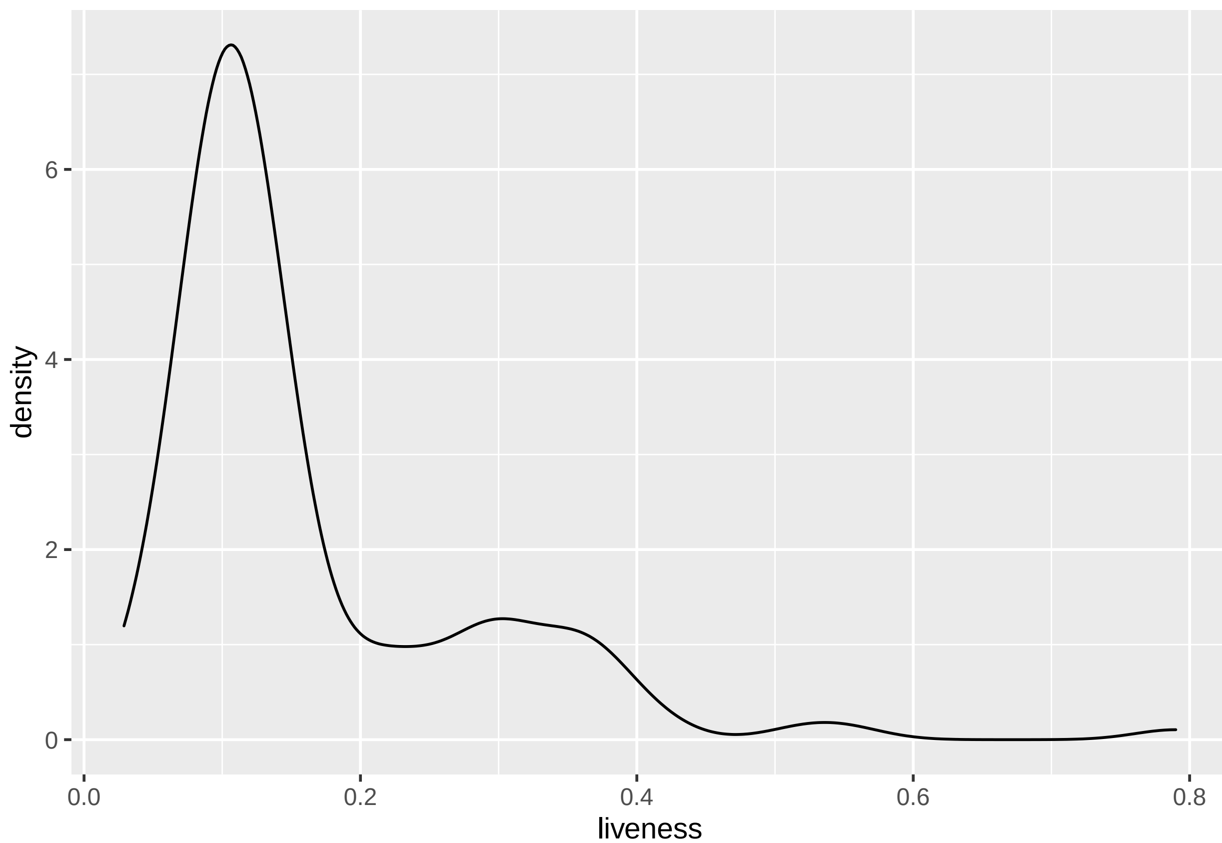
instrumentalness

```
count_instrumentalness = unique_top1 %>%
  filter(!is.na(instrumentalness)) %>%
  count(instrumentalness) %>%
  arrange(desc(n))

count_instrumentalness %>%
  ggplot(aes(x = instrumentalness)) +
  geom_density()
```



