```
## 6 7fvUMiyapMsRRxr07cU8Ef Beautiful~ Ed Sheeran                 67 2yiy9cd2QktrN~
## # i 18 more variables: track_album_name <chr>, track_album_release_date <chr>,
## #   playlist_name <chr>, playlist_id <chr>, playlist_genre <chr>,
## #   playlist_subgenre <chr>, danceability <dbl>, energy <dbl>, key <dbl>,
## #   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   duration_ms <dbl>
```

```
# DATA CLEANING
# Extract released year for each observation in the data set
spotify_df <- spotify_songs %>%
  mutate(year_released = str_match(track_album_release_date, '(\\d{4})')[,2])

# Convert playlist_genre to factor and year_released to numeric
spotify_df <- spotify_df %>% mutate(playlist_genre = factor(playlist_genre, ordered = FALSE),
                                    year_released = as.numeric(year_released))

# Check missing values in the data set
inspect_na(spotify_df)
```

```
## # A tibble: 24 x 3
##    col_name                  cnt   pcnt
##    <chr>                   <int>  <dbl>
##  1 track_name                  5 0.0152
##  2 track_artist                5 0.0152
##  3 track_album_name            5 0.0152
##  4 track_id                    0 0
##  5 track_popularity            0 0
##  6 track_album_id              0 0
##  7 track_album_release_date    0 0
##  8 playlist_name               0 0
##  9 playlist_id                 0 0
## 10 playlist_genre              0 0
## # i 14 more rows
```

```
# Drop missing values
spotify_df <- spotify_df %>% drop_na()

# Check valid track's features
inspect_num(spotify_df)
```

```
## # A tibble: 14 x 10
##    col_name      min       q1   median     mean       q3     max      sd pcnt_na
##    <chr>       <dbl>    <dbl>    <dbl>    <dbl>    <dbl>   <dbl>   <dbl>   <dbl>
##  1 track_p~  0        2.4 e+1  4.5 e+1  4.25e+1  6.2 e+1 1   e+2 2.50e+1       0
##  2 danceab~  0        5.63e-1  6.72e-1  6.55e-1  7.61e-1 9.83e-1 1.45e-1       0
##  3 energy    1.75e-4  5.81e-1  7.21e-1  6.99e-1  8.4 e-1 1   e+0 1.81e-1       0
##  4 key       0        2   e+0  6   e+0  5.37e+0  9   e+0 1.1 e+1 3.61e+0       0
##  5 loudness -4.64e+1 -8.17e+0 -6.17e+0 -6.72e+0 -4.65e+0 1.27e+0 2.99e+0       0
##  6 mode      0        0        1   e+0  5.66e-1  1   e+0 1   e+0 4.96e-1       0
##  7 speechi~  0        4.1 e-2  6.25e-2  1.07e-1  1.32e-1 9.18e-1 1.01e-1       0
##  8 acousti~  0        1.51e-2  8.04e-2  1.75e-1  2.55e-1 9.94e-1 2.20e-1       0
##  9 instrum~  0        0        1.61e-5  8.48e-2  4.83e-3 9.94e-1 2.24e-1       0
```

```
## 10 liveness  0        9.27e-2  1.27e-1  1.90e-1  2.48e-1 9.96e-1 1.54e-1       0
## 11 valence   0        3.31e-1  5.12e-1  5.11e-1  6.93e-1 9.91e-1 2.33e-1       0
## 12 tempo     0        1.00e+2  1.22e+2  1.21e+2  1.34e+2 2.39e+2 2.69e+1       0
## 13 duratio~  4   e+3  1.88e+5  2.16e+5  2.26e+5  2.54e+5 5.18e+5 5.98e+4       0
## 14 year_re~  1.96e+3  2.01e+3  2.02e+3  2.01e+3  2.02e+3 2.02e+3 1.14e+1       0
## # i 1 more variable: hist <named list>
```

```
# Check loudness and tempo oftracks
spotify_df %>% filter(!between(loudness, -60, 0))
```

```
## # A tibble: 6 x 24
##   track_id               track_name track_artist track_popularity track_album_id
##   <chr>                  <chr>      <chr>                   <dbl> <chr>
## 1 0jsbEBnXWgHLkvjV49vYVG Nails      Ghostemane                 49 6DIKWvXlVjvAx~
## 2 02fPJUlHDeH46EB1dHrgmZ Crema      Owin                       43 281103RQ7mvsW~
## 3 5DQGkXXLiOhf5cKqIyWh5L Rockstar   Duki                        2 1F7NrR7X4rxJf~
## 4 3BnVqaDfgKyI4CFCfozors Raw Power~ The Stooges                36 6mxbG8KrOTZIx~
## 5 6TaqE6fbzfOGTSnNqmsAMO Escape Fr~ Eva Simons                  0 51Tg4iZyqpqp1~
## 6 2kOmW169C7UV4SZDN9u0YO Vidrado E~ Dj Guuga                    78 5HebljJgeo97M~
## # i 19 more variables: track_album_name <chr>, track_album_release_date <chr>,
## #   playlist_name <chr>, playlist_id <chr>, playlist_genre <fct>,
## #   playlist_subgenre <chr>, danceability <dbl>, energy <dbl>, key <dbl>,
## #   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   duration_ms <dbl>, year_released <dbl>
```

```
spotify_df %>% filter(tempo == 0)
```

```
## # A tibble: 1 x 24
##   track_id               track_name track_artist track_popularity track_album_id
##   <chr>                  <chr>      <chr>                   <dbl> <chr>
## 1 51w6nRCU68klqNfYaaVP2j Hi, How'r~ DREAMS COME~                0 4wdK52JVu5Gzh~
## # i 19 more variables: track_album_name <chr>, track_album_release_date <chr>,
## #   playlist_name <chr>, playlist_id <chr>, playlist_genre <fct>,
## #   playlist_subgenre <chr>, danceability <dbl>, energy <dbl>, key <dbl>,
## #   loudness <dbl>, mode <dbl>, speechiness <dbl>, acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   duration_ms <dbl>, year_released <dbl>
```

```
# Remove observations having invalid loudness and invalid tempo
spotify_df <- spotify_df %>%
  mutate(key = as.integer(key)) %>%
  filter(-60 <= loudness & loudness <= 0) %>%
  filter(tempo > 0)

# Create a table "playlist_spotify" and a table "track_spotify"
playlist_spotify <- spotify_df %>% dplyr::select(contains("playlist"), track_id)
track_spotify <- spotify_df %>% dplyr::select(!contains("playlist"))

# Clean the table "playlist_spotify"
## Check missing values
inspect_na(playlist_spotify)
```

```
## # A tibble: 5 x 3
##   col_name            cnt  pcnt
##   <chr>             <int> <dbl>
## 1 playlist_name         0     0
## 2 playlist_id           0     0
## 3 playlist_genre        0     0
## 4 playlist_subgenre     0     0
## 5 track_id              0     0
```

```
## Check if a playlist_id has one playlist_name
playlist_spotify %>%
  distinct(playlist_id, playlist_name) %>% count(playlist_id) %>% arrange(desc(n))
```

```
## # A tibble: 471 x 2
##    playlist_id                n
##    <chr>                  <int>
##  1 0275i1VNfBnsNbPl0QIBpG     1
##  2 03qQtbNHoJuFezRu2CnLuF     1
##  3 03sDEv7FN58Mb9CJOs1Tgn     1
##  4 06zrBJ5cts5aemZmqe80J7     1
##  5 07SNJ4MwYba9wwmzrbjmYi     1
##  6 07zF8MjQPsiYUXiAIGZ5TA     1
##  7 08QTrfsYYouffgnPjmllAQ     1
##  8 0AFYmoSuoMQiGGjzvBwr6u     1
##  9 0Ar0Ng9DlAWZtSPBvOQgOa     1
## 10 0B2HdP15IucgE0vk3sluJR     1
## # i 461 more rows
```

```
## Check if a track only appear one time in a playlist
playlist_spotify %>% count(playlist_id, track_id) %>% arrange(desc(n))
```

```
## # A tibble: 32,239 x 3
##    playlist_id            track_id                   n
##    <chr>                  <chr>                  <int>
##  1 4JkkvMpVl4lSioqQjeAL0q 070hrICPvpyzXkbqkRC7Ao     3
##  2 4JkkvMpVl4lSioqQjeAL0q 09oZ9eXQ2fo6YDrPzJqAoP     3
##  3 4JkkvMpVl4lSioqQjeAL0q 0BnTBAGmr9FtYwkZrwKhwS     3
##  4 4JkkvMpVl4lSioqQjeAL0q 0CZ8lquoTX2Dkg7Ak2inwA     3
##  5 4JkkvMpVl4lSioqQjeAL0q 0DiDStADDVh3SvAsoJAFMk     3
##  6 4JkkvMpVl4lSioqQjeAL0q 0bMbDctzMmTyK2j74j3nF3     3
##  7 4JkkvMpVl4lSioqQjeAL0q 0qaWEvPkts34WF68r8Dzx9     3
##  8 4JkkvMpVl4lSioqQjeAL0q 0rIAC4PXANcKmitJfoqmVm     3
##  9 4JkkvMpVl4lSioqQjeAL0q 0t3ZvGKlmYmVsDzBJAXK8C     3
## 10 4JkkvMpVl4lSioqQjeAL0q 14sOS5L36385FJ3OL8hew4     3
## # i 32,229 more rows
```

```
## Only take 1 observation of a track in a playlist
playlist_spotify <- playlist_spotify %>% distinct(playlist_id, track_id, .keep_all = TRUE)

## Check if 1 track has 1 track's genre
playlist_spotify %>% count(track_id, playlist_genre) %>% arrange(desc(n))
```

```
## # A tibble: 30,147 x 3
##    track_id               playlist_genre     n
##    <chr>                  <fct>          <int>
##  1 0azC730Exh71aQlOt9Zj3y pop                4
##  2 14sOS5L36385FJ3OL8hew4 pop                4
##  3 1ahVFh0ViDZr8LvkEVlq3B pop                4
##  4 1dVbC0XoM4VxpDkrv1tcjf edm                4
##  5 2V65y3PX4DkRhy1djlxd9p edm                4
##  6 39N9RPD9MRb5WmoLzNzPeA latin              4
##  7 3wSrPtJpnGaUC2hOmJy0BV edm                4
##  8 40riOy7x9W7GXjyGp4pjAv rock               4
##  9 4alHo6RGd0D3OUbTPExTHN rock               4
## 10 5iwz1NiezX7WWjnCgY5TH4 latin              4
## # i 30,137 more rows
```

```
## Create a column for track's genres ('track_genre'), defined by 'playlist_genre'
## (Ex: If a track appear mainly in a rock playlist, it can be more likely to be a rock track)
playlist_spotify <- playlist_spotify %>% group_by(track_id) %>%
  count(track_id, playlist_genre) %>%
  arrange(track_id, desc(n)) %>% slice(1) %>% dplyr::select(-n) %>%
  rename(track_genre = playlist_genre) %>% ungroup()

# Check table "track"
## Only take unique observations (unique tracks)
track_spotify <- track_spotify %>% distinct()

## Check if unique track_id has unique track_name
track_spotify %>% count(track_id, track_name) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               track_name                      n
##    <chr>                  <chr>                       <int>
##  1 0017A6SJgTbfQVU2EtsPNo Pangarap                         1
##  2 002xjHwzEx66OWFV2IP9dk The Others                       1
##  3 004s3t0ONYlzxII9PLgU6z I Feel Alive                     1
##  4 008MceT31RotUANsKuzy3L Liquid Blue                      1
##  5 008rk8F6ZxspZT4bUlkIQG Fever                            1
##  6 00EPIEnX1JFjff8sC6bccd No Me Acuerdo                    1
##  7 00FR9VQ0uzF4NNxVKKiMz2 Full Of Smoke                    1
##  8 00FROhC5g4iJdax5US8jRr Satisfy You                      1
##  9 00GfGwzlSB8DoA0cDP2Eit Tender Lover                     1
## 10 00Gu3RMpDW2vO9PjlMVFDL Hide Away (feat. Envy Monroe)    1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique track_artist
track_spotify %>% count(track_id, track_artist) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               track_artist          n
##    <chr>                  <chr>             <int>
##  1 0017A6SJgTbfQVU2EtsPNo Barbie's Cradle        1
##  2 002xjHwzEx66OWFV2IP9dk RIKA                   1
##  3 004s3t0ONYlzxII9PLgU6z Steady Rollin          1
```

```
##  4 008MceT31RotUANsKuzy3L The.madpix.project     1
##  5 008rk8F6ZxspZT4bUlkIQG YOSA & TAAR            1
##  6 00EPIEnX1JFjff8sC6bccd Thalía                 1
##  7 00FR9VQ0uzF4NNxVKKiMz2 Christión              1
##  8 00FROhC5g4iJdax5US8jRr Diddy                  1
##  9 00GfGwzlSB8DoA0cDP2Eit Babyface               1
## 10 00Gu3RMpDW2vO9PjlMVFDL Blasterjaxx            1
## # i 28,335 more rows
```

```
## Check if unique track_id is in unique track_album_id
track_spotify %>% count(track_id, track_album_id) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               track_album_id             n
##    <chr>                  <chr>                  <int>
##  1 0017A6SJgTbfQVU2EtsPNo 1srJQ0njEQgd8w4XSqI4JQ     1
##  2 002xjHwzEx66OWFV2IP9dk 1ficfUnZMaY1QkNp15Slzm     1
##  3 004s3t0ONYlzxII9PLgU6z 3z04Lb9Dsilqw68SHt6jLB     1
##  4 008MceT31RotUANsKuzy3L 1Z4ANBVuhTlS6DprlP0m1q     1
##  5 008rk8F6ZxspZT4bUlkIQG 2BuYm9UcKvI0ydXs5JKwt0     1
##  6 00EPIEnX1JFjff8sC6bccd 2phs92sMy029JvPDFXUpCC     1
##  7 00FR9VQ0uzF4NNxVKKiMz2 3xpDg9THHn3h4wX1Jyz9TT     1
##  8 00FROhC5g4iJdax5US8jRr 2dHr0LpUe6CNV5lNsr8x0W     1
##  9 00GfGwzlSB8DoA0cDP2Eit 51fAXJ5bMn7DRSunXQ6PMb     1
## 10 00Gu3RMpDW2vO9PjlMVFDL 5pqG85igfoeWcCDIsSi9x7     1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique track_album_release_day
track_spotify %>% count(track_id, track_album_release_date) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               track_album_release_date     n
##    <chr>                  <chr>                    <int>
##  1 0017A6SJgTbfQVU2EtsPNo 2001-01-01                    1
##  2 002xjHwzEx66OWFV2IP9dk 2018-01-26                    1
##  3 004s3t0ONYlzxII9PLgU6z 2017-11-21                    1
##  4 008MceT31RotUANsKuzy3L 2015-08-07                    1
##  5 008rk8F6ZxspZT4bUlkIQG 2018-11-16                    1
##  6 00EPIEnX1JFjff8sC6bccd 2018-06-01                    1
##  7 00FR9VQ0uzF4NNxVKKiMz2 1997-01-01                    1
##  8 00FROhC5g4iJdax5US8jRr 1999-08-24                    1
##  9 00GfGwzlSB8DoA0cDP2Eit 1989-07-07                    1
## 10 00Gu3RMpDW2vO9PjlMVFDL 2019-06-21                    1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique track_popularity
track_spotify %>% count(track_id, track_popularity) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               track_popularity     n
##    <chr>                             <dbl> <int>
##  1 0017A6SJgTbfQVU2EtsPNo               41     1
```

```
##  2 002xjHwzEx66OWFV2IP9dk               15     1
##  3 004s3t0ONYlzxII9PLgU6z               28     1
##  4 008MceT31RotUANsKuzy3L               24     1
##  5 008rk8F6ZxspZT4bUlkIQG               38     1
##  6 00EPIEnX1JFjff8sC6bccd               12     1
##  7 00FR9VQ0uzF4NNxVKKiMz2               41     1
##  8 00FROhC5g4iJdax5US8jRr               52     1
##  9 00GfGwzlSB8DoA0cDP2Eit               36     1
## 10 00Gu3RMpDW2vO9PjlMVFDL               42     1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique track_album_name
track_spotify %>% count(track_id, track_album_name) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               track_album_name                 n
##    <chr>                  <chr>                        <int>
##  1 0017A6SJgTbfQVU2EtsPNo Trip                             1
##  2 002xjHwzEx66OWFV2IP9dk The Others                       1
##  3 004s3t0ONYlzxII9PLgU6z Love & Loss                      1
##  4 008MceT31RotUANsKuzy3L Liquid Blue                      1
##  5 008rk8F6ZxspZT4bUlkIQG Fever                            1
##  6 00EPIEnX1JFjff8sC6bccd No Me Acuerdo                    1
##  7 00FR9VQ0uzF4NNxVKKiMz2 Ghetto Cyrano                    1
##  8 00FROhC5g4iJdax5US8jRr Forever                          1
##  9 00GfGwzlSB8DoA0cDP2Eit Tender Lover                     1
## 10 00Gu3RMpDW2vO9PjlMVFDL Hide Away (feat. Envy Monroe)    1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique danceability
track_spotify %>% count(track_id, danceability) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               danceability     n
##    <chr>                         <dbl> <int>
##  1 0017A6SJgTbfQVU2EtsPNo        0.682     1
##  2 002xjHwzEx66OWFV2IP9dk        0.582     1
##  3 004s3t0ONYlzxII9PLgU6z        0.303     1
##  4 008MceT31RotUANsKuzy3L        0.659     1
##  5 008rk8F6ZxspZT4bUlkIQG        0.662     1
##  6 00EPIEnX1JFjff8sC6bccd        0.836     1
##  7 00FR9VQ0uzF4NNxVKKiMz2        0.389     1
##  8 00FROhC5g4iJdax5US8jRr        0.764     1
##  9 00GfGwzlSB8DoA0cDP2Eit        0.743     1
## 10 00Gu3RMpDW2vO9PjlMVFDL        0.573     1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique track_album_name
track_spotify %>% count(track_id, energy) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               energy     n
```

```
##    <chr>                   <dbl> <int>
##  1 0017A6SJgTbfQVU2EtsPNo  0.401     1
##  2 002xjHwzEx66OWFV2IP9dk  0.704     1
##  3 004s3t0ONYlzxII9PLgU6z  0.88      1
##  4 008MceT31RotUANsKuzy3L  0.794     1
##  5 008rk8F6ZxspZT4bUlkIQG  0.838     1
##  6 00EPIEnX1JFjff8sC6bccd  0.799     1
##  7 00FR9VQ0uzF4NNxVKKiMz2  0.616     1
##  8 00FROhC5g4iJdax5US8jRr  0.594     1
##  9 00GfGwzlSB8DoA0cDP2Eit  0.86      1
## 10 00Gu3RMpDW2vO9PjlMVFDL  0.746     1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique track_album_name
track_spotify %>% count(track_id, key) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id                 key     n
##    <chr>                  <int> <int>
##  1 0017A6SJgTbfQVU2EtsPNo     2     1
##  2 002xjHwzEx66OWFV2IP9dk     5     1
##  3 004s3t0ONYlzxII9PLgU6z     9     1
##  4 008MceT31RotUANsKuzy3L    10     1
##  5 008rk8F6ZxspZT4bUlkIQG     1     1
##  6 00EPIEnX1JFjff8sC6bccd     7     1
##  7 00FR9VQ0uzF4NNxVKKiMz2     1     1
##  8 00FROhC5g4iJdax5US8jRr     6     1
##  9 00GfGwzlSB8DoA0cDP2Eit     5     1
## 10 00Gu3RMpDW2vO9PjlMVFDL    10     1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique loudness
track_spotify %>% count(track_id, loudness) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               loudness     n
##    <chr>                     <dbl> <int>
##  1 0017A6SJgTbfQVU2EtsPNo   -10.1      1
##  2 002xjHwzEx66OWFV2IP9dk    -6.24     1
##  3 004s3t0ONYlzxII9PLgU6z    -4.74     1
##  4 008MceT31RotUANsKuzy3L    -5.64     1
##  5 008rk8F6ZxspZT4bUlkIQG    -6.3      1
##  6 00EPIEnX1JFjff8sC6bccd    -4.25     1
##  7 00FR9VQ0uzF4NNxVKKiMz2    -8.75     1
##  8 00FROhC5g4iJdax5US8jRr   -10.0      1
##  9 00GfGwzlSB8DoA0cDP2Eit    -6.35     1
## 10 00Gu3RMpDW2vO9PjlMVFDL    -4.89     1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique mode
track_spotify %>% count(track_id, mode) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id                 mode     n
##    <chr>                   <dbl> <int>
##  1 0017A6SJgTbfQVU2EtsPNo     1     1
##  2 002xjHwzEx66OWFV2IP9dk     1     1
##  3 004s3t0ONYlzxII9PLgU6z     1     1
##  4 008MceT31RotUANsKuzy3L     0     1
##  5 008rk8F6ZxspZT4bUlkIQG     1     1
##  6 00EPIEnX1JFjff8sC6bccd     0     1
##  7 00FR9VQ0uzF4NNxVKKiMz2     0     1
##  8 00FROhC5g4iJdax5US8jRr     1     1
##  9 00GfGwzlSB8DoA0cDP2Eit     1     1
## 10 00Gu3RMpDW2vO9PjlMVFDL     1     1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique speechiness
track_spotify %>% count(track_id, speechiness) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               speechiness     n
##    <chr>                        <dbl> <int>
##  1 0017A6SJgTbfQVU2EtsPNo      0.0236     1
##  2 002xjHwzEx66OWFV2IP9dk      0.0347     1
##  3 004s3t0ONYlzxII9PLgU6z      0.0442     1
##  4 008MceT31RotUANsKuzy3L      0.054      1
##  5 008rk8F6ZxspZT4bUlkIQG      0.0499     1
##  6 00EPIEnX1JFjff8sC6bccd      0.0873     1
##  7 00FR9VQ0uzF4NNxVKKiMz2      0.284      1
##  8 00FROhC5g4iJdax5US8jRr      0.185      1
##  9 00GfGwzlSB8DoA0cDP2Eit      0.0445     1
## 10 00Gu3RMpDW2vO9PjlMVFDL      0.0421     1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique acousticness
track_spotify %>% count(track_id, acousticness) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               acousticness     n
##    <chr>                         <dbl> <int>
##  1 0017A6SJgTbfQVU2EtsPNo     0.279        1
##  2 002xjHwzEx66OWFV2IP9dk     0.0651       1
##  3 004s3t0ONYlzxII9PLgU6z     0.0117       1
##  4 008MceT31RotUANsKuzy3L     0.000761     1
##  5 008rk8F6ZxspZT4bUlkIQG     0.114        1
##  6 00EPIEnX1JFjff8sC6bccd     0.187        1
##  7 00FR9VQ0uzF4NNxVKKiMz2     0.453        1
##  8 00FROhC5g4iJdax5US8jRr     0.591        1
##  9 00GfGwzlSB8DoA0cDP2Eit     0.226        1
## 10 00Gu3RMpDW2vO9PjlMVFDL     0.0249       1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique instrumentalness
track_spotify %>% count(track_id, instrumentalness) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               instrumentalness     n
##    <chr>                             <dbl> <int>
##  1 0017A6SJgTbfQVU2EtsPNo         0.0117       1
##  2 002xjHwzEx66OWFV2IP9dk         0            1
##  3 004s3t0ONYlzxII9PLgU6z         0.00994      1
##  4 008MceT31RotUANsKuzy3L         0.132        1
##  5 008rk8F6ZxspZT4bUlkIQG         0.000697     1
##  6 00EPIEnX1JFjff8sC6bccd         0            1
##  7 00FR9VQ0uzF4NNxVKKiMz2         0            1
##  8 00FROhC5g4iJdax5US8jRr         0            1
##  9 00GfGwzlSB8DoA0cDP2Eit         0.000422     1
## 10 00Gu3RMpDW2vO9PjlMVFDL         0            1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique liveness
track_spotify %>% count(track_id, liveness) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               liveness     n
##    <chr>                     <dbl> <int>
##  1 0017A6SJgTbfQVU2EtsPNo   0.0887     1
##  2 002xjHwzEx66OWFV2IP9dk   0.212      1
##  3 004s3t0ONYlzxII9PLgU6z   0.347      1
##  4 008MceT31RotUANsKuzy3L   0.322      1
##  5 008rk8F6ZxspZT4bUlkIQG   0.0881     1
##  6 00EPIEnX1JFjff8sC6bccd   0.092      1
##  7 00FR9VQ0uzF4NNxVKKiMz2   0.916      1
##  8 00FROhC5g4iJdax5US8jRr   0.145      1
##  9 00GfGwzlSB8DoA0cDP2Eit   0.0513     1
## 10 00Gu3RMpDW2vO9PjlMVFDL   0.361      1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique valence
track_spotify %>% count(track_id, valence) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               valence     n
##    <chr>                    <dbl> <int>
##  1 0017A6SJgTbfQVU2EtsPNo   0.566     1
##  2 002xjHwzEx66OWFV2IP9dk   0.698     1
##  3 004s3t0ONYlzxII9PLgU6z   0.404     1
##  4 008MceT31RotUANsKuzy3L   0.852     1
##  5 008rk8F6ZxspZT4bUlkIQG   0.496     1
##  6 00EPIEnX1JFjff8sC6bccd   0.772     1
##  7 00FR9VQ0uzF4NNxVKKiMz2   0.716     1
##  8 00FROhC5g4iJdax5US8jRr   0.695     1
##  9 00GfGwzlSB8DoA0cDP2Eit   0.687     1
## 10 00Gu3RMpDW2vO9PjlMVFDL   0.134     1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique tempo
track_spotify %>% count(track_id, tempo) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               tempo     n
##    <chr>                  <dbl> <int>
##  1 0017A6SJgTbfQVU2EtsPNo  97.1     1
##  2 002xjHwzEx66OWFV2IP9dk 151.      1
##  3 004s3t0ONYlzxII9PLgU6z 135.      1
##  4 008MceT31RotUANsKuzy3L 128.      1
##  5 008rk8F6ZxspZT4bUlkIQG 130.      1
##  6 00EPIEnX1JFjff8sC6bccd  94.0     1
##  7 00FR9VQ0uzF4NNxVKKiMz2 145.      1
##  8 00FROhC5g4iJdax5US8jRr  87.3     1
##  9 00GfGwzlSB8DoA0cDP2Eit 102.      1
## 10 00Gu3RMpDW2vO9PjlMVFDL 130.      1
## # i 28,335 more rows
```

```
## Check if unique track_id has unique duration_ms
track_spotify %>% count(track_id, duration_ms) %>% arrange(desc(n))
```

```
## # A tibble: 28,345 x 3
##    track_id               duration_ms     n
##    <chr>                        <dbl> <int>
##  1 0017A6SJgTbfQVU2EtsPNo      235440     1
##  2 002xjHwzEx66OWFV2IP9dk      197286     1
##  3 004s3t0ONYlzxII9PLgU6z      373512     1
##  4 008MceT31RotUANsKuzy3L      228565     1
##  5 008rk8F6ZxspZT4bUlkIQG      236308     1
##  6 00EPIEnX1JFjff8sC6bccd      217653     1
##  7 00FR9VQ0uzF4NNxVKKiMz2      289227     1
##  8 00FROhC5g4iJdax5US8jRr      286441     1
##  9 00GfGwzlSB8DoA0cDP2Eit      259267     1
## 10 00Gu3RMpDW2vO9PjlMVFDL      188000     1
## # i 28,335 more rows
```

```
# Join table "track_spotify" and table "playlist_spotify" by track_id, to take track's features
spotify_final <- playlist_spotify %>% inner_join(track_spotify, by = "track_id")

# Remove all columns with name (track_name, track_album_name)
# because these name can be given arbitrarily so they may not have relationship with genre
spotify_final <- spotify_final %>% dplyr::select(!(contains("name")))

# Remove track_release_year because variable "year_released" is already created
spotify_final <- spotify_final %>% dplyr::select(-track_album_release_date)

# Check the number of track's genre
print(spotify_final %>% count(track_genre)%>% arrange(n), n = 10)
```

```
## # A tibble: 6 x 2
##   track_genre     n
##   <fct>       <int>
```

```
## 1 rock         4298
## 2 latin        4385
## 3 pop          4580
## 4 r&b          4676
## 5 rap          5078
## 6 edm          5328
```

```
# Set seed for reproducibility
set.seed(1872398)

# Reduce the data set, only take 1,000 tracks per genre
spotify_1000 <- spotify_final %>%
  group_by(track_genre) %>% sample_n(1000) %>% ungroup()
head(spotify_1000, n = 6)
```

```
## # A tibble: 6 x 18
##   track_id track_genre track_artist track_popularity track_album_id danceability
##   <chr>    <fct>       <chr>                  <dbl> <chr>                  <dbl>
## 1 5kUxWpe~ edm         alt-J                     36 4cyGkmKOzaMyi~         0.569
## 2 3MkdA6v~ edm         Matuê                     79 6dXKy9X9o0Hpc~         0.417
## 3 5HBV5At~ edm         Rob Stepwart              15 246E5OvV4QXhP~         0.762
## 4 2hkZW3V~ edm         Deorro                    18 487y7fvivuT9r~         0.681
## 5 2wuYCAL~ edm         Joachim Gar~               0 23REtQMYdDesB~         0.638
## 6 1KcqsSi~ edm         Mike Hawkins               7 47OI2E1EVs6Kv~         0.644
## # i 12 more variables: energy <dbl>, key <int>, loudness <dbl>, mode <dbl>,
## #   speechiness <dbl>, acousticness <dbl>, instrumentalness <dbl>,
## #   liveness <dbl>, valence <dbl>, tempo <dbl>, duration_ms <dbl>,
## #   year_released <dbl>
```

```
# Clean variable track_artist
## Create a table for track's artists including track_id, track_artist and track_genre
spotify_1000_artist_origin <- spotify_1000 %>%
  dplyr::select(track_id, track_artist, track_genre) %>% distinct()

## Check artists having 10 tracks or more
spotify_1000_artist_origin %>% count(track_artist) %>%
  count(n) %>%
  mutate(prop = round(nn*100/ sum(nn),3)) %>%
  filter(n >= 10)
```

```
## # A tibble: 8 x 3
##       n    nn  prop
##   <int> <int> <dbl>
## 1    10     7 0.187
## 2    11     7 0.187
## 3    12     3 0.08
## 4    13     5 0.133
## 5    14     1 0.027
## 6    16     3 0.08
## 7    17     2 0.053
## 8    34     1 0.027
```

```r
## Create a pivot table for track_artist
spotify_1000_artist <- spotify_1000_artist_origin %>%
  add_count(track_id, track_artist, track_genre) %>%
  arrange(desc(n)) %>%
  pivot_wider(names_from = track_artist, values_from = n, values_fill = list(n=0)) %>%
  clean_names()

## Take a list of track's artists having 10 tracks or more and clean artist's names
track_artist_morethan10 <- spotify_1000 %>% count(track_artist) %>%
  filter(n >= 10) %>% pull(track_artist)

track_artist_morethan10 <- make_clean_names(track_artist_morethan10)

## Only take track's artists having 10 tracks or more, do not include variable "track_artist"
spotify_1000_artist <- spotify_1000_artist %>%
  dplyr::select(track_id, track_genre, all_of(track_artist_morethan10))

# Join table "spotify_1000" and table "spotify_1000_artist" together by variable "track_id"
spotify_1000 <- spotify_1000 %>%
  inner_join(spotify_1000_artist %>% dplyr::select(-track_genre), by = c("track_id"))

# Remove all columns with ID (track_id, track_album_id) and track_artist
spotify_1000 <- spotify_1000 %>% dplyr::select(-track_id, -track_album_id, -track_artist)


# EXPLORATORY DATA
# Summary statistics of predictors related to track's features
```

```r
## Summary statistics of track_popularity
spotify_1000 %>% ggplot(aes(x = track_popularity)) + geom_histogram() + ylab("Frequency")
```

```r
skewness(spotify_1000$track_popularity)
```

```
## [1] -0.253588
```

```r
round(summary(spotify_1000$track_popularity),3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   21.00   42.00   39.47   58.00   98.00
```

```r
round(sd(spotify_1000$track_popularity),3)
```

```
## [1] 23.673
```

```r
quantile_v <- round(quantile(spotify_1000$track_popularity),3)
quantile_v
```

```
##   0%  25%  50%  75% 100%
##    0   21   42   58   98
```