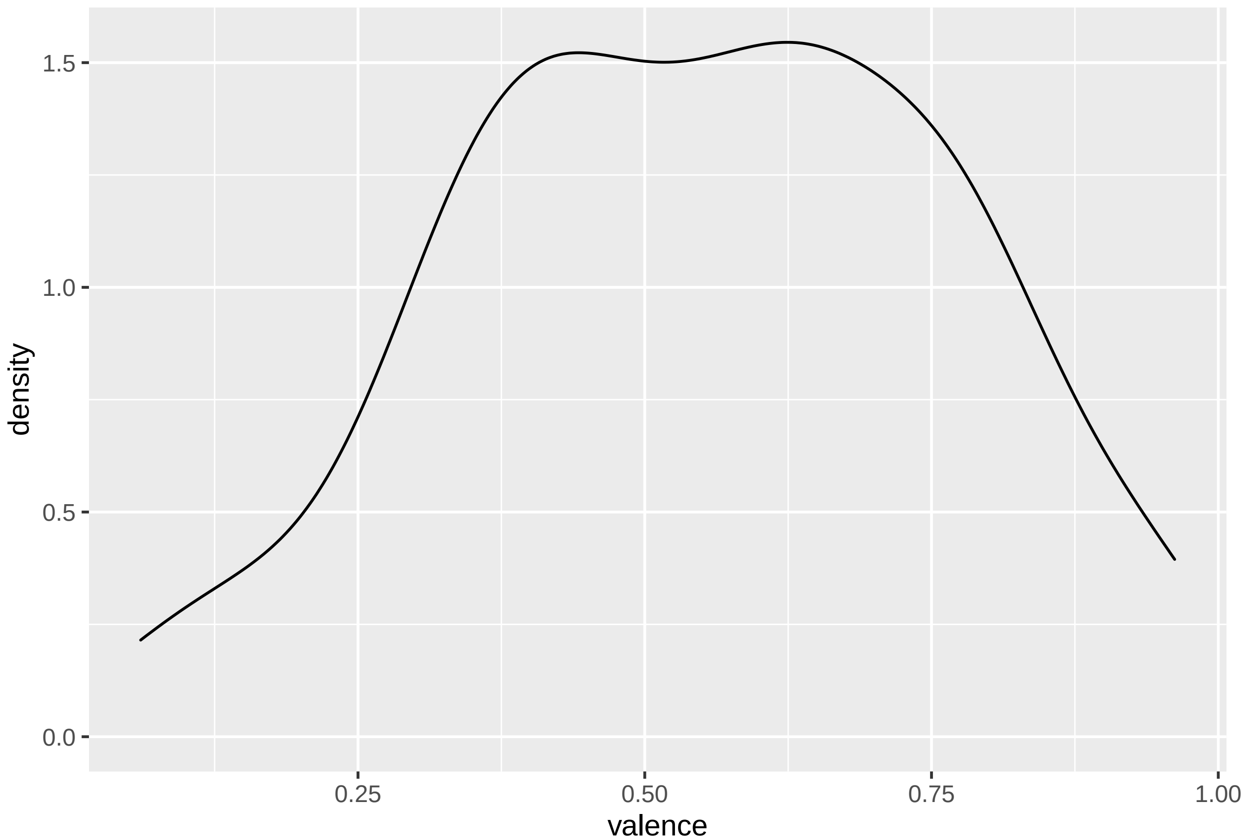
liveness

```
count_liveness = unique_top1 %>%
  filter(!is.na(liveness))

count_liveness %>%
  ggplot(aes(x = liveness)) +
  geom_density()
```



valence

```
count_val = unique_top1 %>%
  filter(!is.na(valence))

count_val %>%
  ggplot(aes(x = valence)) +
  geom_density()
```



rank by number of weeks on chart

```
rank_num_week = unique_top1[order(unique_top1$weeks_on_chart, decreasing = TRUE),]
top10_woc = head(rank_num_week, 10)
top10_woc$spotify_genre
```

```
##  [1] "['dance pop', 'pop', 'pop rap']"
##  [2] "['british soul', 'pop', 'uk pop']"
##  [3] "['dfw rap', 'melodic rap', 'rap']"
##  [4] "['australian pop']"
##  [5] "['neo mellow', 'neo soul', 'pop', 'r&b', 'urban contemporary']"
##  [6] "['pop', 'uk pop']"
##  [7] "['dance pop', 'pop', 'post-teen pop']"
##  [8] "['dance pop', 'pop', 'pop rap']"
##  [9] "['pop', 'uk pop']"
## [10] "['dance pop', 'pop']"
```

```
top10_woc$spotify_genre
```

```
##  [1] "['dance pop', 'pop', 'pop rap']"
##  [2] "['british soul', 'pop', 'uk pop']"
##  [3] "['dfw rap', 'melodic rap', 'rap']"
##  [4] "['australian pop']"
```

```
##  [5] "['neo mellow', 'neo soul', 'pop', 'r&b', 'urban contemporary']"
##  [6] "['pop', 'uk pop']"
##  [7] "['dance pop', 'pop', 'post-teen pop']"
##  [8] "['dance pop', 'pop', 'pop rap']"
##  [9] "['pop', 'uk pop']"
## [10] "['dance pop', 'pop']"
```

```
top10_woc$spotify_track_duration_ms
```

```
##  [1] 262173 228293 215280 244973 269560 233712 215672 289133 263400 269666
```

```
top10_woc$danceability
```

```
##  [1] 0.750 0.729 0.695 0.857 0.422 0.825 0.645 0.741 0.599 0.856
```

```
top10_woc$energy
```

```
##  [1] 0.727 0.756 0.762 0.517 0.264 0.652 0.585 0.748 0.448 0.609
```

```
top10_woc$key
```

```
##  [1] 5 8 0 0 8 1 6 0 8 0
```

```
top10_woc$loudness
```

```
##  [1] -4.210 -5.119 -3.497 -6.972 -7.064 -3.183 -6.122 -6.299 -6.312 -7.223
```

```
top10_woc$mode
```