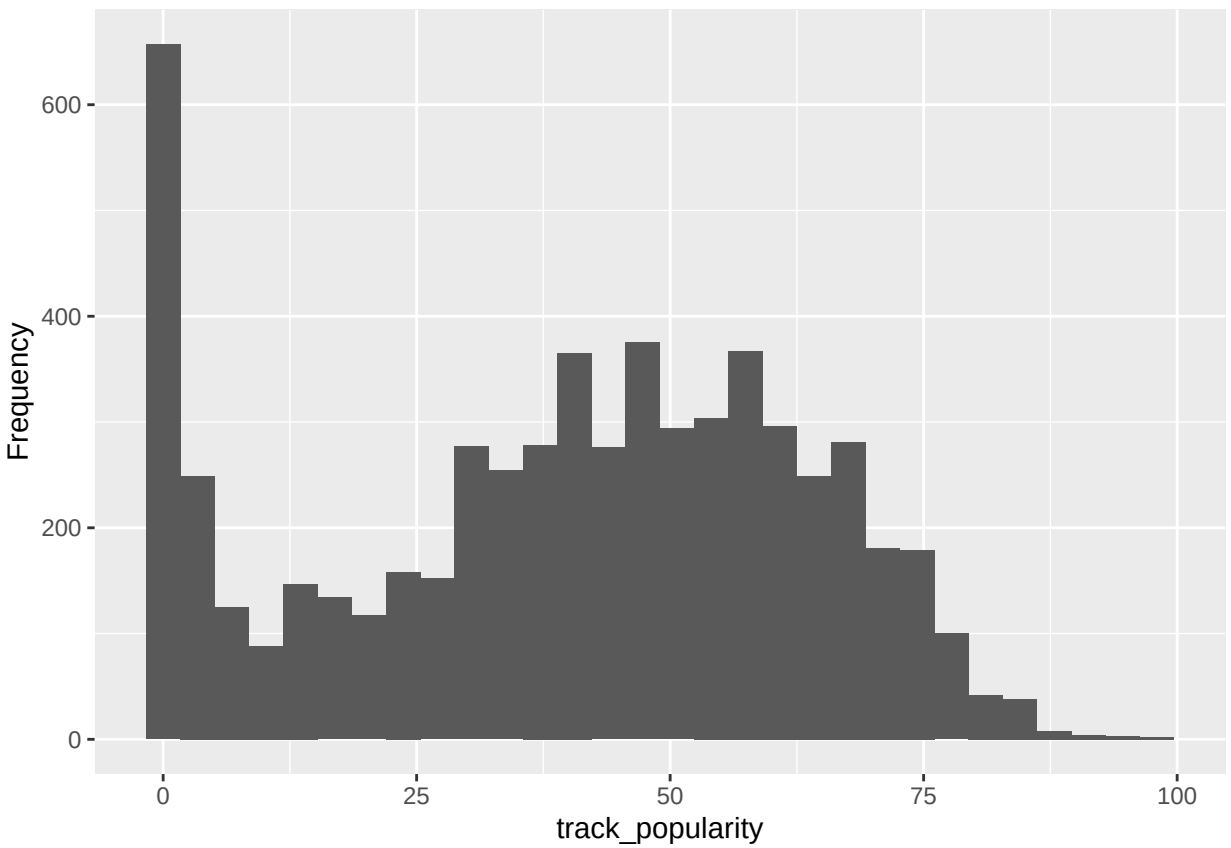


Figure 1: Histogram of track's popularity

```
q_1 <- round(as.numeric(quantile_v[2]),3)
q_3 <- round(as.numeric(quantile_v[4]),3)
iqr_v <- q_3 - q_1
iqr_v
```

```
## [1] 37
```

```
spotify_1000 %>%
  filter(!(between(track_popularity, q_1 - 1.5*iqr_v, q_3 + 1.5*iqr_v))) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1     0
```

```
## Summary statistics of danceability
spotify_1000 %>% ggplot(aes(x = danceability)) + geom_histogram() + ylab("Frequency")
```

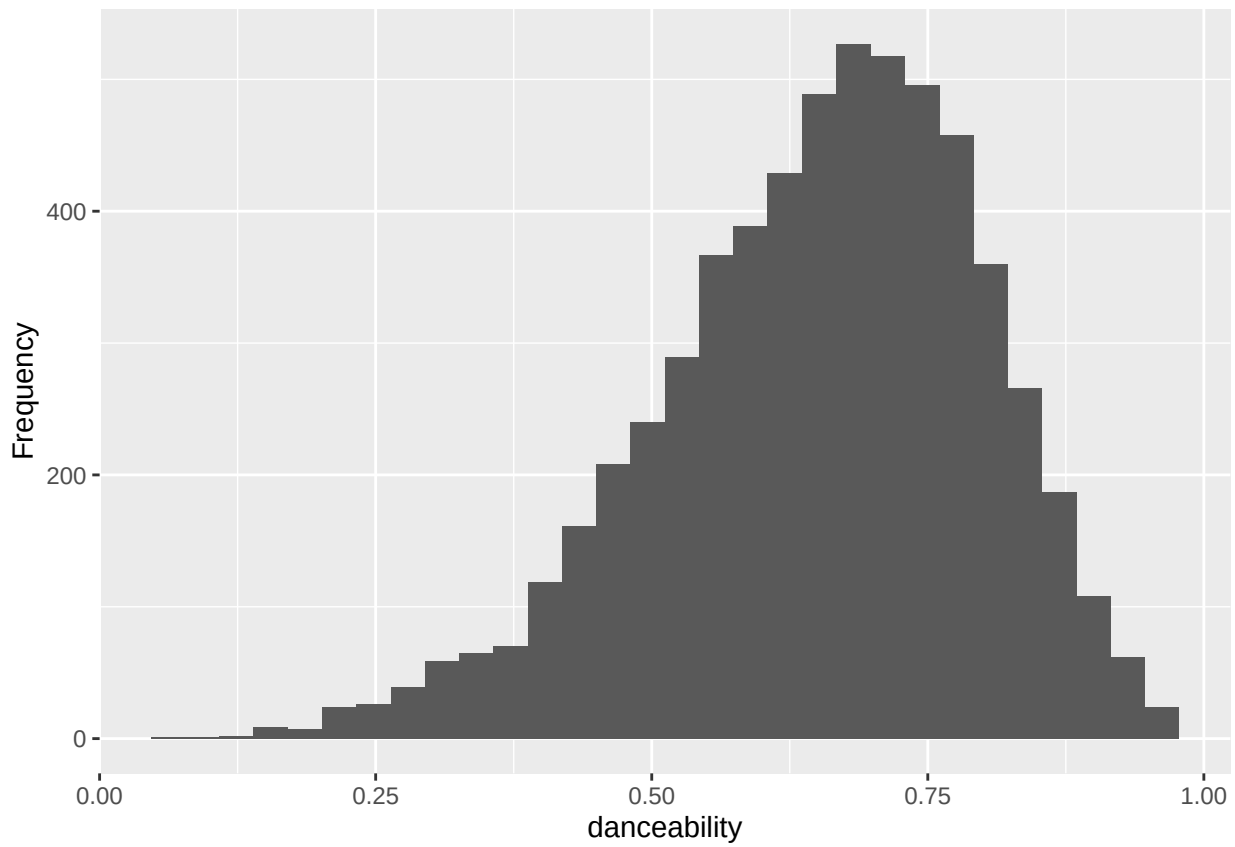


Figure 2: Histogram of track's danceability

```
skewness(spotify_1000$danceability)
```

```
## [1] -0.542421
```

```r
round(summary(spotify_1000$danceability),3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.077   0.559   0.668   0.651   0.758   0.977
```

```r
round(sd(spotify_1000$danceability),3)
```

```
## [1] 0.147
```

```r
quantile_v <- round(quantile(spotify_1000$danceability),3)
quantile_v
```

```
##    0%   25%   50%   75%  100%
## 0.077 0.559 0.668 0.758 0.977
```

```r
q_1 <- round(as.numeric(quantile_v[2]),3)
q_3 <- round(as.numeric(quantile_v[4]),3)
iqr_v <- q_3 - q_1
iqr_v
```

```
## [1] 0.199
```

```r
spotify_1000 %>%
  filter(!(between(danceability, q_1 - 1.5*iqr_v, q_3 + 1.5*iqr_v))) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    64
```

```r
## Summary statistics of energy
spotify_1000 %>% ggplot(aes(x = energy)) + geom_histogram() + ylab("Frequency")
```

```r
skewness(spotify_1000$energy)
```

```
## [1] -0.6599889
```

```r
round(summary(spotify_1000$energy),3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.582   0.724   0.701   0.844   1.000
```

```r
round(sd(spotify_1000$energy),3)
```

```
## [1] 0.182
```

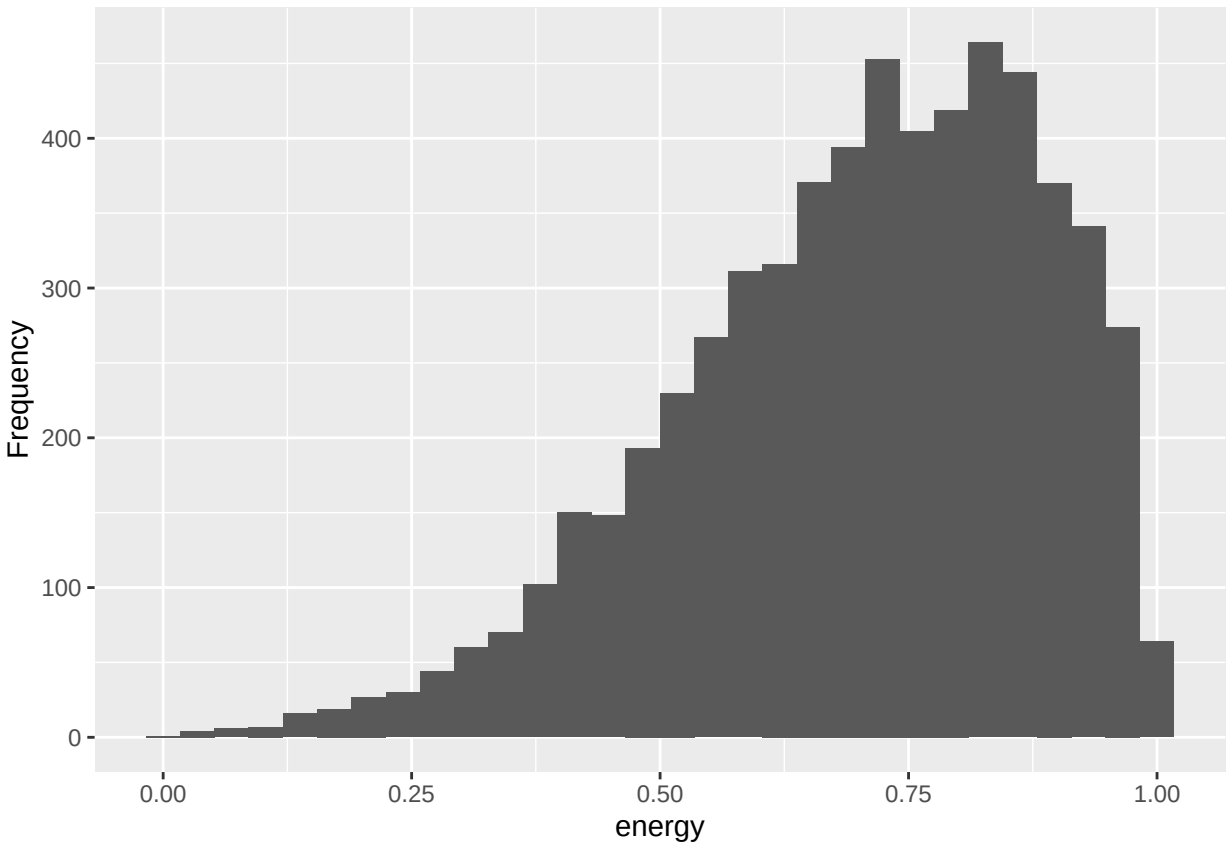


Figure 3: Histogram of track's energy

```r
quantile_v <- round(quantile(spotify_1000$energy),3)
quantile_v
```

```
##    0%   25%   50%   75%  100%
## 0.000 0.582 0.724 0.844 1.000
```

```r
q_1 <- round(as.numeric(quantile_v[2]),3)
q_3 <- round(as.numeric(quantile_v[4]),3)
iqr_v <- q_3 - q_1
iqr_v
```

```
## [1] 0.262
```

```r
spotify_1000 %>%
  filter(!(between(energy, q_1 - 1.5*iqr_v, q_3 + 1.5*iqr_v))) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    52
```

```r
## Summary statistics of key
spotify_1000 %>% ggplot(aes(x = key)) + geom_bar() + ylab("Frequency")
```

```r
spotify_1000 %>%
  count(key) %>% arrange(desc(n)) %>% mutate(prop = round(100*n/sum(n),3))
```

```
## # A tibble: 12 x 3
##      key     n  prop
##    <int> <int> <dbl>
##  1     1   708 11.8
##  2     7   644 10.7
##  3     0   617 10.3
##  4     9   549  9.15
##  5    11   537  8.95
##  6     2   516  8.6
##  7     6   485  8.08
##  8     5   480  8
##  9     4   434  7.23
## 10     8   433  7.22
## 11    10   418  6.97
## 12     3   179  2.98
```

```r
## Summary statistics of loudness
spotify_1000 %>% ggplot(aes(x = loudness)) + geom_histogram() + ylab("Frequency")
```

```r
skewness(spotify_1000$loudness)
```

```
## [1] -1.616112
```