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

```
top10_woc$speechiness
```

```
##  [1] 0.1420 0.0294 0.0395 0.0384 0.0322 0.0802 0.0513 0.0264 0.0232 0.0824
```

```
top10_woc$acousticness
```

```
##  [1] 0.01890 0.13100 0.19200 0.56500 0.92200 0.58100 0.00314 0.08230 0.16300
## [10] 0.00801
```

```
top10_woc$instrumentalness
```

```
##  [1] 0.00e+00 0.00e+00 2.44e-03 1.95e-04 0.00e+00 0.00e+00 0.00e+00 0.00e+00
##  [9] 0.00e+00 8.15e-05
```

```r
top10_woc$liveness
```

```
##  [1] 0.2660 0.0527 0.0863 0.1020 0.1320 0.0931 0.1650 0.3400 0.1060 0.0344
```

```r
top10_woc$valence
```

```
##  [1] 0.359 0.522 0.553 0.754 0.331 0.931 0.353 0.600 0.168 0.928
```

```r
top10_woc$tempo
```

```
##  [1] 129.993 104.945 120.042 129.063 119.930  95.977 131.931 127.965  95.050
## [10] 114.988
```

```r
highest_num_week = unique_all[order(unique_all$weeks_on_chart, decreasing = TRUE),]
head(highest_num_week$song, 10)
```

```
##  [1] "Radioactive"         "Sail"                "Blinding Lights"
##  [4] "Counting Stars"      "Party Rock Anthem"   "Rolling In The Deep"
##  [7] "I Hope"              "Ho Hey"              "Circles"
## [10] "Demons"
```

```r
highest_woc = head(highest_num_week,10) %>%  filter(peak_position != 1)

highest_woc$spotify_genre
```

```
## [1] "['modern rock']"
## [2] "['indie pop', 'la indie', 'modern alternative rock', 'modern rock', 'pop rock', 'rock', 'stomp p
## [3] "['canadian contemporary r&b', 'canadian pop', 'pop']"
## [4] "['dance pop', 'neo mellow', 'piano rock', 'pop', 'pop rock']"
## [5] NA
## [6] "['folk-pop', 'modern rock', 'stomp and holler']"
## [7] "['modern rock']"
```

```r
highest_woc$spotify_track_duration_ms
```

```
## [1] 186813 259102 201573 257839     NA 163133 175200
```

```r
highest_woc$danceability
```

```
## [1] 0.448 0.825 0.513 0.664    NA 0.685 0.505
```

```r
highest_woc$energy
```

```
## [1] 0.784 0.435 0.796 0.705    NA 0.466 0.710
```

```r
highest_woc$key
```

```
## [1]  9  1  1  1 NA  0  3
```

```r
highest_woc$loudness
```

```
## [1] -3.686 -9.582 -4.075 -4.972     NA -9.074 -3.015
```

```r
highest_woc$mode
```

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

```r
highest_woc$speechiness
```

```
## [1] 0.0627 0.0568 0.0629 0.0382     NA 0.0304 0.0321
```

```r
highest_woc$acousticness
```

```
## [1] 0.10600 0.45200 0.00147 0.06540      NA 0.79400 0.19000
```

```r
highest_woc$instrumentalness
```

```
## [1] 1.08e-04 6.09e-01 2.09e-04 0.00e+00       NA 2.06e-06 2.50e-04
```

```r
highest_woc$liveness
```

```
## [1] 0.6680 0.0953 0.0938 0.1150     NA 0.0915 0.3290
```

```r
highest_woc$valence
```

```
## [1] 0.236 0.243 0.345 0.477    NA 0.353 0.428
```

```r
highest_woc$tempo
```

```
## [1] 136.245 119.038 171.017 122.017      NA  79.936  89.938
```