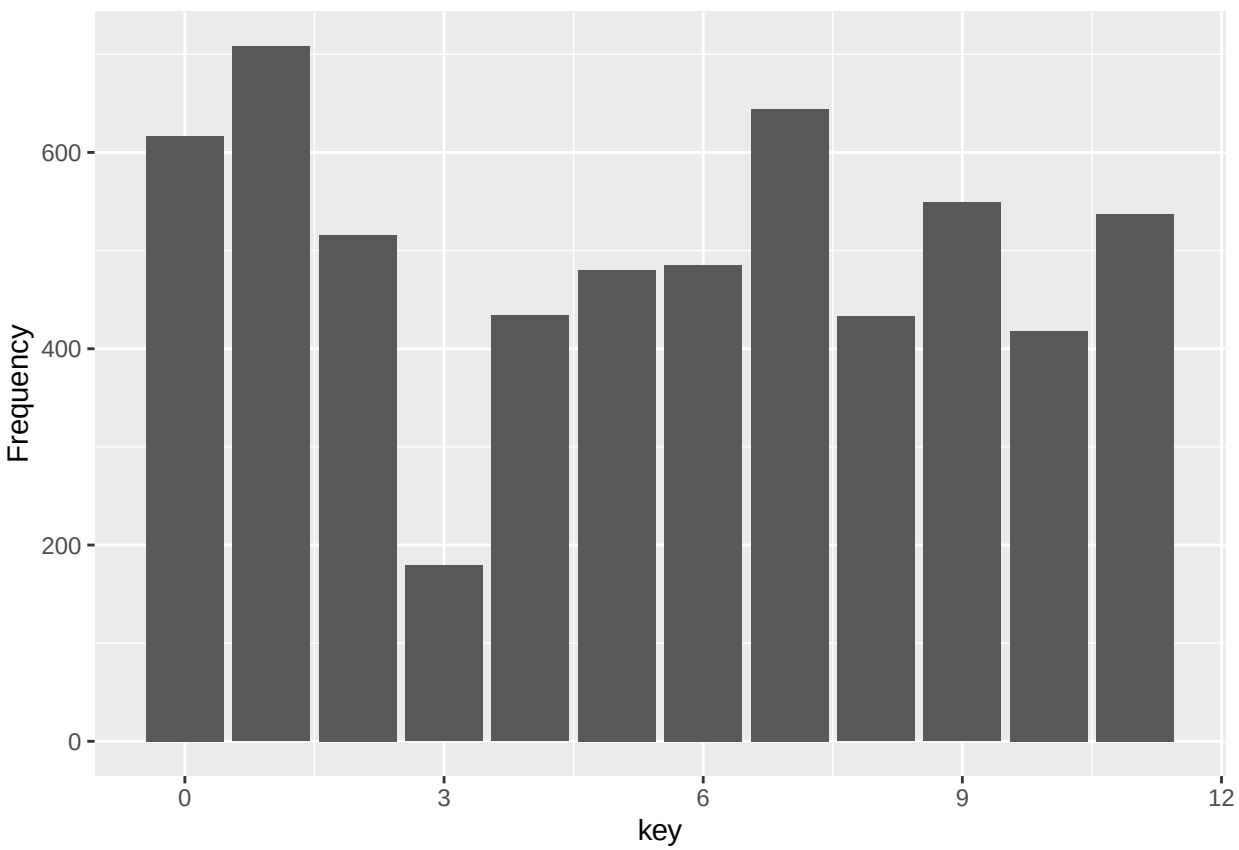


Figure 4: Bar chart of track's key

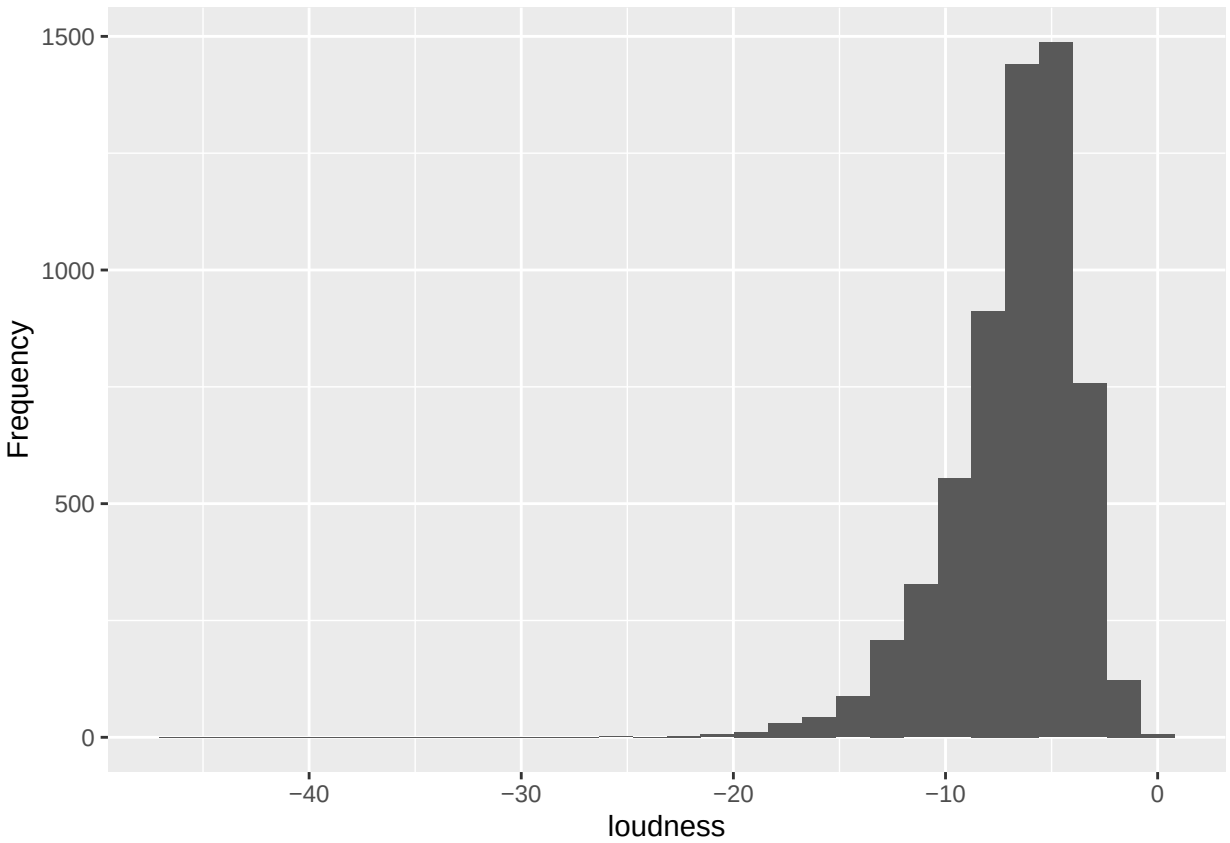


Figure 5: Histogram of track's loudness

```r
round(summary(spotify_1000$loudness),3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -46.448  -8.265  -6.231  -6.803  -4.642  -0.158
```

```r
round(sd(spotify_1000$loudness),3)
```

```
## [1] 3.087
```

```r
quantile_v <- round(quantile(spotify_1000$loudness),3)
quantile_v
```

```
##      0%     25%     50%     75%    100%
## -46.448  -8.265  -6.231  -4.642  -0.158
```

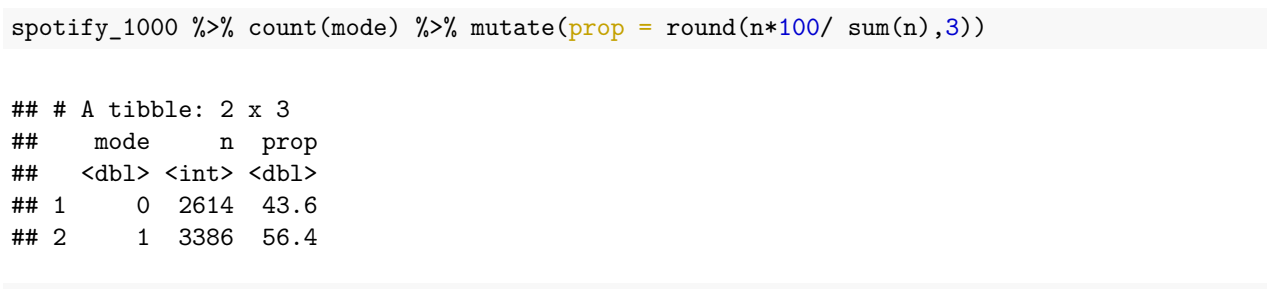
```r
q_1 <- round(as.numeric(quantile_v[2]),3)
q_3 <- round(as.numeric(quantile_v[4]),3)
iqr_v <- q_3 - q_1
iqr_v
```

```
## [1] 3.623
```

```r
spotify_1000 %>%
  filter(!(between(loudness, q_1 - 1.5*iqr_v, q_3 + 1.5*iqr_v))) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   173
```

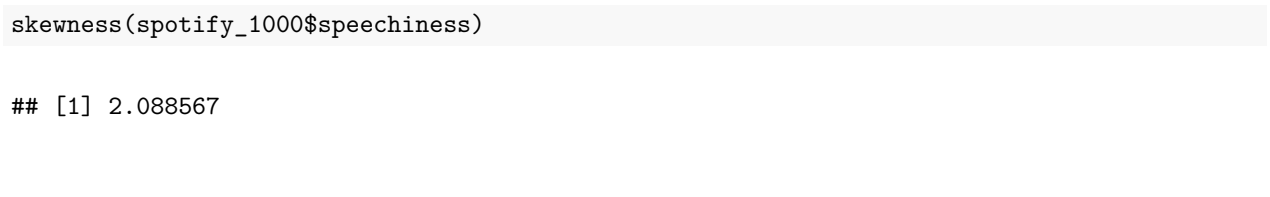
```r
## Summary statistics of mode
spotify_1000 %>% ggplot(aes(x = mode)) + geom_bar() + ylab("Frequency")
```



```r
spotify_1000 %>% count(mode) %>% mutate(prop = round(n*100/ sum(n),3))
```

```
## # A tibble: 2 x 3
##    mode     n  prop
##   <dbl> <int> <dbl>
## 1     0  2614  43.6
## 2     1  3386  56.4
```

```r
## Summary statistics of speechiness
spotify_1000 %>% ggplot(aes(x = speechiness)) + geom_histogram() + ylab("Frequency")
```



```r
skewness(spotify_1000$speechiness)
```

```
## [1] 2.088567
```

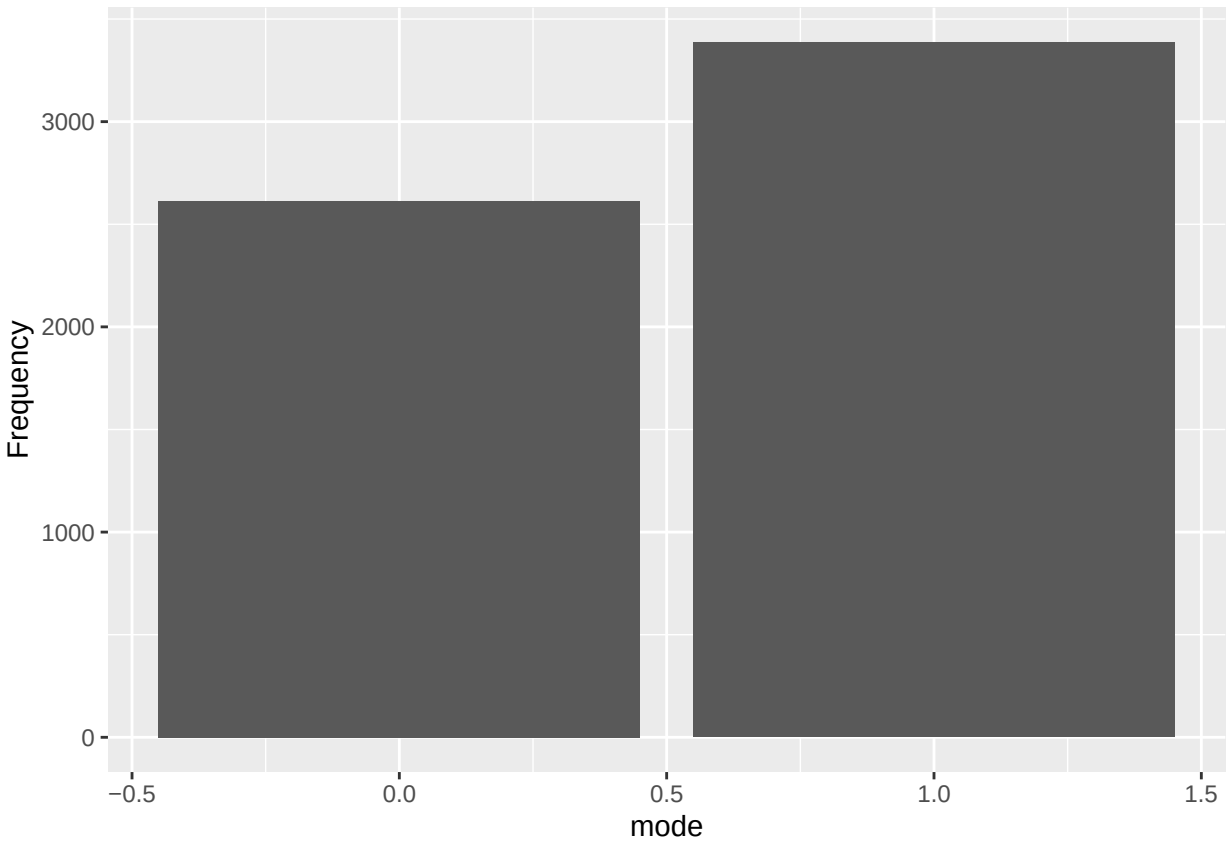


Figure 6: Bar chart of track's mode

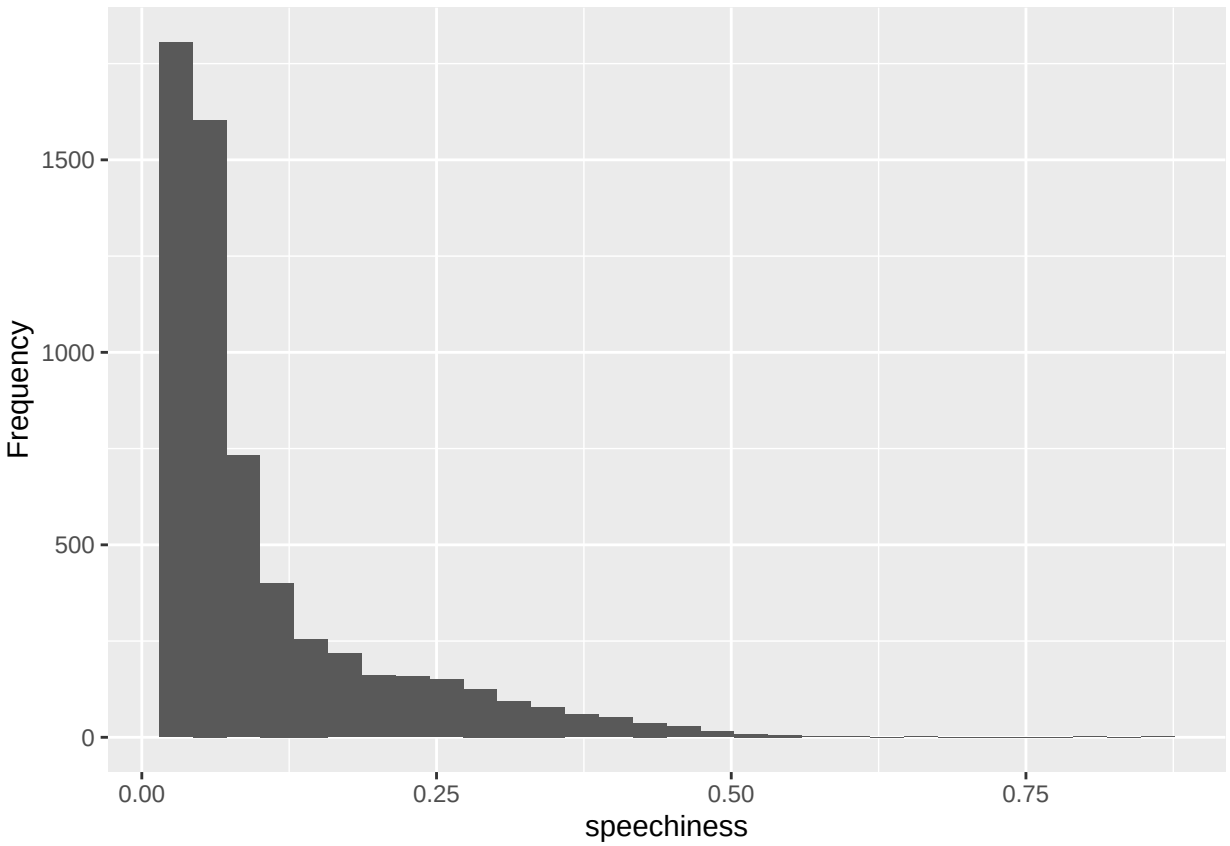


Figure 7: Histogram of track's speechiness

```r
round(summary(spotify_1000$speechiness),3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.023   0.040   0.061   0.104   0.126   0.856
```

```r
quantile_v <- round(quantile(spotify_1000$speechiness),3)
quantile_v
```

```
##    0%   25%   50%   75%  100%
## 0.023 0.040 0.061 0.126 0.856
```

```r
q_1 <- round(as.numeric(quantile_v[2]),3)
q_3 <- round(as.numeric(quantile_v[4]),3)
iqr_v <- q_3 - q_1
iqr_v
```

```
## [1] 0.086
```

```r
spotify_1000 %>%
  filter(!(between(speechiness, q_1 - 1.5*iqr_v, q_3 + 1.5*iqr_v))) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   596
```

```r
## Summary statistics of acousticness
spotify_1000 %>% ggplot(aes(x = acousticness)) + geom_histogram() + ylab("Frequency")
```

```r
skewness(spotify_1000$acousticness)
```

```
## [1] 1.584317
```

```r
round(summary(spotify_1000$acousticness),3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.015   0.079   0.177   0.259   0.992
```

```r
quantile_v <- round(quantile(spotify_1000$acousticness),3)
quantile_v
```

```
##    0%   25%   50%   75%  100%
## 0.000 0.015 0.079 0.259 0.992
```

```r
q_1 <- round(as.numeric(quantile_v[2]),3)
q_3 <- round(as.numeric(quantile_v[4]),3)
iqr_v <- q_3 - q_1
iqr_v
```

```
## [1] 0.244
```

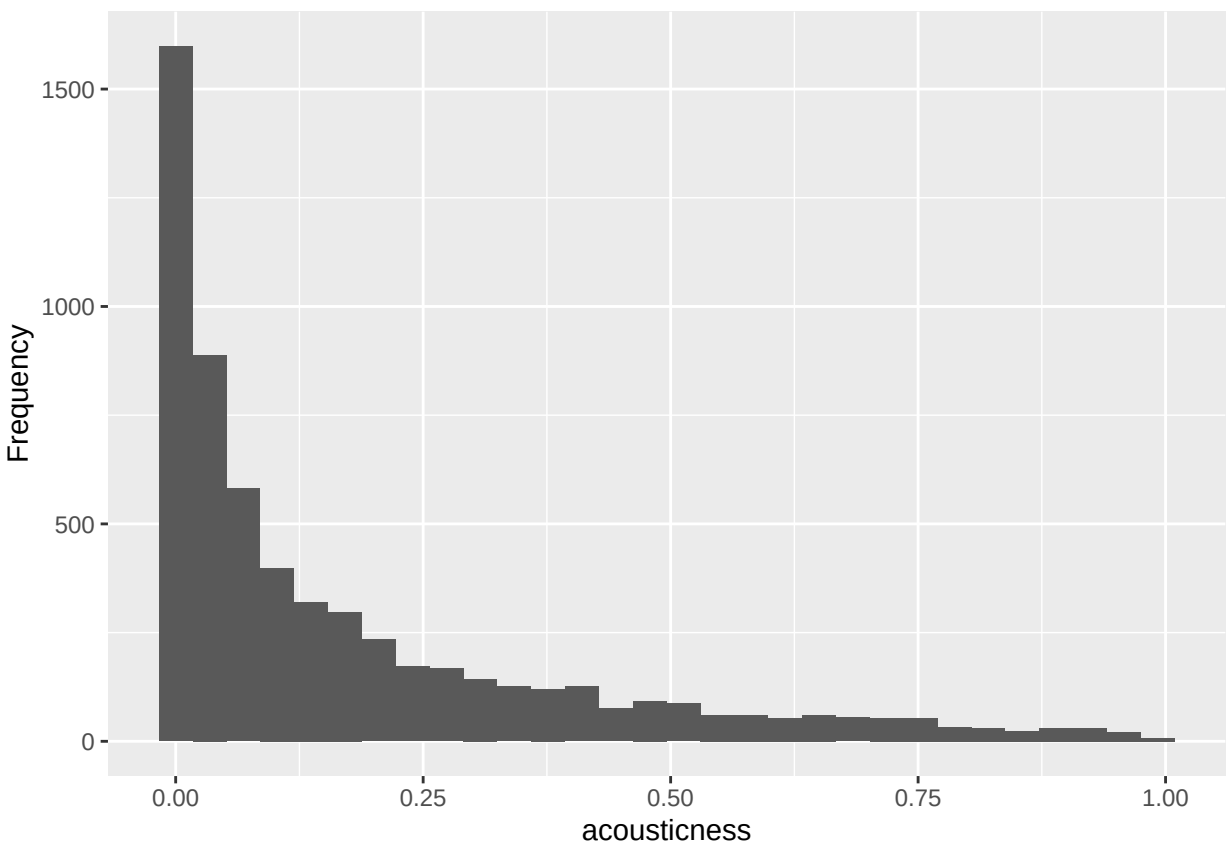


Figure 8: Histogram of track's acousticness

```
spotify_1000 %>%
  filter(!(between(acousticness, q_1 - 1.5*iqr_v, q_3 + 1.5*iqr_v))) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   408
```

```
## Summary statistics of instrumentalness
spotify_1000 %>% ggplot(aes(x = instrumentalness)) +
  geom_histogram() + ylab("Frequency")
```

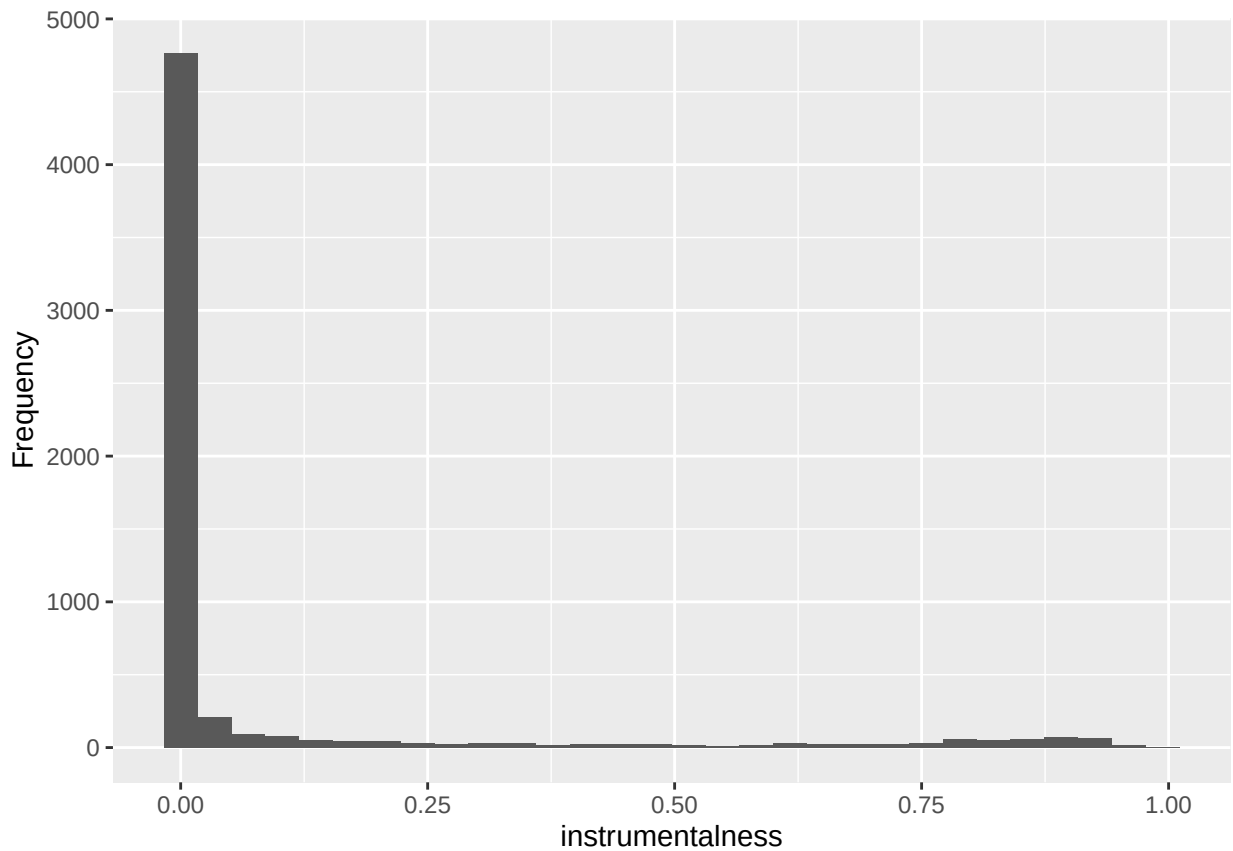


Figure 9: Histogram of track's instrumentalness

```
skewness(spotify_1000$instrumentalness)
```

```
## [1] 2.742894
```

```
round(summary(spotify_1000$instrumentalness),3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   0.086   0.005   0.994
```

```r
quantile_v <- round(quantile(spotify_1000$instrumentalness),3)
quantile_v
```

```
##    0%   25%   50%   75%  100%
## 0.000 0.000 0.000 0.005 0.994
```

```r
q_1 <- round(as.numeric(quantile_v[2]),3)
q_3 <- round(as.numeric(quantile_v[4]),3)
iqr_v <- q_3 - q_1
iqr_v
```

```
## [1] 0.005
```

```r
spotify_1000 %>%
  filter(!(between(instrumentalness, q_1 - 1.5*iqr_v, q_3 + 1.5*iqr_v))) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1  1306
```

```r
## Summary statistics of liveness
spotify_1000 %>% ggplot(aes(x = liveness)) + geom_histogram() + ylab("Frequency")
```



```r
skewness(spotify_1000$liveness)
```

```
## [1] 2.145734
```

```r
round(summary(spotify_1000$liveness),3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.009   0.093   0.127   0.192   0.246   0.991
```

```r
quantile_v <- round(quantile(spotify_1000$liveness),3)
quantile_v
```

```
##    0%   25%   50%   75%  100%
## 0.009 0.093 0.127 0.246 0.991
```

```r
q_1 <- round(as.numeric(quantile_v[2]),3)
q_3 <- round(as.numeric(quantile_v[4]),3)
iqr_v <- q_3 - q_1
iqr_v
```

```
## [1] 0.153
```

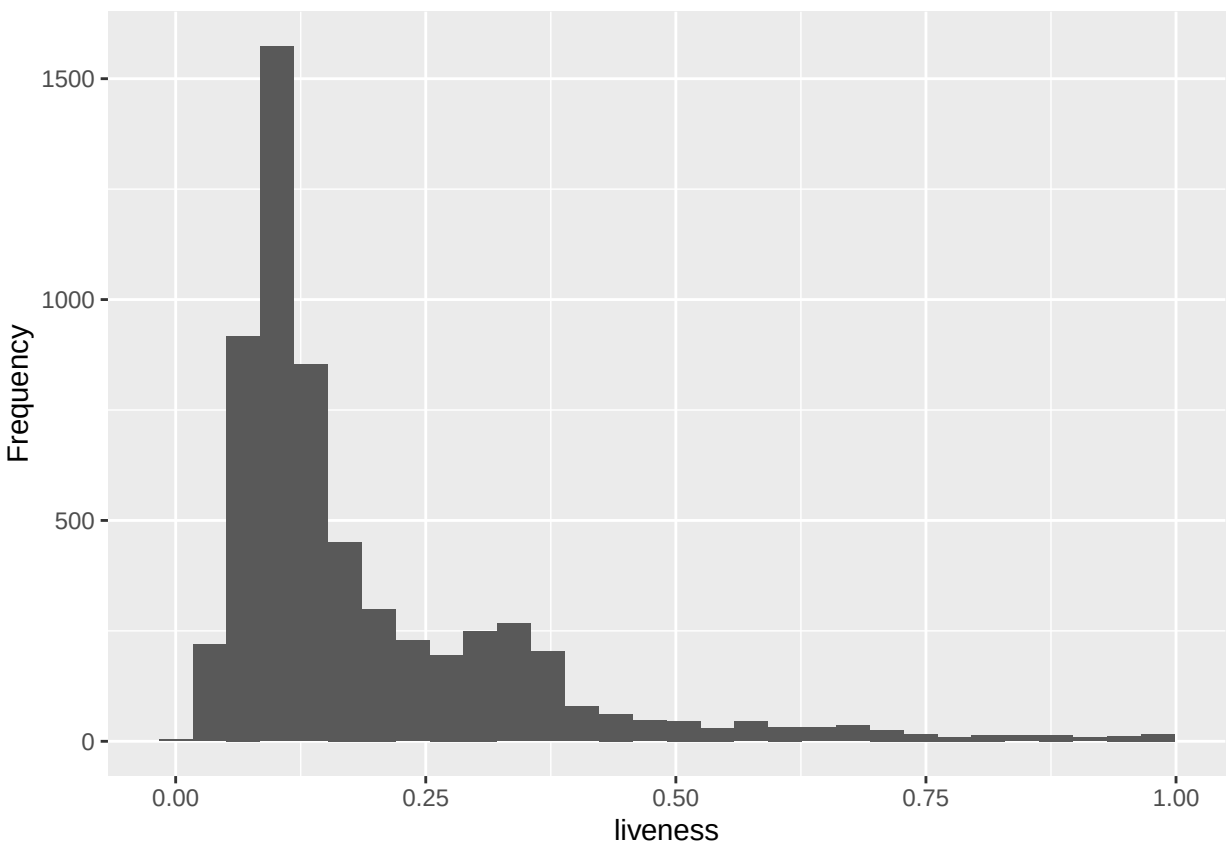


Figure 10: Histogram of track's liveness

```
spotify_1000 %>%
  filter(!(between(liveness, q_1 - 1.5*iqr_v, q_3 + 1.5*iqr_v))) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   369
```

```
## Summary statistics of valence
spotify_1000 %>% ggplot(aes(x = valence)) + geom_histogram() + ylab("Frequency")
```

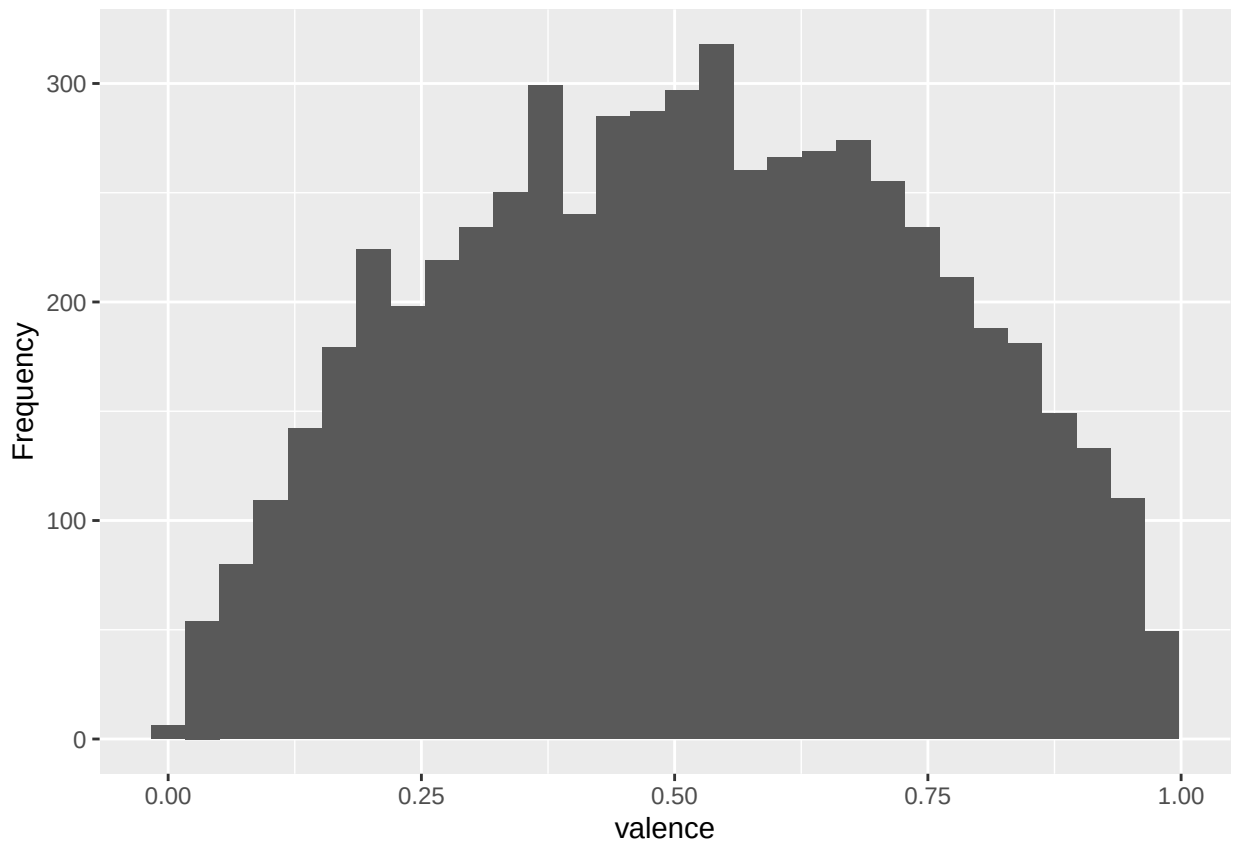


Figure 11: Histogram of track's valence

```
skewness(spotify_1000$valence)
```

```
## [1] -0.01470599
```

```
round(summary(spotify_1000$valence),3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.330   0.513   0.511   0.694   0.981
```

```r
round(sd(spotify_1000$valence),3)
```

```
## [1] 0.234
```

```r
quantile_v <- round(quantile(spotify_1000$valence),3)
quantile_v
```

```
##    0%   25%   50%   75%  100%
## 0.000 0.330 0.513 0.694 0.981
```

```r
q_1 <- round(as.numeric(quantile_v[2]),3)
q_3 <- round(as.numeric(quantile_v[4]),3)
iqr_v <- q_3 - q_1
iqr_v
```

```
## [1] 0.364
```

```r
spotify_1000 %>%
  filter(!(between(valence, q_1 - 1.5*iqr_v, q_3 + 1.5*iqr_v))) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1     0
```

```r
## Summary statistics of tempo
spotify_1000 %>% ggplot(aes(x = tempo)) + geom_histogram() + ylab("Frequency")
```



```r
skewness(spotify_1000$tempo)
```

```
## [1] 0.527669
```

```r
round(summary(spotify_1000$tempo),3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   52.02   99.97  121.93  121.13  134.98  211.64
```

```r
round(sd(spotify_1000$tempo),3)
```

```
## [1] 26.961
```

```r
quantile_v <- round(quantile(spotify_1000$tempo),3)
quantile_v
```

```
##      0%     25%     50%     75%    100%
##  52.017  99.975 121.929 134.978 211.644
```

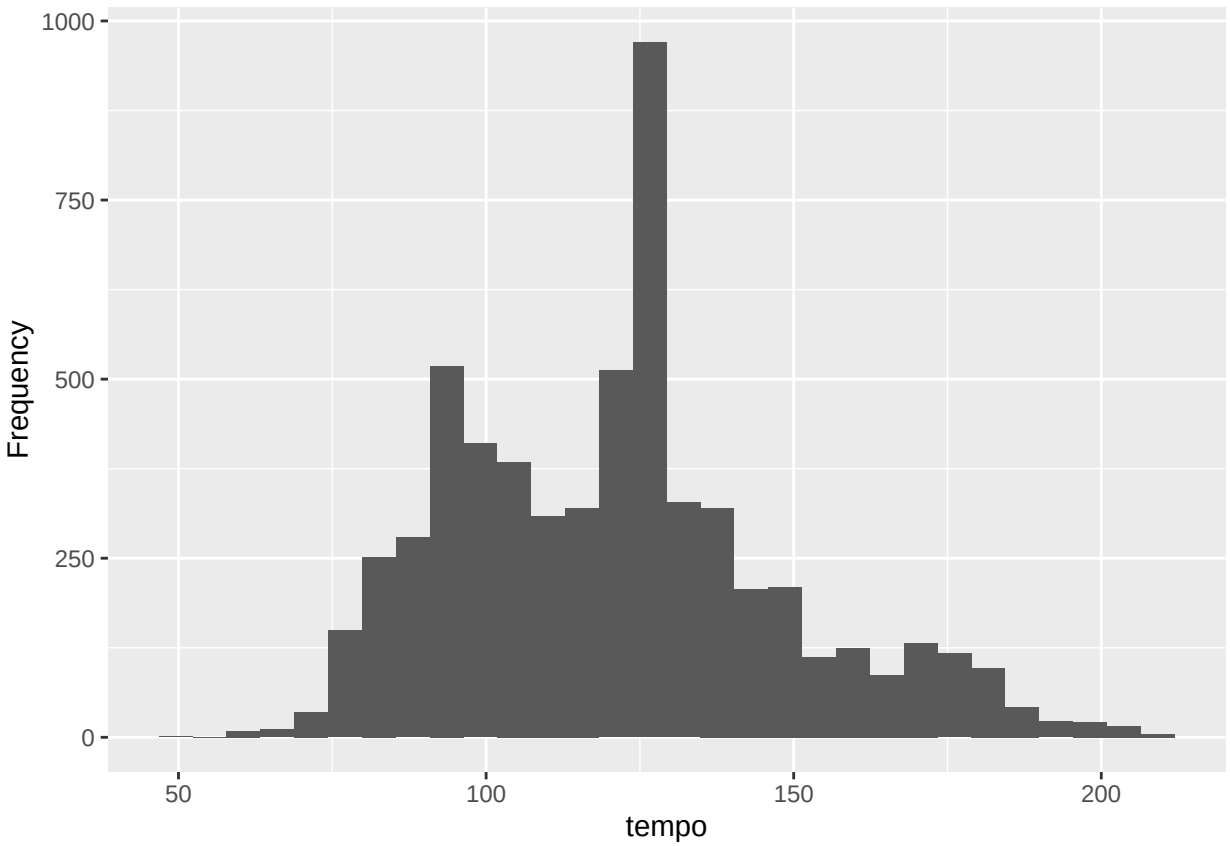


Figure 12: Histogram of track's tempo

```
q_1 <- round(as.numeric(quantile_v[2]),3)
q_3 <- round(as.numeric(quantile_v[4]),3)
iqr_v <- q_3 - q_1
iqr_v
```

```
## [1] 35.003
```

```
spotify_1000 %>%
  filter(!(between(tempo, q_1 - 1.5*iqr_v, q_3 + 1.5*iqr_v))) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    82
```

```
## Summary statistics of duration_ms
spotify_1000 %>% ggplot(aes(x = duration_ms)) + geom_histogram() + ylab("Frequency")
```

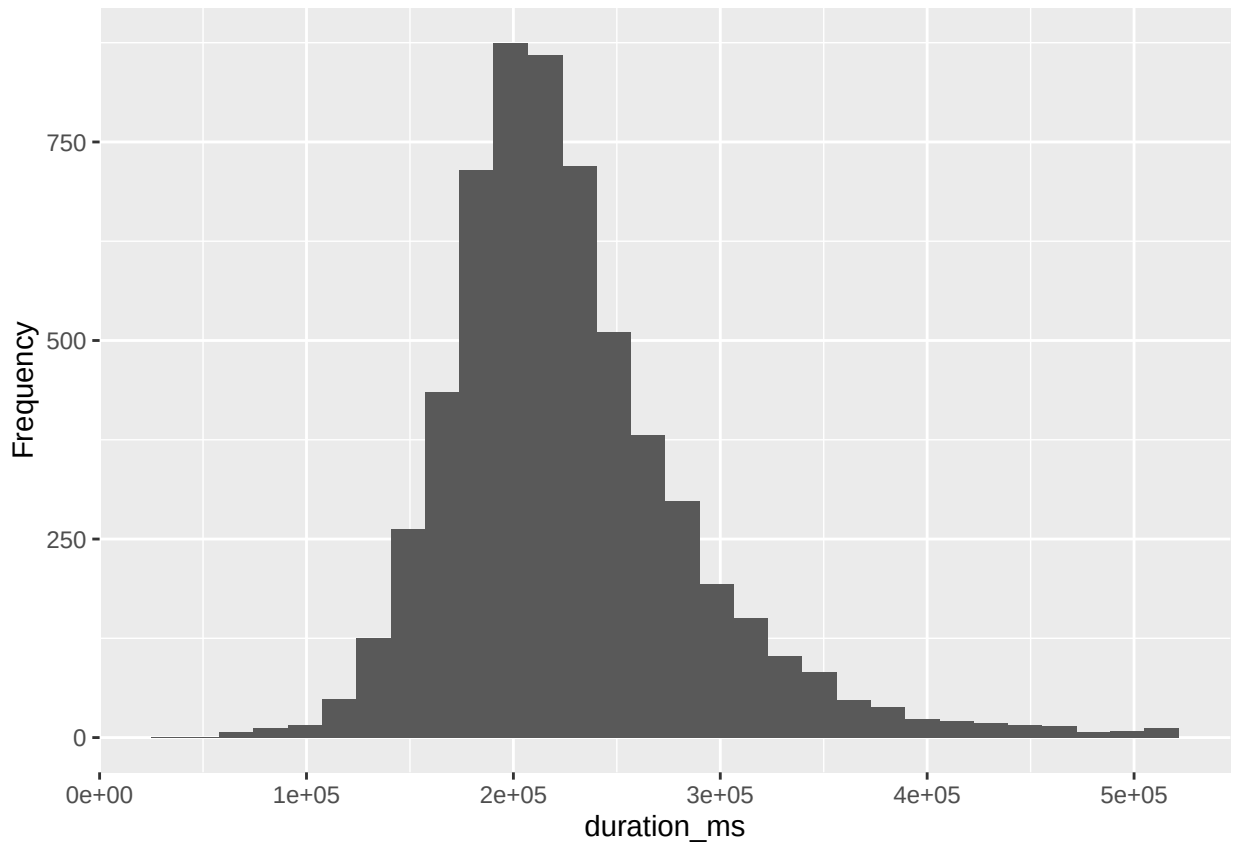


Figure 13: Histogram of track's duration_ms

```
skewness(spotify_1000$duration_ms)
```

```
## [1] 1.203942
```

```
round(summary(spotify_1000$duration_ms),3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   37500  188087  216700  226294  254010  517810
```

```
round(sd(spotify_1000$duration_ms),3)
```

```
## [1] 59775.64
```

```
quantile_v <- round(quantile(spotify_1000$duration_ms),3)
quantile_v
```

```
##       0%      25%      50%      75%     100%
##  37500.0 188087.2 216700.0 254010.2 517810.0
```

```
q_1 <- round(as.numeric(quantile_v[2]),3)
q_3 <- round(as.numeric(quantile_v[4]),3)
iqr_v <- q_3 - q_1
iqr_v
```

```
## [1] 65923
```

```
spotify_1000 %>%
  filter(!(between(duration_ms, q_1 - 1.5*iqr_v, q_3 + 1.5*iqr_v))) %>% count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   239
```

```
## Summary statistics of year_released
spotify_1000 %>%
  count(year_released) %>% mutate(prop = round(n*100/sum(n),3)) %>% arrange(desc(prop))
```

```
## # A tibble: 61 x 3
##    year_released     n  prop
##            <dbl> <int> <dbl>
##  1          2019  1529 25.5
##  2          2018   596  9.93
##  3          2017   475  7.92
##  4          2016   370  6.17
##  5          2015   332  5.53
##  6          2014   258  4.3
##  7          2013   183  3.05
##  8          2012   154  2.57
##  9          2008   135  2.25
## 10          2011   125  2.08
## # i 51 more rows
```

```
# Relationship between predictors and track's genres
## Relationship between track_popularity and track's genres
spotify_1000 %>%
  ggplot(aes(x = fct_reorder(track_genre, track_popularity), y = track_popularity)) +
  geom_boxplot() + xlab("Track_genre")
```

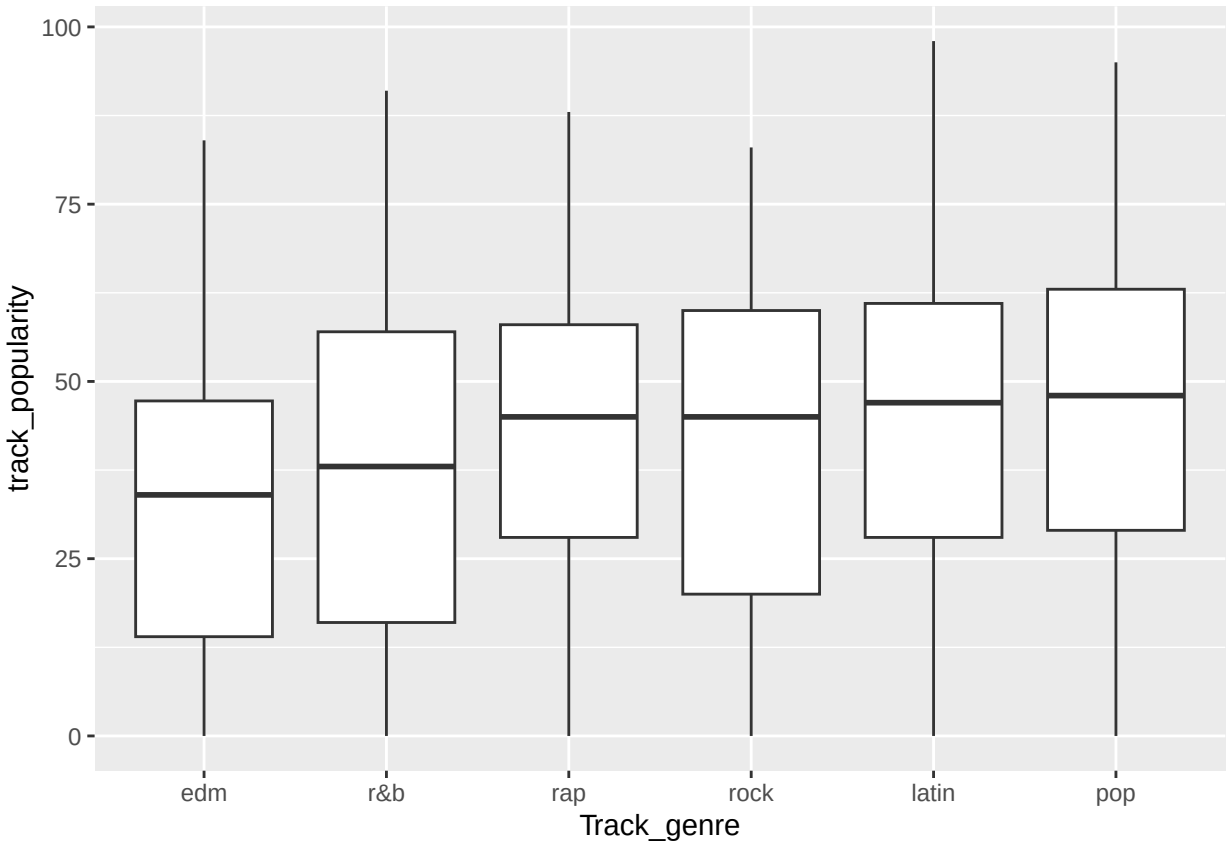


Figure 14: Boxplot between track_popularity and track_genre

```
spotify_1000 %>% group_by(track_genre) %>%
  summarise(mean_v = median(track_popularity)) %>% arrange(mean_v)
```

```
## # A tibble: 6 x 2
##   track_genre mean_v
##   <fct>        <dbl>
## 1 edm             34
## 2 r&b             38
## 3 rap             45
## 4 rock            45
## 5 latin           47
## 6 pop             48
```

```
## Relationship between danceability and track's genres
spotify_1000 %>%
```

```
  ggplot(aes(x = fct_reorder(track_genre, danceability), y = danceability)) +
  geom_boxplot() + xlab("Track_genre")
```