Code for Building models

```r
library(caret)

index <- createDataPartition(unique_all$peak_position, p = .90, list = FALSE)
train <- chart_audio[index, ]
test <- chart_audio[-index, ]

train <- select(train, peak_position, spotify_track_duration_ms, danceability, key, loudness, energy,
                speechiness, mode, acousticness, instrumentalness, liveness, valence, tempo,
                weeks_on_chart)
```

linear regression

- Predicting weeks_on_chart

```
train_woc = train %>% select(-peak_position)
fit_woc <- lm(weeks_on_chart ~ loudness + spotify_track_duration_ms + liveness + tempo,
              data =train_woc)
kable(fit_woc$coefficients)
```

	x
(Intercept)	4.8782361
loudness	0.0441189
spotify_track_duration_ms	0.0000030
liveness	-0.6880831
tempo	-0.0032878

```
test <- test %>% filter(!is.na(spotify_track_duration_ms), !is.na(danceability), !is.na(key),
                             !is.na(loudness), !is.na(energy), !is.na(speechiness), !is.na(mode),
                             !is.na(acousticness), !is.na(instrumentalness), !is.na(liveness),
                             !is.na(valence), !is.na(tempo))

pred_vals_woc <- predict(fit_woc, newdata = test)

target_woc <- test$weeks_on_chart #observed val

rmse_woc <- sqrt(mean((target_woc - pred_vals_woc)^2))
rmse_woc
```

```
## [1] 9.1125
```

- Predicting peak_position

```
train_pp = train %>% select(-weeks_on_chart)

fit_position <- lm(peak_position ~ spotify_track_duration_ms + instrumentalness + tempo + mode + key,

                   data = train_pp)
kable(fit_position$coefficients)
```

	x
(Intercept)	43.6009036
spotify_track_duration_ms	-0.0000190
instrumentalness	6.3592451
tempo	0.0184489
mode	2.7699921
key	0.0624038

```
pred_vals_position <- predict(fit_position, newdata = test)

target_position <- test$peak_position #observed val

rmse_position <- sqrt(mean((target_position - pred_vals_position)^2))
rmse_position
```

```
## [1] 29.47454
```