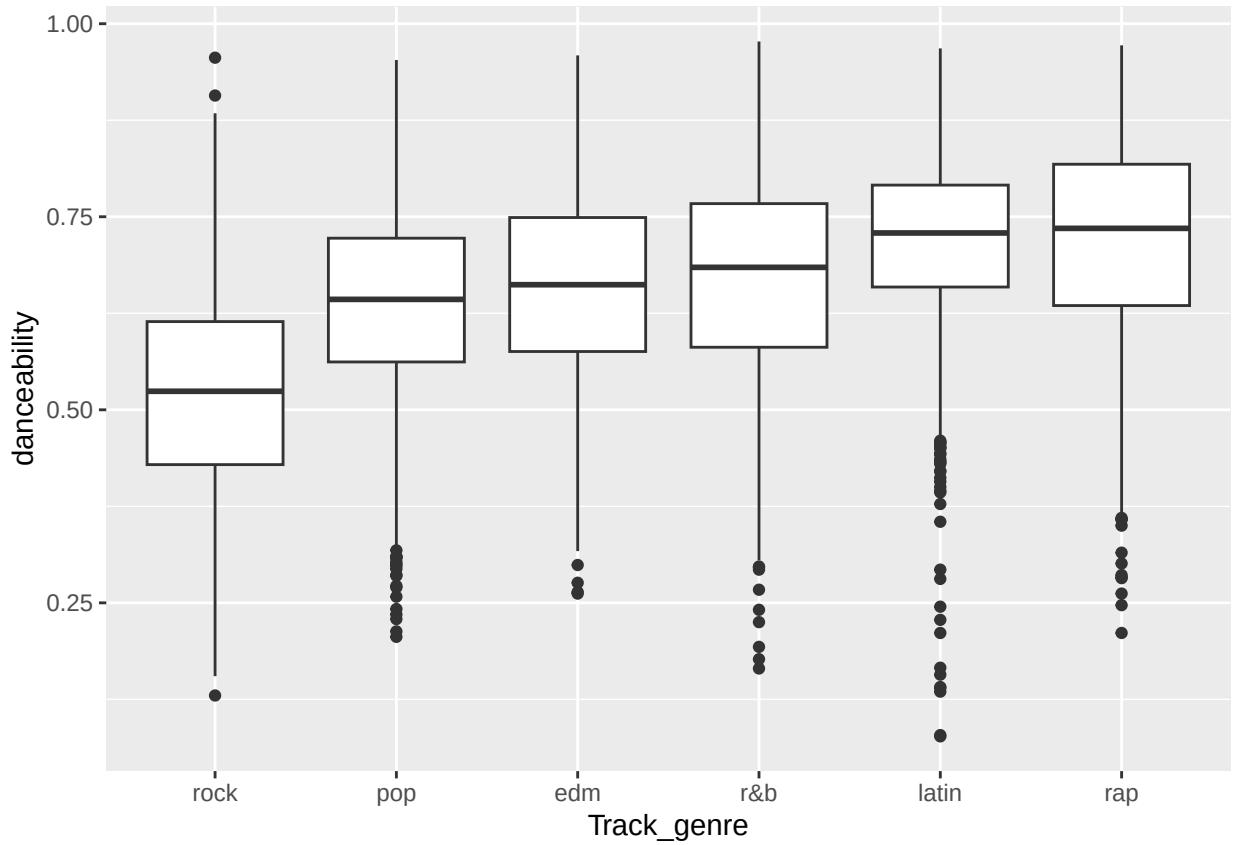


Figure 15: Boxplot between danceability and track_genre

```
spotify_1000 %>%
  group_by(track_genre) %>%
  summarise(mean_v = median(danceability)) %>% arrange(mean_v)
```

```
## # A tibble: 6 x 2
##   track_genre mean_v
##   <fct>        <dbl>
## 1 rock         0.524
## 2 pop          0.643
## 3 edm          0.662
## 4 r&b          0.685
## 5 latin        0.729
## 6 rap          0.735
```

```
## Relationship between energy and track's genres
spotify_1000 %>% ggplot(aes(x = fct_reorder(track_genre, energy), y = energy)) +
  geom_boxplot() + xlab("Track_genre")
```

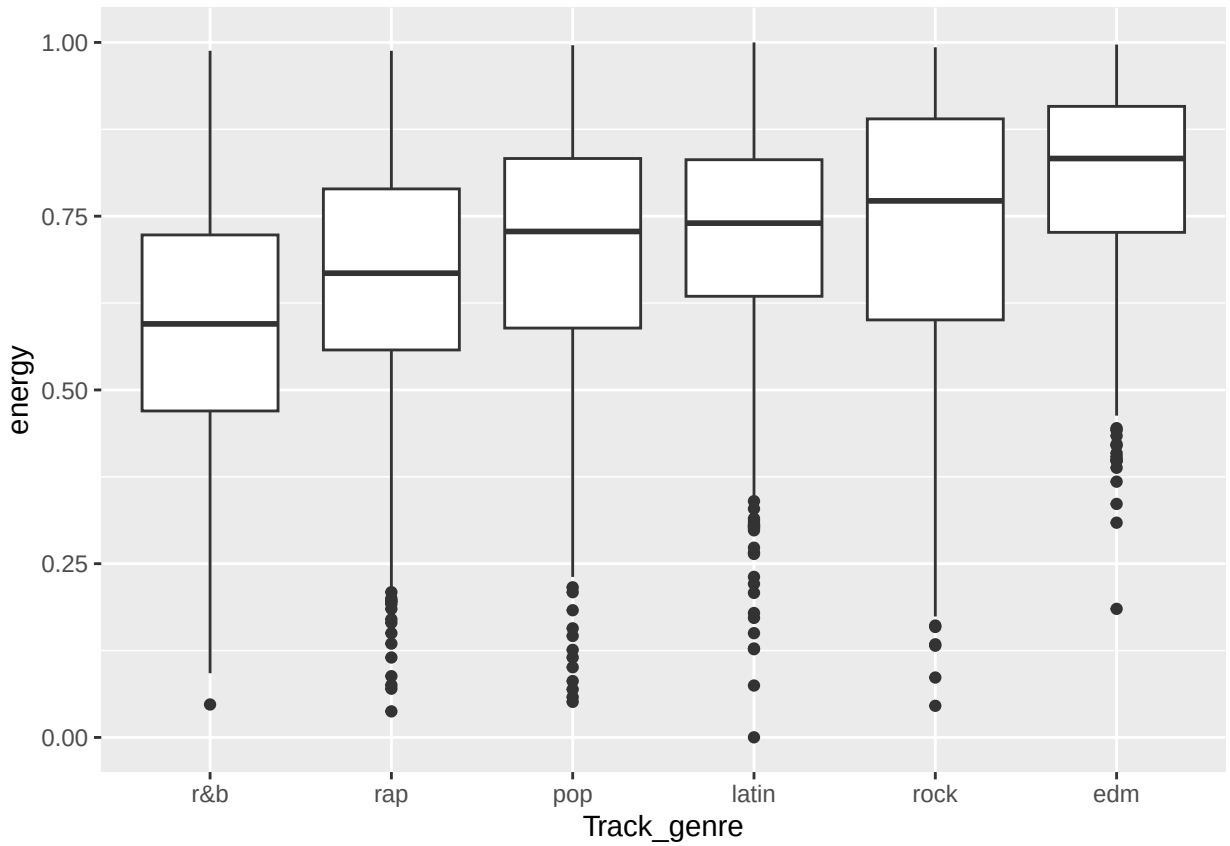


Figure 16: Boxplot between energy and track_genre

```
spotify_1000 %>%
  group_by(track_genre) %>% summarise(mean_v = median(energy)) %>% arrange(mean_v)
```

```
## # A tibble: 6 x 2
##   track_genre mean_v
##   <fct>        <dbl>
## 1 r&b          0.595
## 2 rap          0.668
## 3 pop          0.728
## 4 latin        0.74
## 5 rock         0.772
## 6 edm          0.833
```

```
## Relationship between key and track's genres
spotify_1000 %>% ggplot(aes(x = fct_reorder(track_genre, key), y = key)) +
  geom_boxplot() + xlab("Track_genre")
```

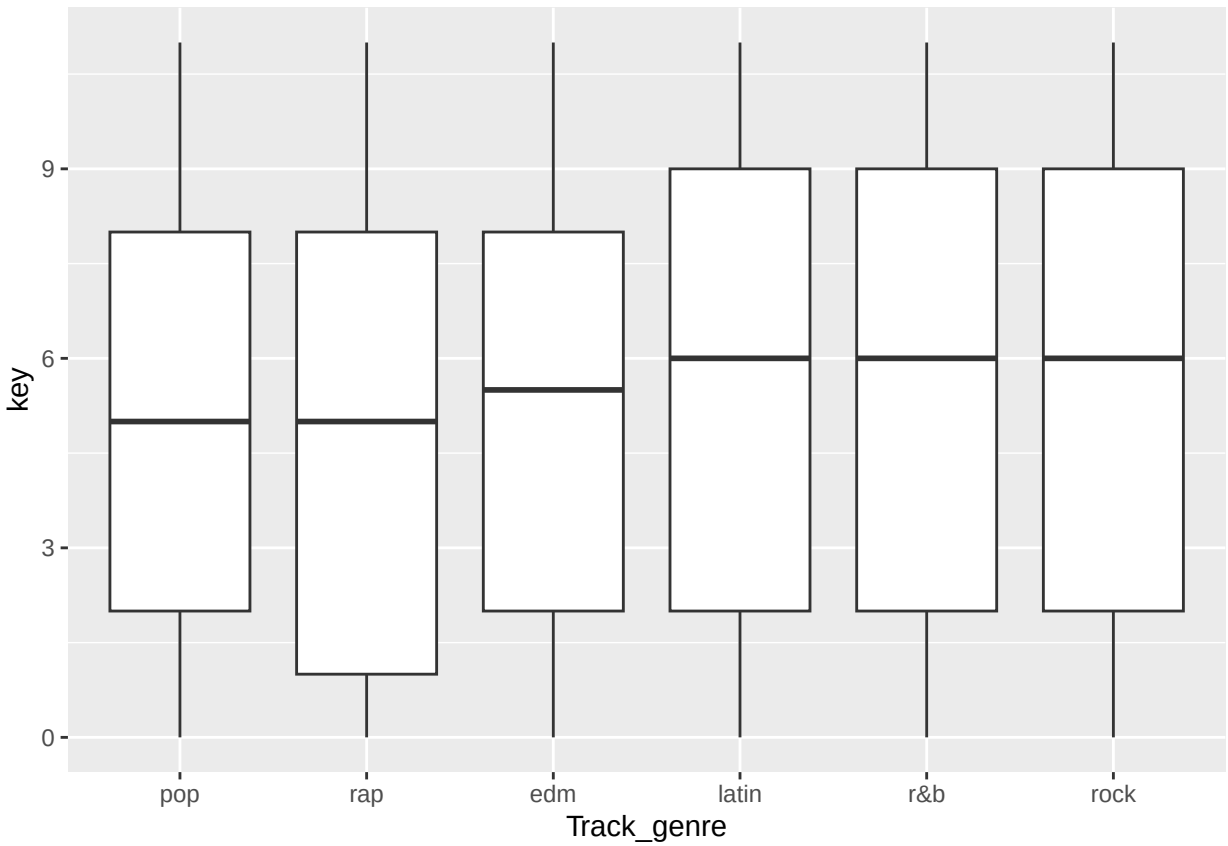


Figure 17: Boxplot between key and track_genre

```
spotify_1000 %>%
  group_by(track_genre) %>% summarise(mean_v = median(key)) %>% arrange(mean_v)
```

```
## # A tibble: 6 x 2
```

```
##   track_genre mean_v
##   <fct>        <dbl>
## 1 pop            5
## 2 rap            5
## 3 edm            5.5
## 4 latin          6
## 5 r&b            6
## 6 rock           6
```

```
## Relationship between loudness and track's genres
spotify_1000 %>% ggplot(aes(x = fct_reorder(track_genre, loudness), y = loudness)) +
  geom_boxplot() + xlab("Track_genre")
```

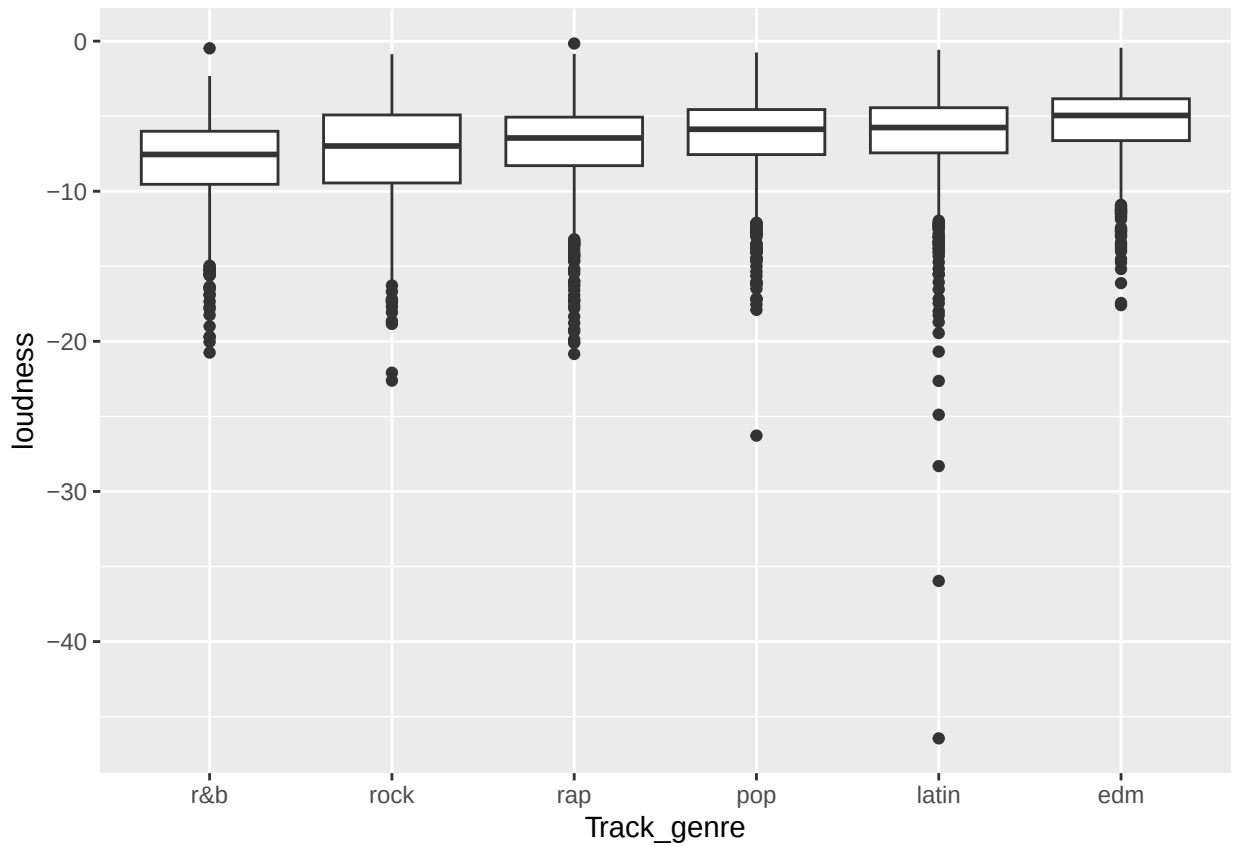


Figure 18: Boxplot between loudness and track_genre

```
spotify_1000 %>%
  group_by(track_genre) %>% summarise(mean_v = median(loudness)) %>% arrange(mean_v)
```

```
## # A tibble: 6 x 2
##   track_genre mean_v
##   <fct>        <dbl>
## 1 r&b          -7.55
## 2 rock         -6.99
## 3 rap          -6.45
```

```
## 4 pop          -5.87
## 5 latin        -5.76
## 6 edm          -4.95
```

```
## Relationship between mode and track's genres
spotify_1000 %>% ggplot(aes(x = track_genre, fill = factor(mode))) +
  geom_bar() + xlab("Track_genre") + ylab("Frequency")
```

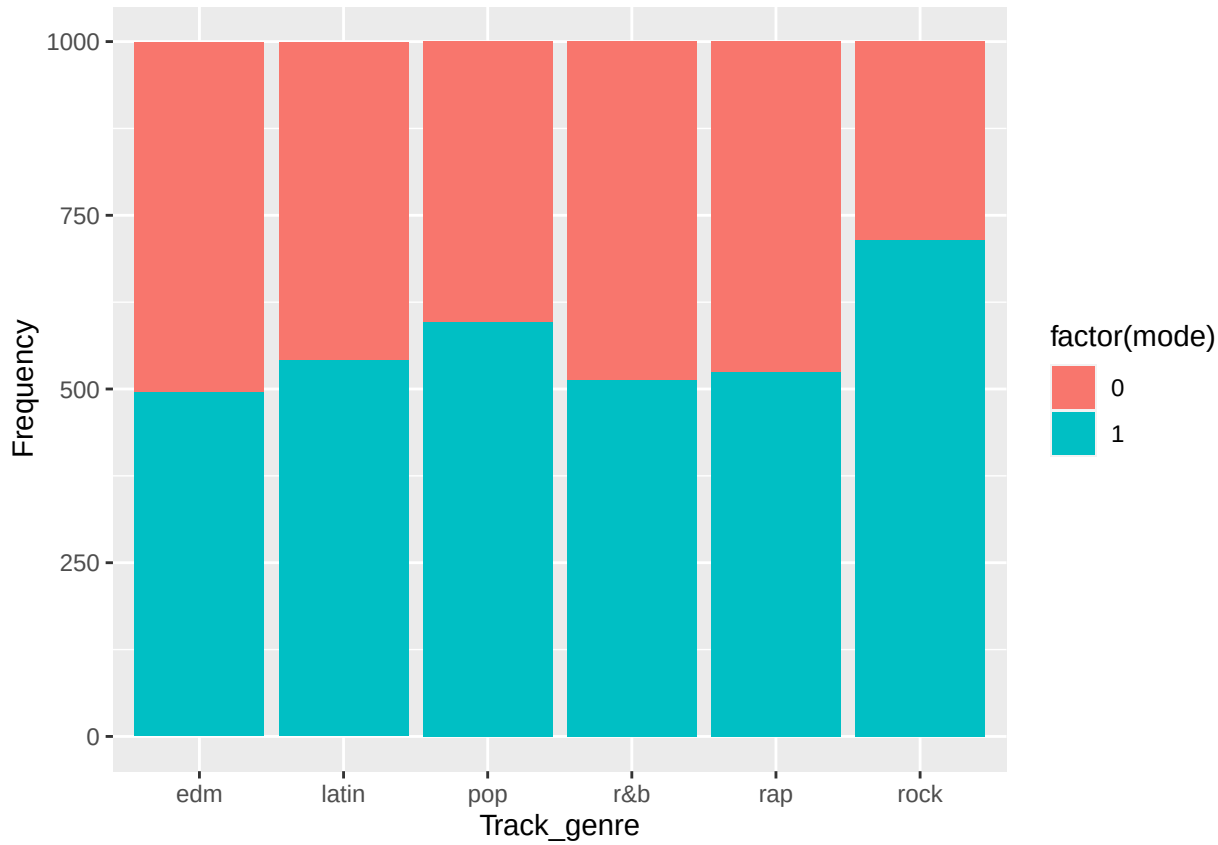


Figure 19: Bar chart of track_genre, colored by mode

```
## Relationship between speechiness and track's genres
spotify_1000 %>%
  ggplot(aes(x = fct_reorder(track_genre, speechiness), y = speechiness)) +
  geom_boxplot() + xlab("Track_genre")
```

```
spotify_1000 %>%
  group_by(track_genre) %>% summarise(mean_v = median(speechiness)) %>% arrange(mean_v)
```

```
## # A tibble: 6 x 2
##   track_genre mean_v
##   <fct>        <dbl>
## 1 rock        0.0415
## 2 pop         0.0476
```

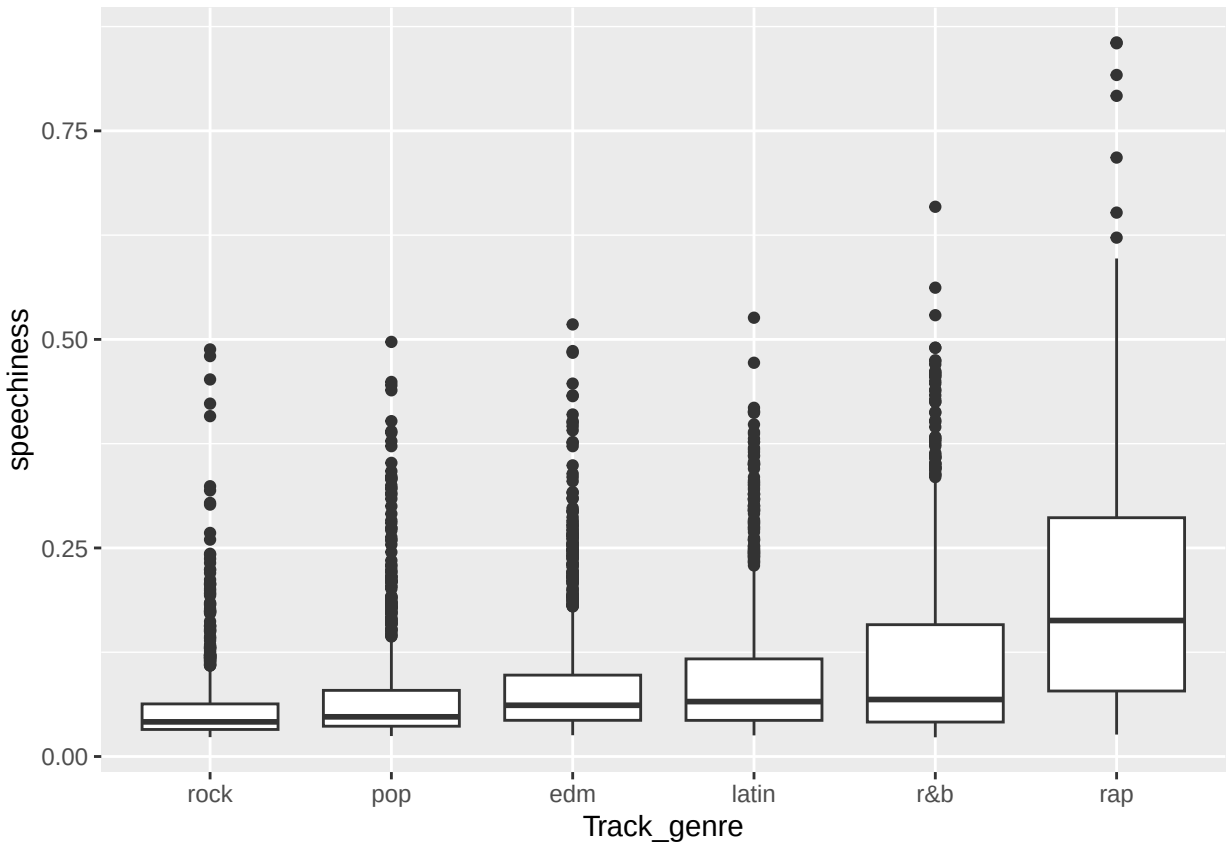


Figure 20: Boxplot between speechiness and track_genre

```
## 3 edm          0.0614
## 4 latin        0.0658
## 5 r&b          0.0683
## 6 rap          0.163
```

```
## Relationship between acousticness and track's genres
spotify_1000 %>%
  ggplot(aes(x = fct_reorder(track_genre, acousticness), y = acousticness)) +
  geom_boxplot() + xlab("Track_genre")
```

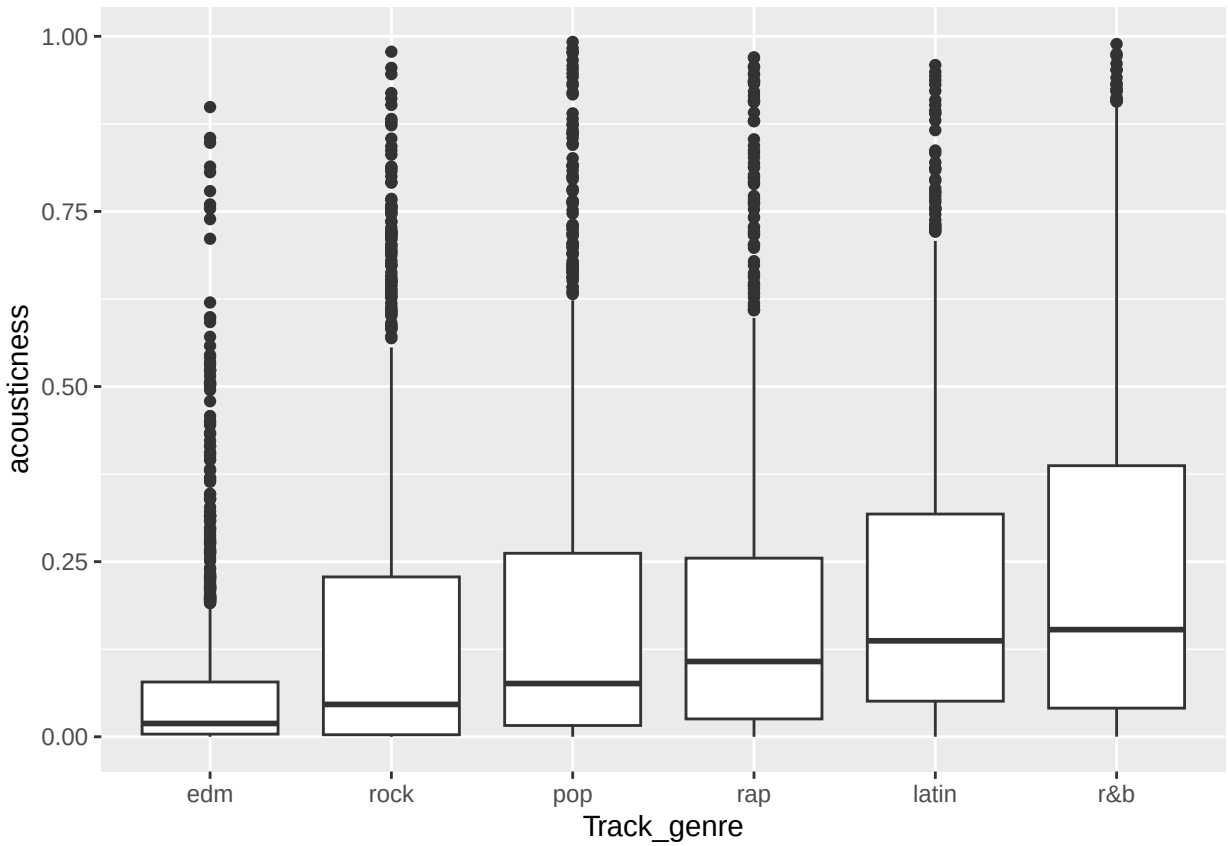


Figure 21: Boxplot between acousticness and track_genre

```
spotify_1000 %>%
  group_by(track_genre) %>%
  summarise(mean_v = median(acousticness)) %>% arrange(mean_v)
```

```
## # A tibble: 6 x 2
##   track_genre mean_v
##   <fct>        <dbl>
## 1 edm          0.019
## 2 rock         0.0462
## 3 pop          0.0761
## 4 rap          0.108
## 5 latin        0.137
## 6 r&b          0.153
```

```
## Relationship between instrumentalness and track's genres
spotify_1000 %>% ggplot(aes(x = fct_reorder(track_genre, instrumentalness),
                            y = instrumentalness)) +
  geom_boxplot() + xlab("Track_genre")
```

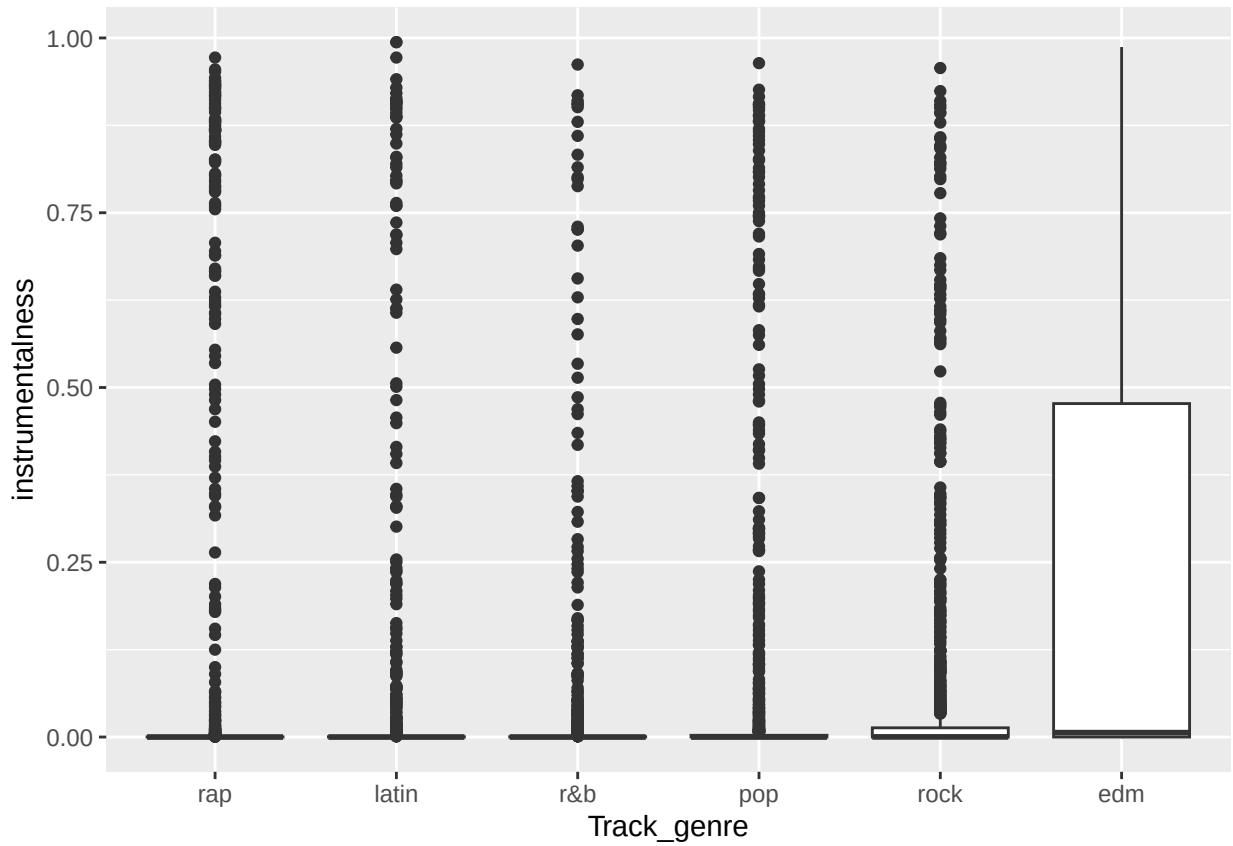


Figure 22: Boxplot between instrumentalness and track_genre

```
spotify_1000 %>%
  group_by(track_genre) %>%
  summarise(mean_v = median(instrumentalness)) %>% arrange(mean_v)
```

```
## # A tibble: 6 x 2
##   track_genre     mean_v
##   <fct>            <dbl>
## 1 rap         0
## 2 latin       0.00000261
## 3 r&b         0.00000449
## 4 pop         0.0000116
## 5 rock        0.000204
## 6 edm         0.00627
```

```
## Relationship between liveness and track's genres
spotify_1000 %>% ggplot(aes(x = fct_reorder(track_genre, liveness), y = liveness)) +
  geom_boxplot() + xlab("Track_genre")
```

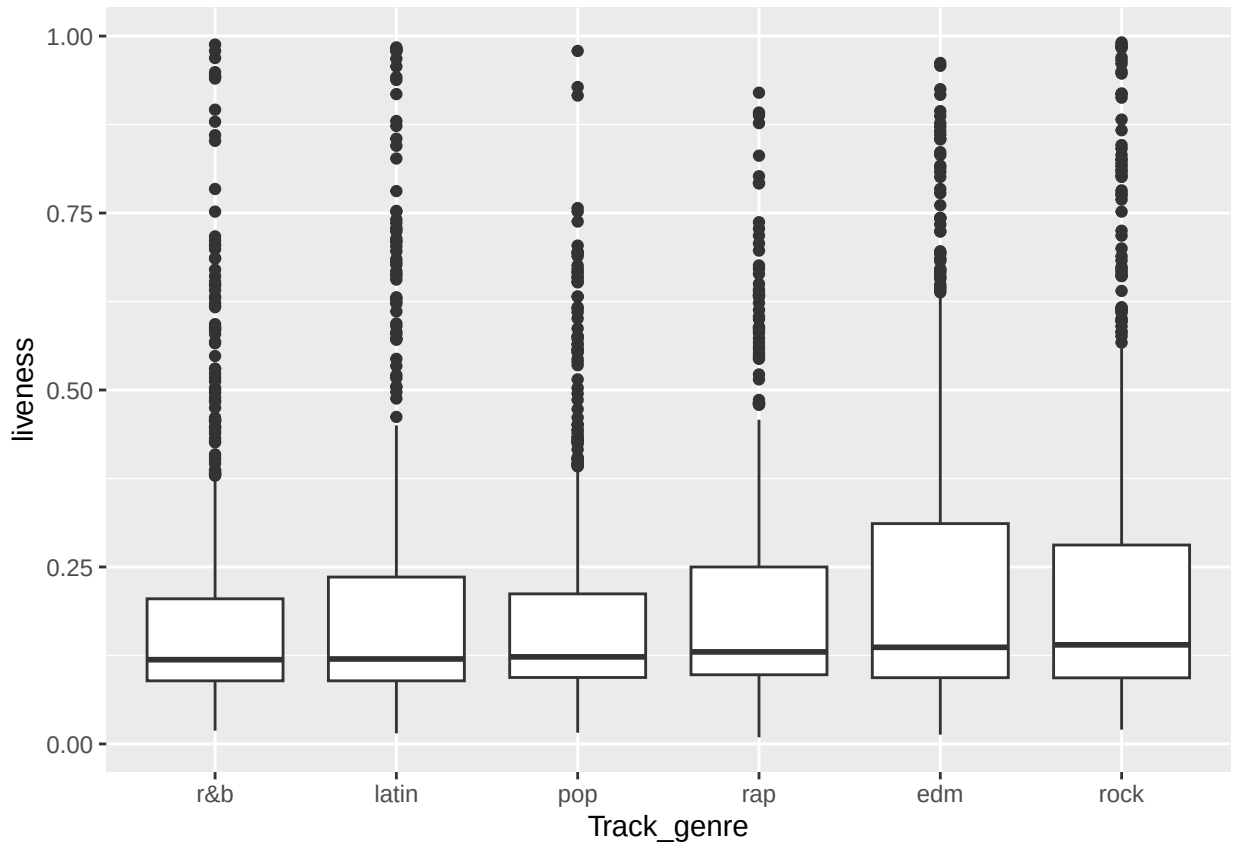


Figure 23: Boxplot between liveness and track_genre

```
spotify_1000 %>%
  group_by(track_genre) %>% summarise(mean_v = median(liveness)) %>% arrange(mean_v)
```

```
## # A tibble: 6 x 2
##   track_genre mean_v
##   <fct>        <dbl>
## 1 r&b          0.119
## 2 latin        0.12
## 3 pop          0.123
## 4 rap          0.13
## 5 edm          0.136
## 6 rock         0.14
```

```
## Relationship between valence and track's genres
spotify_1000 %>% ggplot(aes(x = fct_reorder(track_genre, valence), y = valence)) +
  geom_boxplot() + xlab("Track_genre")
```

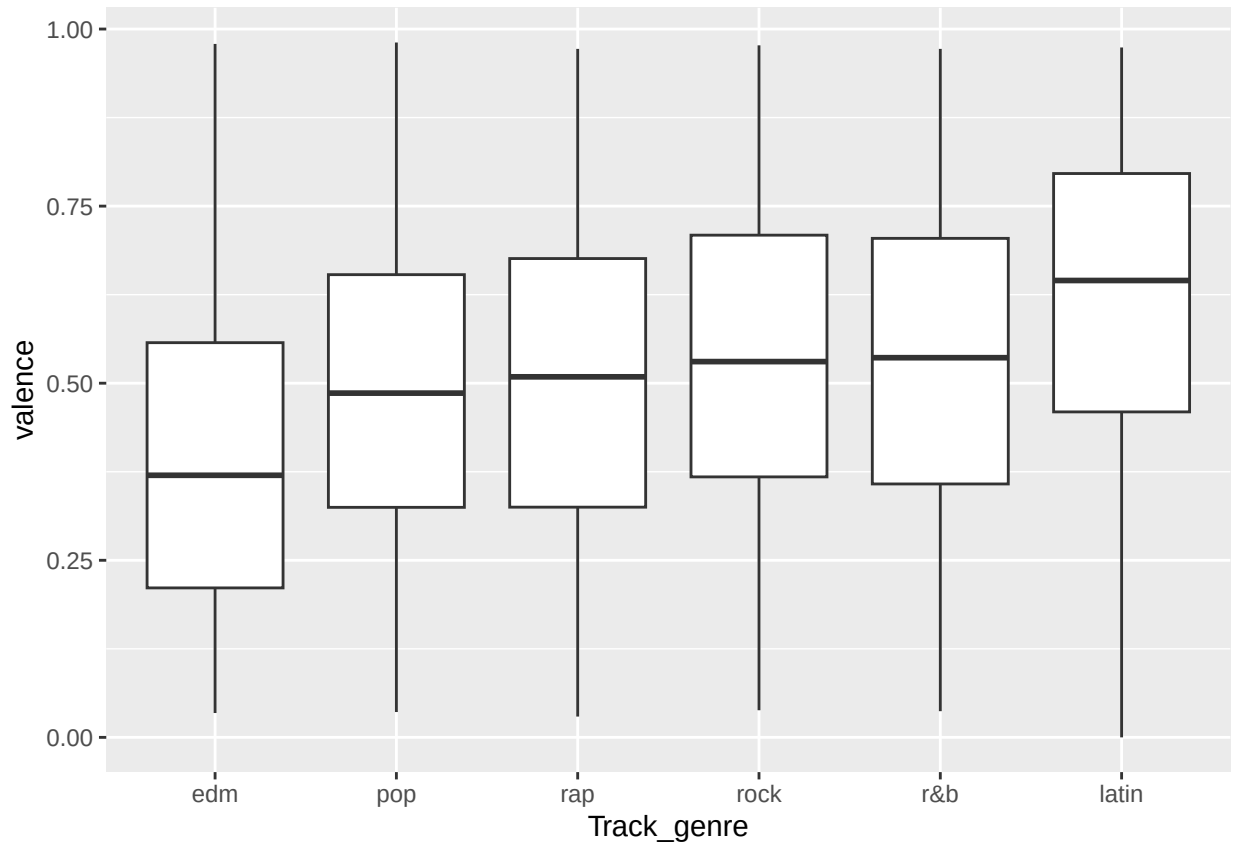


Figure 24: Boxplot between valence and track_genre

```
spotify_1000 %>%
  group_by(track_genre) %>% summarise(mean_v = median(valence)) %>% arrange(mean_v)
```

```
## # A tibble: 6 x 2
```

```
##   track_genre mean_v
##   <fct>        <dbl>
## 1 edm          0.37
## 2 pop          0.486
## 3 rap          0.509
## 4 rock         0.530
## 5 r&b          0.536
## 6 latin        0.645
```

```
## Relationship between tempo and track's genres
spotify_1000 %>% ggplot(aes(x = fct_reorder(track_genre, tempo), y = tempo)) +
  geom_boxplot() + xlab("Track_genre")
```

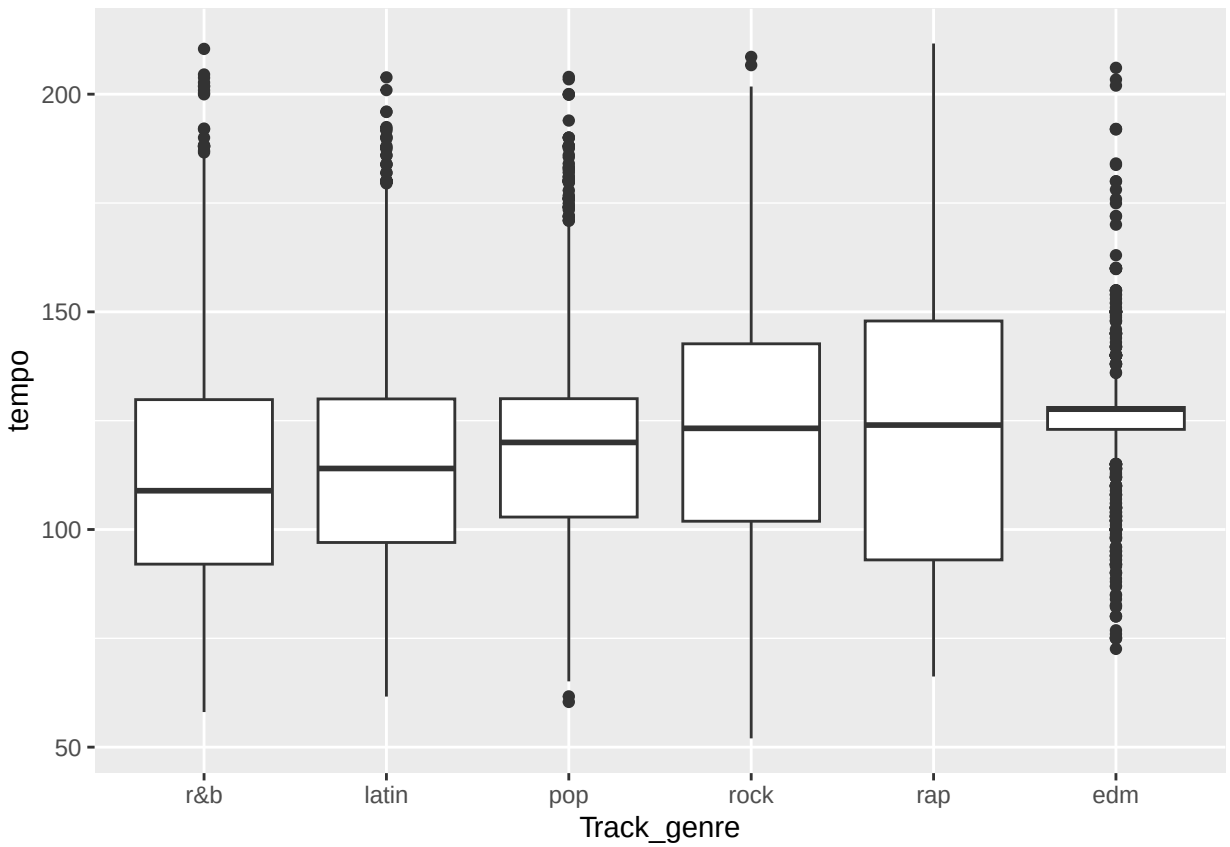


Figure 25: Boxplot between tempo and track_genre

```
spotify_1000 %>%
  group_by(track_genre) %>% summarise(mean_v = median(tempo)) %>% arrange(mean_v)
```

```
## # A tibble: 6 x 2
##   track_genre mean_v
##   <fct>        <dbl>
## 1 r&b           109.
## 2 latin         114.
## 3 pop           120.
```

```
## 4 rock          123.
## 5 rap           124.
## 6 edm           128.
```

```
## Relationship between duration_ms and track's genres
spotify_1000 %>%
  ggplot(aes(x = fct_reorder(track_genre, duration_ms), y = duration_ms)) +
  geom_boxplot() + xlab("Track_genre")
```