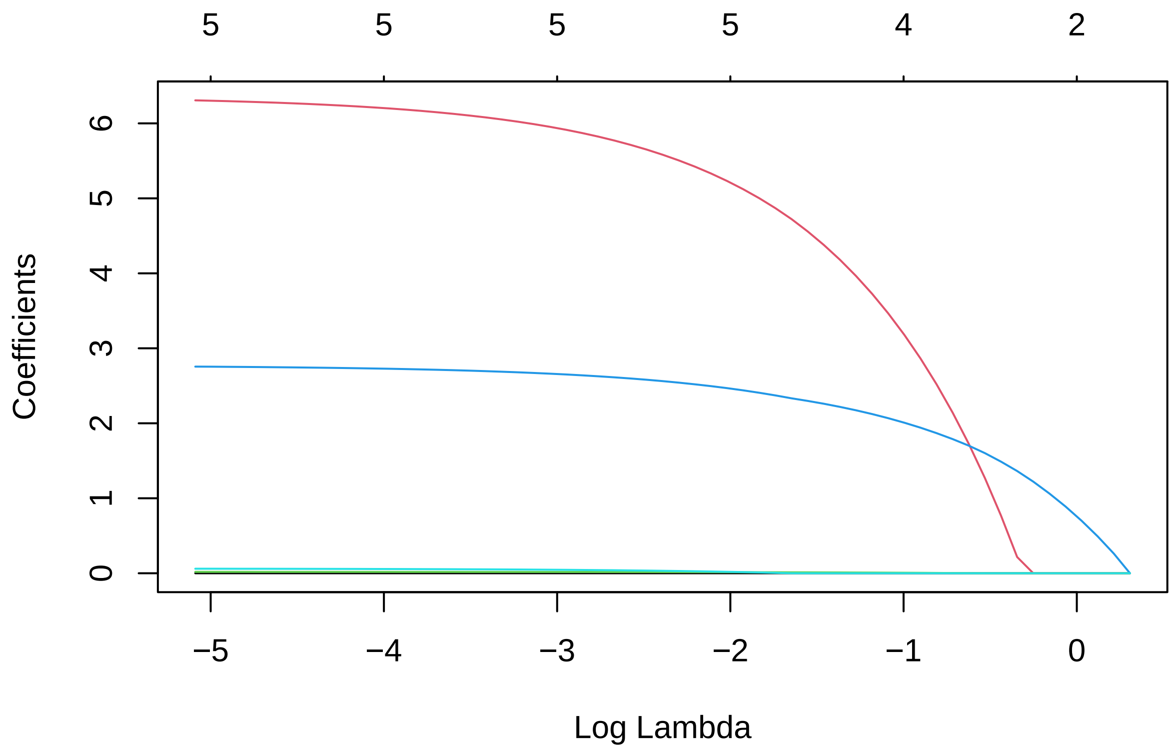
lasso regression

- Predicting peak_position

```
train_lasso_pp = train_pp %>%
  filter(!is.na(spotify_track_duration_ms), !is.na(danceability), !is.na(key), !is.na(loudness),
         !is.na(energy), !is.na(speechiness), !is.na(mode), !is.na(acousticness),
         !is.na(instrumentalness), !is.na(liveness), !is.na(valence), !is.na(tempo))


fit_peak <- lm(peak_position ~ spotify_track_duration_ms + instrumentalness + tempo + mode + key,
               data = train_lasso_pp)
kable(fit_peak$coefficients)
```

	x
(Intercept)	43.6009036
spotify_track_duration_ms	-0.0000190
instrumentalness	6.3592451
tempo	0.0184489
mode	2.7699921
key	0.0624038

```
X_peak <- model.matrix(fit_peak)
y_peak <- train_lasso_pp$peak_position

beta_ols_peak <- solve(crossprod(X_peak)) %*% crossprod(X_peak, y_peak)

lambda_peak <- 1.0
p_peak <- ncol(X_peak)
beta_ridge_peak <- solve(crossprod(X_peak) + diag(lambda_peak, ncol = p_peak, nrow = p_peak)) %*%
  crossprod(X_peak, y_peak)

library(glmnet)
X_peak <- X_peak[, -1]
fit_lasso_peak <- glmnet(X_peak, y_peak)
plot(fit_lasso_peak, xvar = "lambda")
```




```
X_test_peak <- model.matrix(~ spotify_track_duration_ms + instrumentalness + tempo + mode + key, data =
X_test_peak <- as.matrix(X_test_peak)

y_test_peak <- test$peak_position
y_pred_peak <- X_test_peak %*% beta_ridge_peak

X_test_peak <- X_test_peak[, -1]
pred_lasso_peak <- predict(fit_lasso_peak, newx = X_test_peak, s = 1.0)
rmse_lasso_peak <- sqrt(mean((y_test_peak - pred_lasso_peak)^2))
rmse_lasso_peak
```