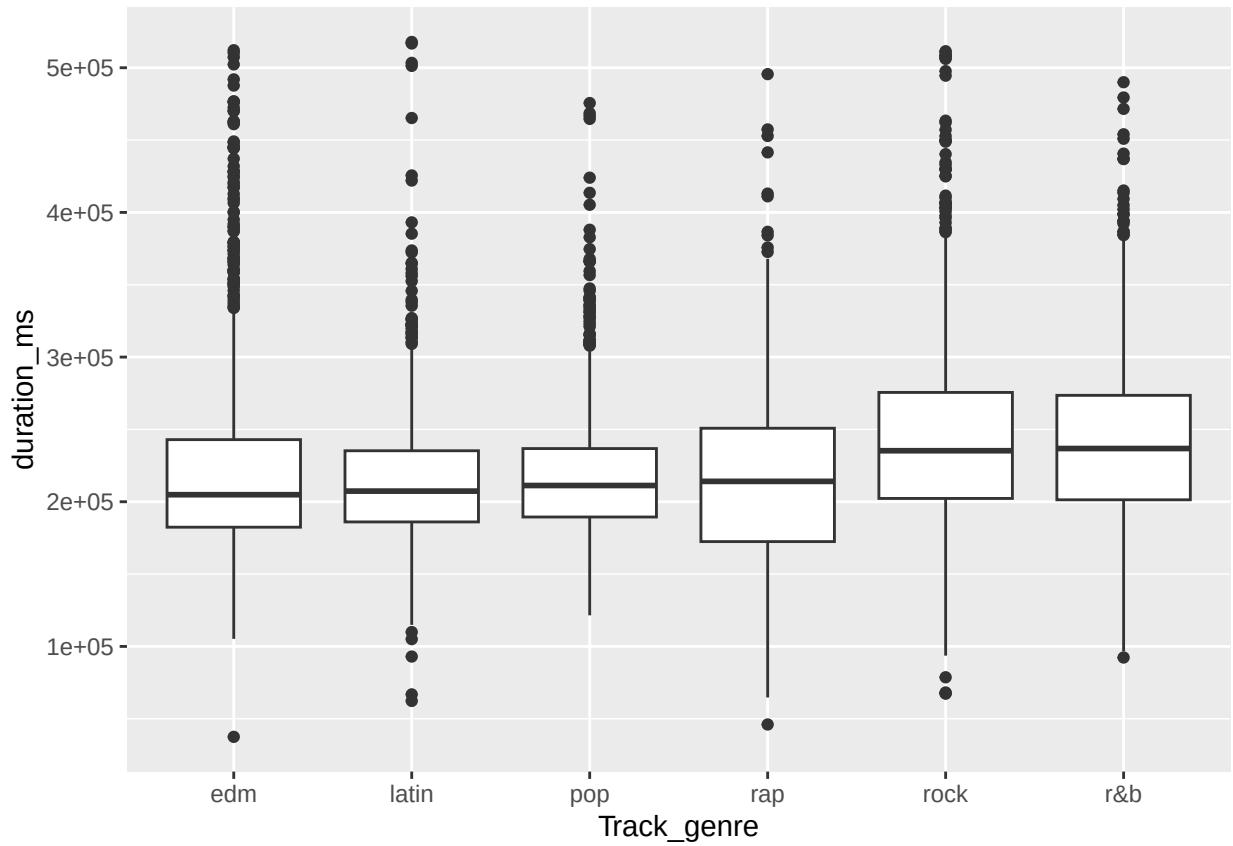


Figure 26: Boxplot between duration_ms and track_genre

```
spotify_1000 %>%
  group_by(track_genre) %>%
  summarise(mean_v = median(duration_ms)) %>% arrange(mean_v)
```

```
## # A tibble: 6 x 2
##   track_genre  mean_v
##   <fct>         <dbl>
## 1 edm         204862.
## 2 latin       207354
## 3 pop         211225
## 4 rap         214076.
## 5 rock        235266.
## 6 r&b         236796.
```

```
## Relationship between year_released and track's genres
spotify_1000 %>% ggplot(aes(x = factor(track_genre), fill = factor(year_released))) +
  geom_bar(position = "fill", col = "black") + xlab("Track_genre") +
  ylab("Frequency") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```

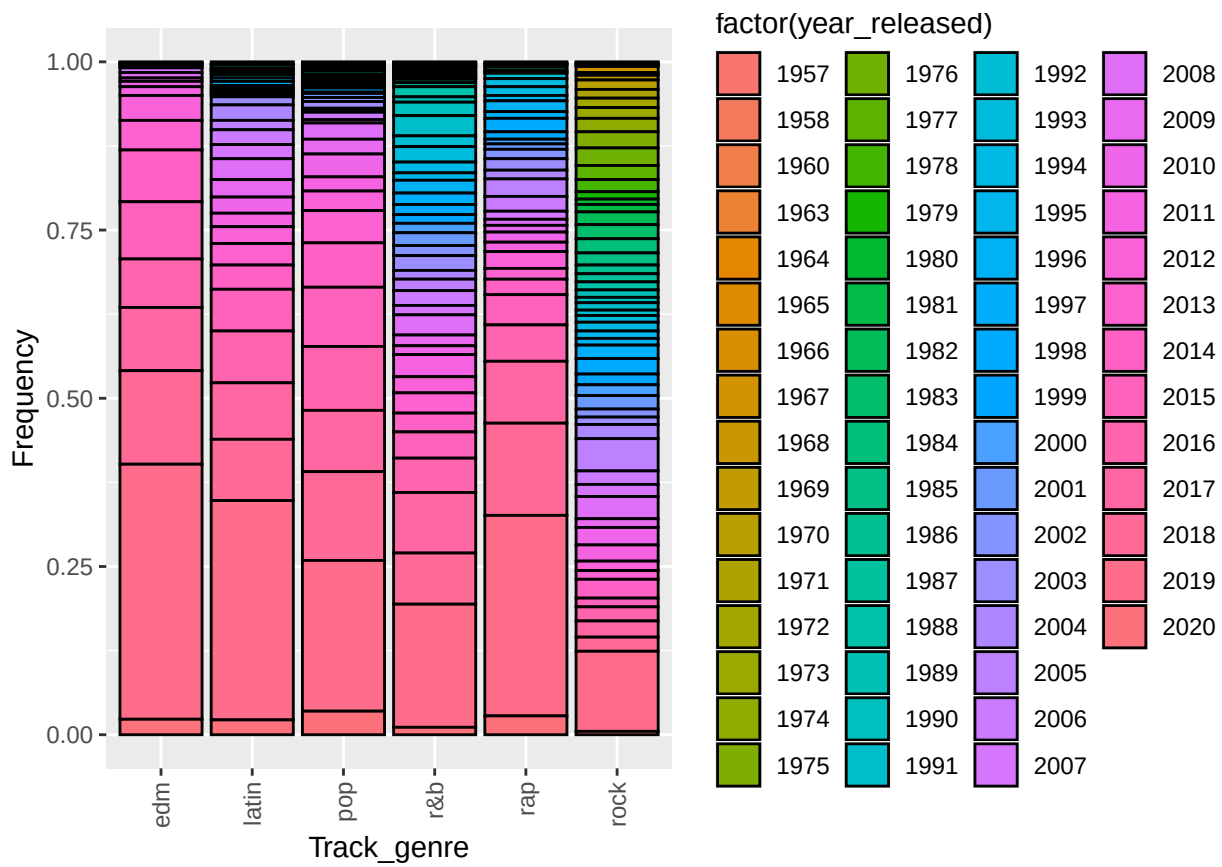


Figure 27: Bar chart of track_genre, colored by year_released

```
## Check proportion of track's genres from 1957 to 1990
spotify_1000 %>% count(track_genre, year_released) %>%
  filter(between(year_released,1957, 1990)) %>%
  group_by(track_genre) %>% mutate(total = sum(n)) %>% ungroup() %>%
  distinct(track_genre, total) %>% mutate(prop = round(total*100/sum(total),3))
```

```
## # A tibble: 6 x 3
##   track_genre total   prop
##   <fct>       <int>  <dbl>
## 1 edm             1  0.205
## 2 latin           9  1.85
## 3 pop            24  4.93
## 4 r&b            80 16.4
## 5 rap            15  3.08
## 6 rock          358 73.5
```

```
## Check proportion of track's genres from 1991 to 2000
spotify_1000 %>% count(track_genre, year_released) %>%
  filter(between(year_released,1991, 2000)) %>%
  group_by(track_genre) %>% mutate(total = sum(n)) %>% ungroup() %>%
  distinct(track_genre, total) %>% mutate(prop = round(total*100/sum(total),3))
```

```
## # A tibble: 6 x 3
##   track_genre total   prop
##   <fct>       <int>  <dbl>
## 1 edm             2  0.416
## 2 latin          37  7.69
## 3 pop            23  4.78
## 4 r&b           174 36.2
## 5 rap           107 22.2
## 6 rock          138 28.7
```

```
## Check proportion of track's genres from 2001 to 2020
spotify_1000 %>% count(track_genre, year_released) %>%
  filter(between(year_released,2001, 2020)) %>%
  group_by(track_genre) %>% mutate(total = sum(n)) %>% ungroup() %>%
  distinct(track_genre, total) %>% mutate(prop = round(total*100/sum(total),3))
```

```
## # A tibble: 6 x 3
##   track_genre total  prop
##   <fct>       <int> <dbl>
## 1 edm           997  19.8
## 2 latin         954  19.0
## 3 pop           953  18.9
## 4 r&b           746  14.8
## 5 rap           878  17.4
## 6 rock          504  10.0
```

```
## Check proportion of released year in edm tracks
print(spotify_final %>% filter(track_genre == "edm") %>%
        count(year_released) %>% arrange(desc(year_released)) %>%
        mutate(prop = n/sum(n), cum_sum = cumsum(prop)), n = 40)
```

```
## # A tibble: 36 x 4
##    year_released     n     prop cum_sum
##            <dbl> <int>    <dbl>   <dbl>
##  1          2020   135 0.0253    0.0253
##  2          2019  1979 0.371     0.397
##  3          2018   712 0.134     0.530
##  4          2017   453 0.0850    0.615
##  5          2016   450 0.0845    0.700
##  6          2015   431 0.0809    0.781
##  7          2014   462 0.0867    0.867
##  8          2013   276 0.0518    0.919
##  9          2012   145 0.0272    0.947
## 10          2011    69 0.0130    0.959
## 11          2010    37 0.00694   0.966
## 12          2009    45 0.00845   0.975
```

```
## 13          2008    43 0.00807   0.983
## 14          2007    29 0.00544   0.988
## 15          2006    16 0.00300   0.991
## 16          2005     8 0.00150   0.993
## 17          2004     3 0.000563  0.993
## 18          2003     5 0.000938  0.994
## 19          2002     4 0.000751  0.995
## 20          2001     3 0.000563  0.996
## 21          2000     4 0.000751  0.996
## 22          1999     3 0.000563  0.997
## 23          1998     1 0.000188  0.997
## 24          1996     1 0.000188  0.997
## 25          1995     1 0.000188  0.998
## 26          1994     1 0.000188  0.998
## 27          1993     1 0.000188  0.998
## 28          1991     1 0.000188  0.998
## 29          1990     1 0.000188  0.998
## 30          1983     1 0.000188  0.998
## 31          1982     2 0.000375  0.999
## 32          1981     1 0.000188  0.999
## 33          1980     1 0.000188  0.999
## 34          1979     2 0.000375  1.00
## 35          1978     1 0.000188  1.00
## 36          1977     1 0.000188  1
```

```
## How does track popularity change over time?
spotify_1000 %>% ggplot(aes(x = factor(year_released), y = track_popularity)) +
  geom_boxplot() + xlab("Year_released") +
  theme(axis.text.x = element_text(angle =90, hjust = 1, vjust = 0.5))
```

```
# Check top 10 artists having the highest number of tracks
## Get list of top track artists
spotify_1000_artist_origin %>% count(track_artist) %>% arrange(desc(n)) %>% top_n(10)
```

```
## # A tibble: 12 x 2
##    track_artist             n
##    <chr>                <int>
##  1 Queen                   34
##  2 Ballin Entertainment    17
##  3 Don Omar                17
##  4 2Pac                    16
##  5 Martin Garrix           16
##  6 Rihanna                 16
##  7 David Guetta            14
##  8 Drake                   13
##  9 Hardwell                13
## 10 Logic                   13
## 11 R3HAB                   13
## 12 The Chainsmokers        13
```

```
## Check proportion of track's genres of Queen
spotify_1000_artist %>% filter(queen==1) %>% count(track_genre) %>%
  mutate(total = sum(n),prop = round(n*100/total,3)) %>% arrange(desc(n))
```

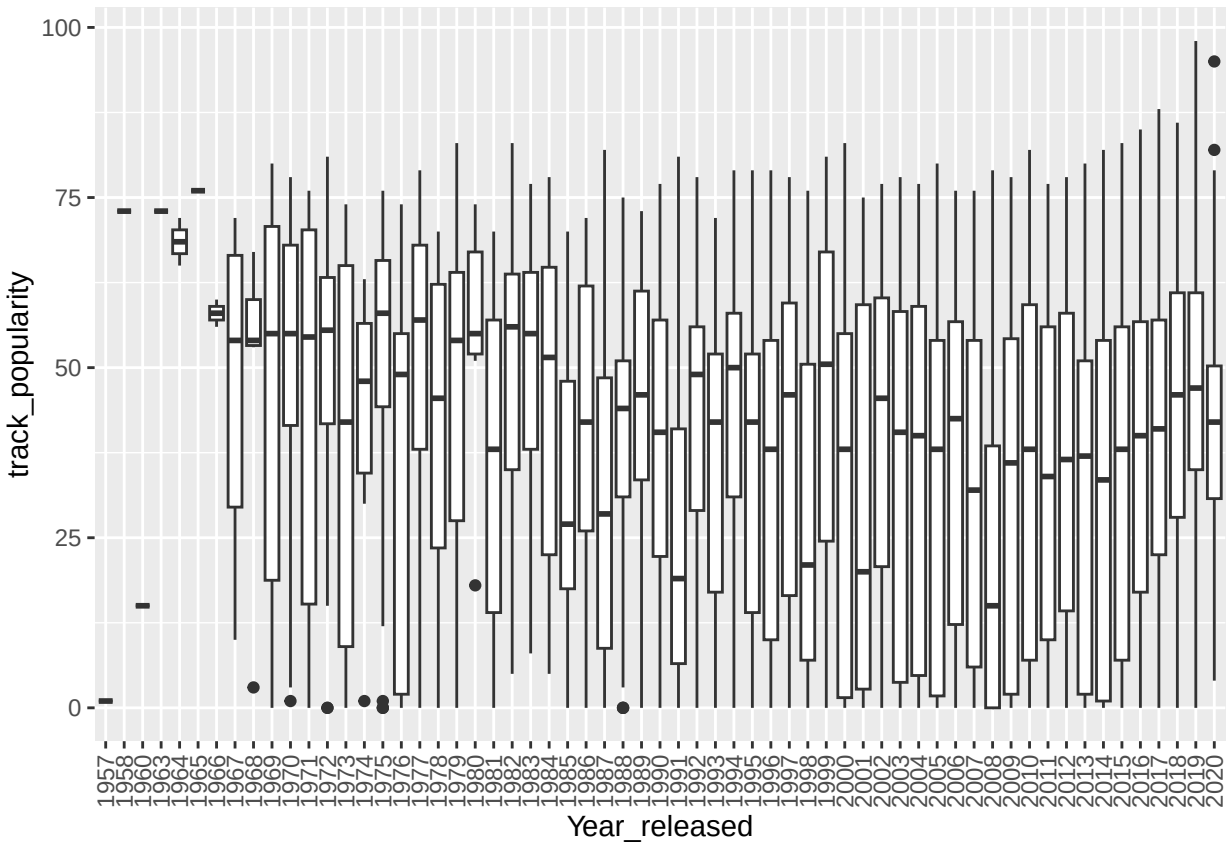


Figure 28: Boxplot between track_popularity and year_released

```
## # A tibble: 1 x 4
##   track_genre     n total  prop
##   <fct>       <int> <int> <dbl>
## 1 rock           34    34   100
```

```
## Check proportion of track's genres of Ballin Entertainment
spotify_1000_artist %>% filter(ballin_entertainment==1) %>% count(track_genre) %>%
  mutate(total = sum(n),prop = round(n*100/total,3)) %>% arrange(desc(n))
```

```
## # A tibble: 1 x 4
##   track_genre     n total  prop
##   <fct>       <int> <int> <dbl>
## 1 latin          17    17   100
```

```
## Check proportion of track's genres of Don Omar
spotify_1000_artist %>% filter(don_omar==1) %>% count(track_genre) %>%
  mutate(total = sum(n),prop = round(n*100/total,3)) %>% arrange(desc(n))
```

```
## # A tibble: 1 x 4
##   track_genre     n total  prop
##   <fct>       <int> <int> <dbl>
## 1 latin          17    17   100
```

```
## Check proportion of track's genres of 2Pac
spotify_1000_artist %>% filter(x2pac==1) %>% count(track_genre) %>%
  mutate(total = sum(n),prop = round(n*100/total,3)) %>% arrange(desc(n))
```

```
## # A tibble: 2 x 4
##   track_genre     n total  prop
##   <fct>       <int> <int> <dbl>
## 1 rap            15    16 93.8
## 2 r&b             1    16  6.25
```

```
## Check proportion of track's genres of Martin Garrix
spotify_1000_artist %>% filter(martin_garrix==1) %>% count(track_genre) %>%
  mutate(total = sum(n),prop = round(n*100/total,3)) %>% arrange(desc(n))
```

```
## # A tibble: 2 x 4
##   track_genre     n total  prop
##   <fct>       <int> <int> <dbl>
## 1 edm            14    16  87.5
## 2 pop             2    16  12.5
```

```
## Check proportion of track's genres of Rihanna
spotify_1000_artist %>% filter(rihanna==1) %>% count(track_genre) %>%
  mutate(total = sum(n),prop = round(n*100/total,3)) %>% arrange(desc(n))
```

```
## # A tibble: 5 x 4
##   track_genre     n total  prop
##   <fct>       <int> <int> <dbl>
```

```
## 1 edm             7    16 43.8
## 2 pop             4    16 25
## 3 latin           2    16 12.5
## 4 r&b             2    16 12.5
## 5 rap             1    16  6.25
```

```
## Check proportion of track's genres of David Guetta
spotify_1000_artist %>% filter(david_guetta==1) %>% count(track_genre) %>%
  mutate(total = sum(n),prop = round(n*100/total,3)) %>% arrange(desc(n))
```

```
## # A tibble: 4 x 4
##   track_genre     n total  prop
##   <fct>       <int> <int> <dbl>
## 1 edm             7    14 50
## 2 pop             5    14 35.7
## 3 latin           1    14  7.14
## 4 rap             1    14  7.14
```

```
## Check proportion of track's genres of Drake
spotify_1000_artist %>% filter(drake==1) %>% count(track_genre) %>%
  mutate(total = sum(n),prop = round(n*100/total,3)) %>% arrange(desc(n))
```

```
## # A tibble: 4 x 4
##   track_genre     n total  prop
##   <fct>       <int> <int> <dbl>
## 1 r&b             7    13  53.8
## 2 latin           2    13  15.4
## 3 pop             2    13  15.4
## 4 rap             2    13  15.4
```

```
## Check proportion of track's genres of Hardwell
spotify_1000_artist %>% filter(hardwell==1) %>% count(track_genre) %>%
  mutate(total = sum(n),prop = round(n*100/total,3)) %>% arrange(desc(n))
```

```
## # A tibble: 1 x 4
##   track_genre     n total  prop
##   <fct>       <int> <int> <dbl>
## 1 edm            13    13   100
```

```
## Check proportion of track's genres of Logic
spotify_1000_artist %>% filter(logic==1) %>% count(track_genre) %>%
  mutate(total = sum(n),prop = round(n*100/total,3)) %>% arrange(desc(n))
```

```
## # A tibble: 2 x 4
##   track_genre     n total  prop
##   <fct>       <int> <int> <dbl>
## 1 rap            12    13 92.3
## 2 pop             1    13  7.69
```

```r
## Check proportion of track's genres of R3HAb
spotify_1000_artist %>% filter(r3hab==1) %>% count(track_genre) %>%
  mutate(total = sum(n),prop = round(n*100/total,3)) %>% arrange(desc(n))
```

```
## # A tibble: 3 x 4
##   track_genre     n total  prop
##   <fct>       <int> <int> <dbl>
## 1 edm             8    13 61.5
## 2 pop             4    13 30.8
## 3 latin           1    13  7.69
```

```r
## Check proportion of track's genres of The Chainsmokers
spotify_1000_artist %>% filter(the_chainsmokers==1) %>% count(track_genre) %>%
  mutate(total = sum(n),prop = round(n*100/total,3)) %>% arrange(desc(n))
```

```
## # A tibble: 3 x 4
##   track_genre     n total  prop
##   <fct>       <int> <int> <dbl>
## 1 pop             8    13 61.5
## 2 edm             4    13 30.8
## 3 rap             1    13  7.69
```

```r
# BUILDING MODELS
# Split the data set into a training set and a testing set
spotify_split <- initial_split(spotify_1000)
spotify_train <- training(spotify_split)
spotify_test <- testing(spotify_split)
```

```r
# Create recipe to process, including PCA
recipe_spotify_pca <- recipe(track_genre ~ ., data = spotify_train) %>%
  step_downsample(track_genre) %>%
  step_zv(all_predictors()) %>%
  step_normalize(all_predictors()) %>%
  step_corr(all_predictors()) %>%
  step_pca(all_predictors()) %>%
  prep()

recipe_spotify_pca
```

```r
# Take variance explained by principle components
sdev_value <- recipe_spotify_pca$steps[[5]]$res$sdev
ve <- sdev_value^2 / sum(sdev_value^2)

# Find necessary number of principle components to get 90% variance explained
pc_sdev <- tibble(pc = fct_inorder(str_c("PC",1:43)),
                  pve = cumsum(ve))
pc_sdev %>% filter(pve >= 0.9)
```

```
## # A tibble: 8 x 2
##   pc      pve
##   <fct> <dbl>
```

```
## 1 PC36  0.907
## 2 PC37  0.925
## 3 PC38  0.943
## 4 PC39  0.959
## 5 PC40  0.973
## 6 PC41  0.987
## 7 PC42  0.995
## 8 PC43  1
```

```
# Because PCA does not reduce many predictors, therefore, I decided not to use PCA
recipe_spotify <- recipe(track_genre ~ ., data = spotify_train) %>%
  step_downsample(track_genre) %>%
  step_zv(all_predictors()) %>%
  step_normalize(all_predictors()) %>%
  step_corr(all_predictors()) %>%
  prep()

recipe_spotify
```

```
# Apply recipe on the training set to process training set
pre_spotify_train <- juice(recipe_spotify)
skim(pre_spotify_train) %>% knit_print()
```

Table 1: Data summary

Name	pre_spotify_train
Number of rows	4464
Number of columns	44
Column type frequency:	
factor	1
numeric	43
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
track_genre	0	1	FALSE	6	edm: 744, lat: 744, pop: 744, r&b: 744

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
track_popularity	0	1	0	1	-1.66	-0.77	0.11	0.78	2.42	
danceability	0	1	0	1	-3.89	-0.62	0.13	0.72	2.19	
energy	0	1	0	1	-3.81	-0.66	0.13	0.78	1.64	
key	0	1	0	1	-1.51	-0.95	0.17	0.73	1.57	
loudness	0	1	0	1	-12.66	-0.47	0.19	0.69	2.13	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
mode	0	1	0	1	-1.13	-1.13	0.88	0.88	0.88	
speechiness	0	1	0	1	-0.81	-0.64	-0.43	0.22	7.39	
acousticness	0	1	0	1	-0.80	-0.73	-0.43	0.36	3.64	
instrumentalness	0	1	0	1	-0.38	-0.38	-0.38	-0.36	4.03	
liveness	0	1	0	1	-1.15	-0.62	-0.41	0.36	5.02	
valence	0	1	0	1	-2.16	-0.78	0.01	0.79	2.00	
tempo	0	1	0	1	-2.33	-0.78	0.04	0.52	3.36	
duration_ms	0	1	0	1	-3.14	-0.63	-0.16	0.47	4.84	
year_released	0	1	0	1	-4.51	-0.23	0.46	0.72	0.80	
x2pac	0	1	0	1	-0.05	-0.05	-0.05	-0.05	20.12	
x50_cent	0	1	0	1	-0.04	-0.04	-0.04	-0.04	23.60	
ariana_grande	0	1	0	1	-0.04	-0.04	-0.04	-0.04	22.25	
ballin_entertainment	0	1	0	1	-0.06	-0.06	-0.06	-0.06	17.22	
calvin_harris	0	1	0	1	-0.04	-0.04	-0.04	-0.04	22.25	
coldplay	0	1	0	1	-0.04	-0.04	-0.04	-0.04	22.25	
creedence_clearwater_revival	0	1	0	1	-0.04	-0.04	-0.04	-0.04	25.23	
david_guetta	0	1	0	1	-0.05	-0.05	-0.05	-0.05	18.50	
don_omar	0	1	0	1	-0.06	-0.06	-0.06	-0.06	17.83	
drake	0	1	0	1	-0.05	-0.05	-0.05	-0.05	20.12	
eminem	0	1	0	1	-0.03	-0.03	-0.03	-0.03	29.86	
gloria_estefan	0	1	0	1	-0.05	-0.05	-0.05	-0.05	21.10	
guns_n_roses	0	1	0	1	-0.04	-0.04	-0.04	-0.04	23.60	
hardwell	0	1	0	1	-0.04	-0.04	-0.04	-0.04	23.60	
j_balvin	0	1	0	1	-0.05	-0.05	-0.05	-0.05	21.10	
janelle_monae	0	1	0	1	-0.04	-0.04	-0.04	-0.04	22.25	
kiss	0	1	0	1	-0.05	-0.05	-0.05	-0.05	21.10	
logic	0	1	0	1	-0.04	-0.04	-0.04	-0.04	27.25	
martin_garrix	0	1	0	1	-0.05	-0.05	-0.05	-0.05	20.12	
ozuna	0	1	0	1	-0.04	-0.04	-0.04	-0.04	22.25	
queen	0	1	0	1	-0.08	-0.08	-0.08	-0.08	12.59	
r3hab	0	1	0	1	-0.04	-0.04	-0.04	-0.04	22.25	
rick_ross	0	1	0	1	-0.04	-0.04	-0.04	-0.04	23.60	
rihanna	0	1	0	1	-0.05	-0.05	-0.05	-0.05	20.12	
scorpions	0	1	0	1	-0.04	-0.04	-0.04	-0.04	23.60	
soda_stereo	0	1	0	1	-0.04	-0.04	-0.04	-0.04	27.25	
the_chainsmokers	0	1	0	1	-0.04	-0.04	-0.04	-0.04	22.25	
the_cranberries	0	1	0	1	-0.04	-0.04	-0.04	-0.04	27.25	
tiesto	0	1	0	1	-0.04	-0.04	-0.04	-0.04	23.60	

```
# Apply recipe on the testing set to process testing set
pre_spotify_test <- bake(recipe_spotify, spotify_test)
```

```
# Make cross validation resamples from preprocessed training set
spotify_cv_10 <- vfold_cv(pre_spotify_train, v = 10, strata = track_genre)
spotify_cv_10
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 2
##    splits             id
##    <list>             <chr>
##  1 <split [4014/450]> Fold01
```

```
##  2 <split [4014/450]> Fold02
##  3 <split [4014/450]> Fold03
##  4 <split [4014/450]> Fold04
##  5 <split [4020/444]> Fold05
##  6 <split [4020/444]> Fold06
##  7 <split [4020/444]> Fold07
##  8 <split [4020/444]> Fold08
##  9 <split [4020/444]> Fold09
## 10 <split [4020/444]> Fold10
```

```
# Build a linear discriminant analysis
## Set the model specification
lda_spotify <- discrim_linear(mode = "classification") %>%
  set_engine("MASS")

## Fit the model to cross validation resamples
lda_fit <- fit_resamples(lda_spotify,
                         preprocessor = recipe(track_genre ~., data = pre_spotify_train),
                         resamples = spotify_cv_10)

## Get average accuracy and average AUC of the model
lda_fit %>% collect_metrics()
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy multiclass 0.505    10 0.00447 Preprocessor1_Model1
## 2 roc_auc  hand_till  0.817    10 0.00311 Preprocessor1_Model1
```

```
# A K-nearest neighbours model
doParallel::registerDoParallel()
## Set model specification
knearest_spotify <- nearest_neighbor(mode = "classification", neighbors = tune()) %>%
  set_engine("kknn")

## Create a grid containing possible values of neighbors
k_grid <- grid_regular(neighbors(range = c(1, 100)), levels = 20)
k_grid
```

```
## # A tibble: 20 x 1
##    neighbors
##        <int>
##  1         1
##  2         6
##  3        11
##  4        16
##  5        21
##  6        27
##  7        32
##  8        37
##  9        42
## 10        47
## 11        53
```

```
## 12        58
## 13        63
## 14        68
## 15        73
## 16        79
## 17        84
## 18        89
## 19        94
## 20       100
```

```
## Fit the model to cross validation resamples and tune neighbors
knearest_spotify_tune <- tune_grid(knearest_spotify,
                                   preprocessor = recipe(track_genre ~ . ,
                                                         data = pre_spotify_train),
                                   resamples = spotify_cv_10,
                                   grid = k_grid)

## Get average accuracy and average AUC for each value of neighbor
knearest_spotify_tune %>% collect_metrics()
```

```
## # A tibble: 40 x 7
##    neighbors .metric  .estimator  mean     n std_err .config
##        <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1         1 accuracy multiclass 0.428    10 0.00686 Preprocessor1_Model01
##  2         1 roc_auc  hand_till  0.657    10 0.00412 Preprocessor1_Model01
##  3         6 accuracy multiclass 0.444    10 0.00779 Preprocessor1_Model02
##  4         6 roc_auc  hand_till  0.773    10 0.00406 Preprocessor1_Model02
##  5        11 accuracy multiclass 0.474    10 0.00910 Preprocessor1_Model03
##  6        11 roc_auc  hand_till  0.795    10 0.00412 Preprocessor1_Model03
##  7        16 accuracy multiclass 0.488    10 0.00692 Preprocessor1_Model04
##  8        16 roc_auc  hand_till  0.806    10 0.00368 Preprocessor1_Model04
##  9        21 accuracy multiclass 0.494    10 0.00768 Preprocessor1_Model05
## 10        21 roc_auc  hand_till  0.812    10 0.00360 Preprocessor1_Model05
## # i 30 more rows
```

```
## Get the value of neighbors which can give the highest accuracy
k_best <- select_best(knearest_spotify_tune, "accuracy")
k_best
```

```
## # A tibble: 1 x 2
##   neighbors .config
##       <int> <chr>
## 1        58 Preprocessor1_Model12
```

```
## Finalize the model with the best neighbors value
final_knearest <- finalize_model(knearest_spotify, k_best)
final_knearest
```

```
## K-Nearest Neighbor Model Specification (classification)
##
## Main Arguments:
##   neighbors = 58
##
## Computational engine: kknn
```

```
## Get average accuracy and average AUC of the best neighbors value
knearest_spotify_tune %>% collect_metrics() %>% filter(neighbors == k_best$neighbors)
```

```
## # A tibble: 2 x 7
##   neighbors .metric  .estimator  mean     n std_err .config
##       <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1        58 accuracy multiclass 0.519    10 0.00587 Preprocessor1_Model12
## 2        58 roc_auc  hand_till  0.824    10 0.00392 Preprocessor1_Model12
```

```
# A random forest with 100 trees and 5 levels
## Set model specification
rand_spotify <- rand_forest(mode = "classification", trees = 100, min_n = tune(),
                            mtry = tune()) %>%
  set_engine("ranger", importance = "permutation")

## Create a grid containing possible combinations of mtry and min_n
mtry_tune <- grid_regular(finalize(mtry(), pre_spotify_train %>%
                                     dplyr::select(-track_genre)),
                          min_n(),
                          levels = 5)

## Fit the model to cross validation resamples, and tune mtry and min_n
rand_spotify_fit <- tune_grid(rand_spotify,
                              preprocessor = recipe(track_genre ~.,
                                                    data = pre_spotify_train),
                              resamples = spotify_cv_10,
                              grid = mtry_tune)

## Get average accuracy and average AUC for each combination of mtry and min_n
rand_spotify_fit %>% collect_metrics()
```

```
## # A tibble: 50 x 8
##     mtry min_n .metric  .estimator  mean     n std_err .config
##    <int> <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1     1     2 accuracy multiclass 0.513    10 0.00735 Preprocessor1_Model01
##  2     1     2 roc_auc  hand_till  0.818    10 0.00406 Preprocessor1_Model01
##  3    11     2 accuracy multiclass 0.556    10 0.00418 Preprocessor1_Model02
##  4    11     2 roc_auc  hand_till  0.853    10 0.00232 Preprocessor1_Model02
##  5    22     2 accuracy multiclass 0.554    10 0.00458 Preprocessor1_Model03
##  6    22     2 roc_auc  hand_till  0.854    10 0.00301 Preprocessor1_Model03
##  7    32     2 accuracy multiclass 0.557    10 0.00532 Preprocessor1_Model04
##  8    32     2 roc_auc  hand_till  0.853    10 0.00295 Preprocessor1_Model04
##  9    43     2 accuracy multiclass 0.558    10 0.00584 Preprocessor1_Model05
## 10    43     2 roc_auc  hand_till  0.850    10 0.00302 Preprocessor1_Model05
## # i 40 more rows
```

```
## Get the combination of mtry and min_n which can give the highest accuracy
mtry_best <- select_best(rand_spotify_fit, "accuracy")
mtry_best
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
```

```
##   <int> <int> <chr>
## 1    11    21 Preprocessor1_Model12
```

```
## Finalize the model with the best combination of mtry and min_n
final_randforest <- finalize_model(rand_spotify, mtry_best)
final_randforest
```

```
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = 11
##   trees = 100
##   min_n = 21
##
## Engine-Specific Arguments:
##   importance = permutation
##
## Computational engine: ranger
```

```
## Get average accuracy and average AUC of the best combination of mtry and min_n
rand_spotify_fit %>% collect_metrics() %>% filter(mtry == mtry_best$mtry,
                                                  min_n == mtry_best$min_n)
```

```
## # A tibble: 2 x 8
##    mtry min_n .metric  .estimator  mean     n std_err .config
##   <int> <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1    11    21 accuracy multiclass 0.561    10 0.00504 Preprocessor1_Model12
## 2    11    21 roc_auc  hand_till  0.858    10 0.00348 Preprocessor1_Model12
```

```
# Random forest is the best model, compared to other models
# fit random forest model to processed training set
final_randforest_fit <- final_randforest %>% fit(track_genre ~., pre_spotify_train)

# Make class predictions on processed testing set
spotify_pred_class <- predict(final_randforest_fit, pre_spotify_test, type = "class") %>%
  bind_cols(pre_spotify_test)
```

```
# Make confusion matrix based on class predictions
spotify_pred_class %>% conf_mat(truth = track_genre, estimate = .pred_class)
```

```
##           Truth
## Prediction edm latin pop r&b rap rock
##      edm   171    18  28  12  21   12
##      latin  18   100  38  23  23    5
##      pop    38    41 117  34  22   20
##      r&b     7    27  23 108  32   12
##      rap    10    36  22  47 147    8
##      rock    3    15  28  28   7  199
```

```
# Calculate average sensitivity for track's genres
spotify_pred_class %>% sens(truth = track_genre, estimate = .pred_class, estimator = "macro")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    macro          0.560
```

```
## Calculate sensitivity for edm
spotify_pred_class %>% mutate(track_genre =
                                case_when(track_genre == "edm" ~ 0, TRUE ~ 1),
                              .pred_class =
                                case_when(.pred_class == "edm" ~ 0, TRUE ~ 1),
                              track_genre = factor(track_genre,
                                                   ordered = is.ordered(c(1,0))),
                              .pred_class = factor(.pred_class,
                                                   ordered =  is.ordered(c(1,0)))) %>%
  sens(truth = track_genre, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.692
```

```
## Calculate sensitivity for latin
spotify_pred_class %>% mutate(track_genre =
                                case_when(track_genre == "latin" ~ 0, TRUE ~ 1),
                              .pred_class =
                                case_when(.pred_class == "latin" ~ 0, TRUE ~ 1),
                              track_genre = factor(track_genre,
                                                   ordered = is.ordered(c(1,0))),
                              .pred_class = factor(.pred_class,
                                                   ordered = is.ordered(c(1,0)))) %>%
  sens(truth = track_genre, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.422
```

```
## Calculate sensitivity for pop
spotify_pred_class %>% mutate(track_genre =
                                case_when(track_genre == "pop" ~ 0, TRUE ~ 1),
                              .pred_class =
                                case_when(.pred_class == "pop" ~ 0, TRUE ~ 1),
                              track_genre = factor(track_genre,
                                                   ordered = is.ordered(c(1,0))),
                              .pred_class = factor(.pred_class,
                                                   ordered = is.ordered(c(1,0)))) %>%
  sens(truth = track_genre, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.457
```

```r
## Calculate sensitivity for r&b
spotify_pred_class %>% mutate(track_genre =
                                case_when(track_genre == "r&b" ~ 0, TRUE ~ 1),
                              .pred_class =
                                case_when(.pred_class == "r&b" ~ 0, TRUE ~ 1),
                              track_genre = factor(track_genre,
                                                   ordered = is.ordered(c(1,0))),
                              .pred_class = factor(.pred_class,
                                                   ordered = is.ordered(c(1,0)))) %>%
  sens(truth = track_genre, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.429
```

```r
## Calculate sensitivity for rap
spotify_pred_class %>% mutate(track_genre =
                                case_when(track_genre == "rap" ~ 0, TRUE ~ 1),
                              .pred_class =
                                case_when(.pred_class == "rap" ~ 0, TRUE ~ 1),
                              track_genre = factor(track_genre,
                                                   ordered = is.ordered(c(1,0))),
                              .pred_class = factor(.pred_class,
                                                   ordered = is.ordered(c(1,0)))) %>%
  sens(truth = track_genre, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.583
```

```r
## Calculate sensitivity for rock
spotify_pred_class %>% mutate(track_genre =
                                case_when(track_genre == "rock" ~ 0, TRUE ~ 1),
                              .pred_class =
                                case_when(.pred_class == "rock" ~ 0, TRUE ~ 1),
                              track_genre = factor(track_genre,
                                                   ordered = is.ordered(c(1,0))),
                              .pred_class = factor(.pred_class,
                                                   ordered = is.ordered(c(1,0)))) %>%
  sens(truth = track_genre, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 sens    binary         0.777
```

```r
# Calculate average specificity for track's genres
spotify_pred_class %>% spec(truth = track_genre, estimate = .pred_class, estimator = "macro")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    macro          0.912
```

```
## Calculate specificity for edm
spotify_pred_class %>% mutate(track_genre =
                                case_when(track_genre == "edm" ~ 0, TRUE ~ 1),
                              .pred_class =
                                case_when(.pred_class == "edm" ~ 0, TRUE ~ 1),
                              track_genre = factor(track_genre,
                                                   ordered = is.ordered(c(1,0))),
                              .pred_class = factor(.pred_class,
                                                   ordered = is.ordered(c(1,0)))) %>%
  spec(truth = track_genre, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    binary         0.927
```

```
## Calculate specificity for latin
spotify_pred_class %>% mutate(track_genre =
                                case_when(track_genre == "latin" ~ 0, TRUE ~ 1),
                              .pred_class =
                                case_when(.pred_class == "latin" ~ 0, TRUE ~ 1),
                              track_genre = factor(track_genre,
                                                   ordered = is.ordered(c(1,0))),
                              .pred_class = factor(.pred_class,
                                                   ordered = is.ordered(c(1,0)))) %>%
  spec(truth = track_genre, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    binary         0.915
```

```
## Calculate specificity for pop
spotify_pred_class %>% mutate(track_genre =
                                case_when(track_genre == "pop" ~ 0, TRUE ~ 1),
                              .pred_class =
                                case_when(.pred_class == "pop" ~ 0, TRUE ~ 1),
                              track_genre = factor(track_genre,
                                                   ordered = is.ordered(c(1,0))),
                              .pred_class = factor(.pred_class,
                                                   ordered = is.ordered(c(1,0)))) %>%
  spec(truth = track_genre, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    binary         0.875
```

```r
## Calculate specificity for r&b
spotify_pred_class %>% mutate(track_genre =
                                case_when(track_genre == "r&b" ~ 0, TRUE ~ 1),
                              .pred_class =
                                case_when(.pred_class == "r&b" ~ 0, TRUE ~ 1),
                              track_genre = factor(track_genre,
                                                   ordered = is.ordered(c(1,0))),
                              .pred_class = factor(.pred_class,
                                                   ordered = is.ordered(c(1,0)))) %>%
  spec(truth = track_genre, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    binary         0.919
```

```r
## Calculate specificity for rap
spotify_pred_class %>% mutate(track_genre =
                                case_when(track_genre == "rap" ~ 0, TRUE ~ 1),
                              .pred_class =
                                case_when(.pred_class == "rap" ~ 0, TRUE ~ 1),
                              track_genre = factor(track_genre,
                                                   ordered = is.ordered(c(1,0))),
                              .pred_class = factor(.pred_class,
                                                   ordered = is.ordered(c(1,0)))) %>%
  spec(truth = track_genre, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    binary         0.901
```

```r
## Calculate specificity for rock
spotify_pred_class %>% mutate(track_genre =
                                case_when(track_genre == "rock" ~ 0, TRUE ~ 1),
                              .pred_class =
                                case_when(.pred_class == "rock" ~ 0, TRUE ~ 1),
                              track_genre = factor(track_genre,
                                                   ordered = is.ordered(c(1,0))),
                              .pred_class = factor(.pred_class,
                                                   ordered = is.ordered(c(1,0)))) %>%
  spec(truth = track_genre, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 spec    binary         0.935
```

```r
# Make probability predictions on processed testing set
spotify_pred_prob <- predict(final_randforest_fit, pre_spotify_test, type = "prob") %>%
  bind_cols(pre_spotify_test)
```

```
# Draw roc curves for each track's genre
spotify_pred_prob %>% roc_curve(truth = track_genre, estimate = 1:6) %>% autoplot()
```

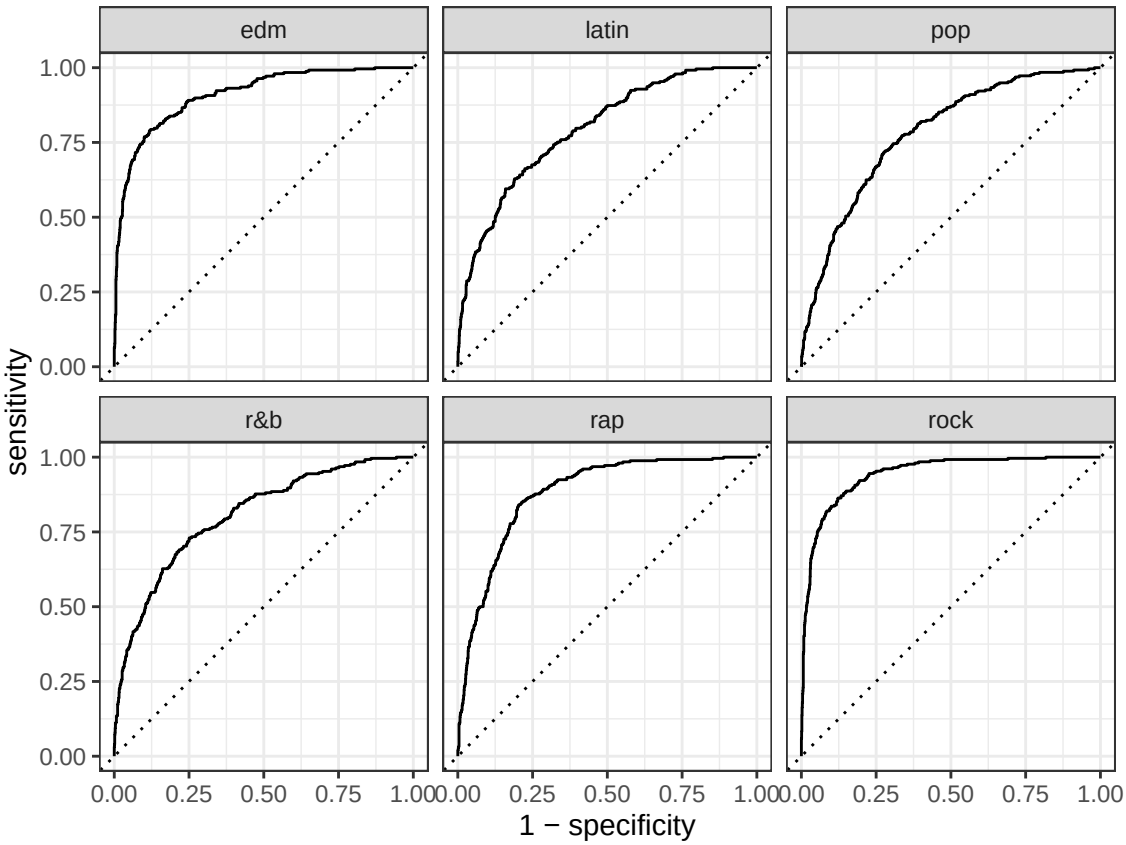


Figure 29: ROC curves for each track's genre

```
# Calculate average AUC for track's genres
spotify_pred_prob %>% roc_auc(truth = track_genre, estimate = 1:6,
                              estimator = "hand_till")
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.852
```

```
# Get variable importance for each predictor
final_randforest_fit %>% vip(num_features = 50)
```

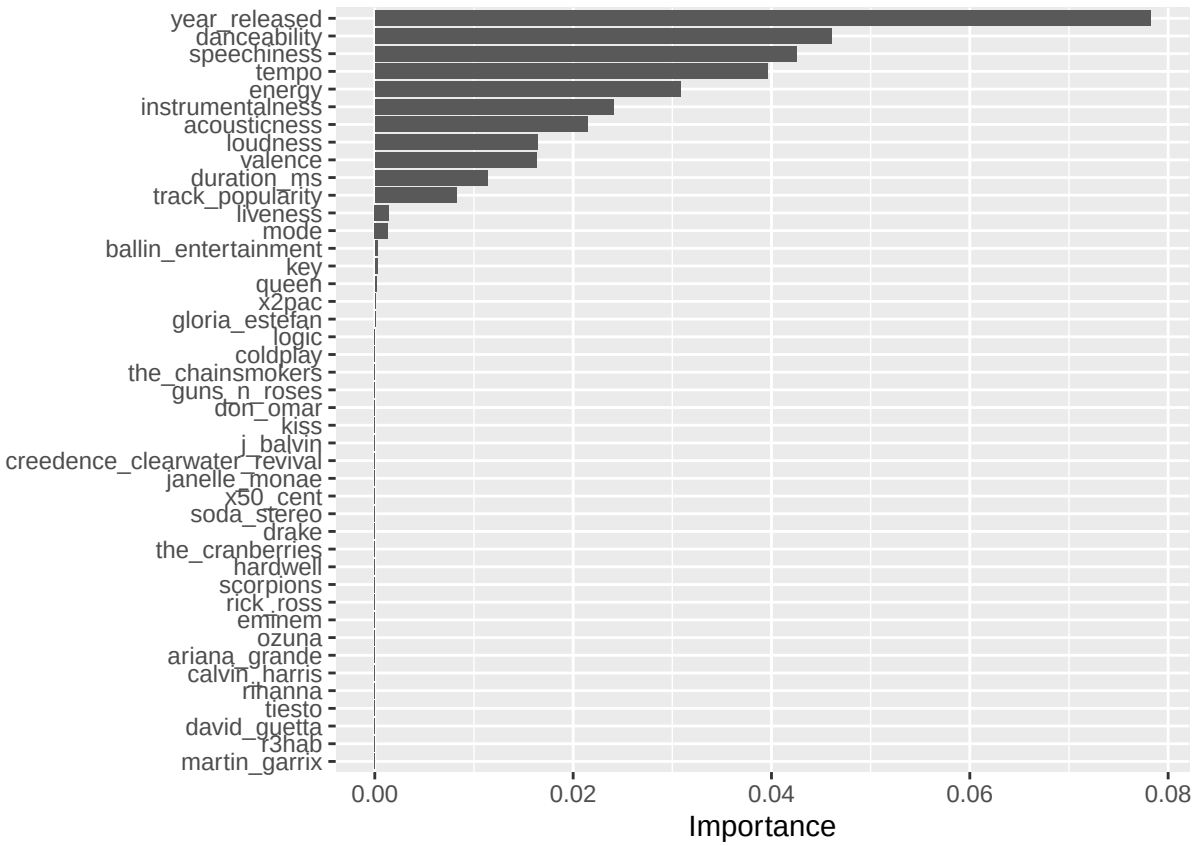


Figure 30: Variable importance of each predictor