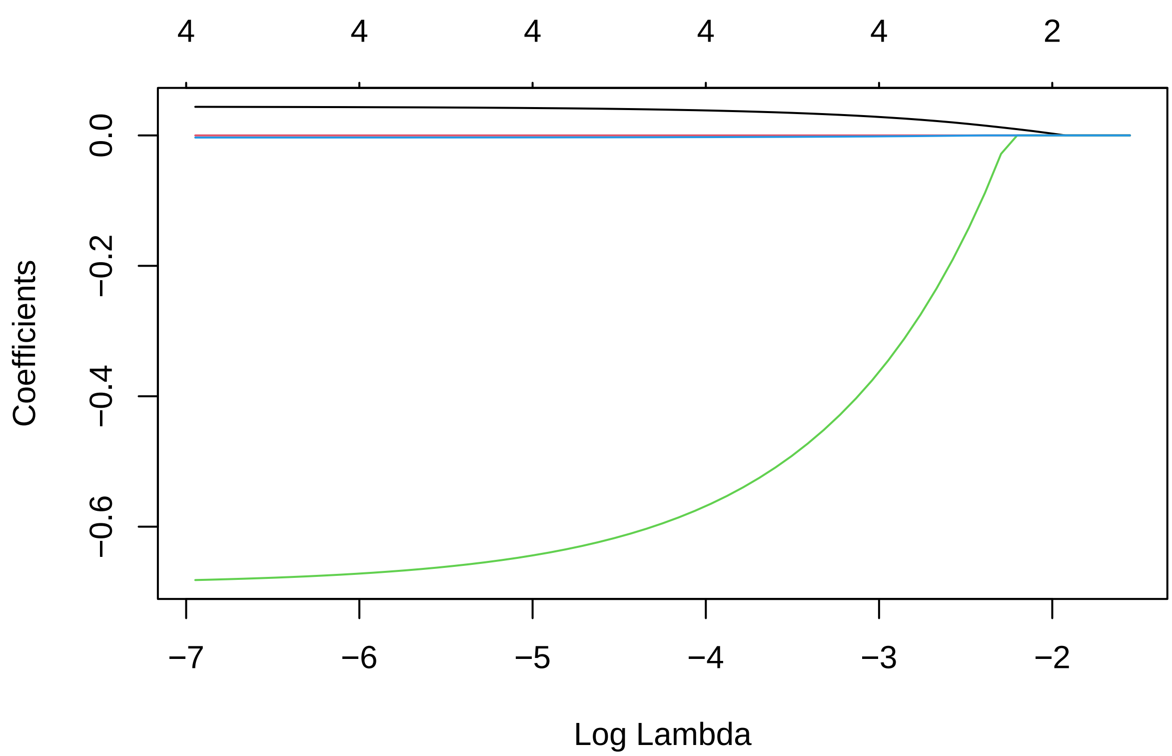
```
## [1] 29.50833
```

- Predicting weeks_on_chart

```
train_lasso_woc <- train_woc %>%
  filter(!is.na(spotify_track_duration_ms), !is.na(danceability), !is.na(key), !is.na(loudness),
         !is.na(energy), !is.na(speechiness), !is.na(mode), !is.na(acousticness),
         !is.na(instrumentalness), !is.na(liveness), !is.na(valence), !is.na(tempo))

fit_lasso_woc <- lm(weeks_on_chart ~ loudness + spotify_track_duration_ms + liveness + tempo,
                    data = train_lasso_woc)
kable(fit_lasso_woc$coefficients)
```

	x
(Intercept)	4.8782361
loudness	0.0441189
spotify_track_duration_ms	0.0000030
liveness	-0.6880831
tempo	-0.0032878

```
X_woc <- model.matrix(fit_lasso_woc)
y_woc <- train_lasso_woc$weeks_on_chart

beta_ols_woc <- solve(crossprod(X_woc)) %*% crossprod(X_woc, y_woc)

lambda_woc <- 1.0
p_woc <- ncol(X_woc)
beta_ridge_woc <- solve(crossprod(X_woc) + diag(lambda_woc, ncol = p_woc, nrow = p_woc)) %*%
  crossprod(X_woc, y_woc)

X_woc <- X_woc[, -1]
fit_lasso_woc <- glmnet(X_woc, y_woc)
plot(fit_lasso_woc, xvar = "lambda")
```




```
X_test_woc <- model.matrix(~ loudness + spotify_track_duration_ms + liveness + tempo, data = test)
X_test_woc <- as.matrix(X_test_woc)
y_test_woc <- test$weeks_on_chart
```

```
y_pred_woc <- X_test_woc %*% beta_ridge_woc

X_test_woc <- X_test_woc[, -1]
pred_lasso_woc <- predict(fit_lasso_woc, newx = X_test_woc, s = 1.0) # s = lambda
rmse_lasso_woc <- sqrt(mean((y_test_woc - pred_lasso_woc)^2))
rmse_lasso_woc
```

```
## [1] 9.145577
```

Code for nearest neighbor

Create a csv file for songs_after_2010_only_audio_features_genre

```
songs_after_2010 %>% filter(!is.na(spotify_genre)) %>% filter(spotify_genre != "[]") -> songs_after_2010
songs_after_2010_only_audio_features_genre[, names(songs_after_2010_only_audio_features_genre) %in%
                                                        c("spotify_genre","key","loudness","mode",
                                                          "speechiness","acousticness",
                                                          "instrumentalness","liveness","valence",
                                                          "time_signature",
                                                          "spotify_track_popularity",
                                                          "danceability","energy","tempo")] %>%
  write.csv("songs_after_2010_only_audio_features_genre.csv")
```

Python

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier

import numpy as np

df = pd.read_csv('songs_after_2010_only_audio_features_genre.csv')

df = df.iloc[: , 1:]

df = df.fillna(0)

def nearest_neighbor(data): X = data.drop('spotify_genre', axis = 1) y = data.spotify_genre

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1)
(X_train.shape, X_test.shape)

model = KNeighborsClassifier(n_neighbors = 9)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = np.mean(np.equal(y_test, y_pred))
return accuracy, y_pred
```

print(nearest_neighbor(df))