Modeling Spotify User Behaviour

Luka Vukovic

11/05/2021

```r
library(tidyverse)
library(data.table)
library(glue)
library(lubridate)

# PCA
library("FactoMineR")
library("factoextra")
#library(psych)

# For visualization
library(cowplot)
library(ggplot2)
library(quantreg)
library(tidyquant) ## rolling average for geom_smooth
library(grid)

# Themes
library(ggthemes)
library(egg)
library(RColorBrewer)
```

Data Exploration

```r
# User 1
usr1_lib <- read.csv('Enhanced_Data/User1/usr1_lib.csv')
usr1_streams <- read.csv('Enhanced_Data/User1/usr1_streams.csv')

# User 2
usr2_lib <- read.csv('Enhanced_Data/User2/usr2_lib.csv')
usr2_streams <- read.csv('Enhanced_Data/User2/usr2_streams.csv')

data <- list(usr1_lib, usr1_streams, usr2_lib, usr2_streams)
```

- Most undefined rows that the API couldn't retrieve information for have been removed in the original data acquisition file.
- It seems just a few rows still contain na values but will be removed.
- Our cleanup function also scales the data for later processing.

```r
cleanup <- function(df) {
  # Keep only rows without NA values
  df <- df[!rowSums(is.na(df)) >= 1, ]
```

```r
  df
}

# Rows including NA
unlist(lapply(data, nrow))
```

```
## [1] 29980 55278  6806 52283
```

```r
# Rows excluding NA
clean_data <- lapply(data, cleanup)
unlist(lapply(clean_data, nrow))
```

```
## [1] 29979 55278  6790 52267
```

```r
top_100_songs <- function(streaming_frame) {
  summary_streams <- streaming_frame %>%
    group_by(trackName, artistName, # main vars
             danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, li
    summarise(plays = n(), hoursPlayed = sum(msPlayed)/(1000*60*60))

  top_tracks <- head(summary_streams[order(summary_streams$hoursPlayed, decreasing = TRUE), ], 100)
  top_tracks
}

top_100_genres <- function(lib_frame) {
  summary_lib <- lib_frame %>%
    group_by(genres) %>%
    summarise(instances = n())

  # need n_songs to get genre occurence across the library
  songs <- lib_frame %>%
    group_by(id) %>%
    summarise()

  summary_lib$`%occurence` <- round(100*summary_lib$instances / nrow(songs), 2)
  top_genres <- head(summary_lib[order(summary_lib$instances, decreasing = TRUE), ], 100)
  top_genres
}

top_100_songs(clean_data[[2]])
```

```
## `summarise()` has grouped output by 'trackName', 'artistName', 'danceability', 'energy', 'key', 'lou
```

```
## # A tibble: 100 x 16
## # Groups:   trackName, artistName, danceability, energy, key, loudness, mode,
## #   speechiness, acousticness, instrumentalness, liveness, valence, tempo [100]
##    trackName    artistName  danceability energy   key loudness  mode speechiness
##    <chr>        <chr>              <dbl>  <dbl> <int>    <dbl> <int>       <dbl>
##  1 Marigold     Periphery          0.788  0.533     4    -8.81     1      0.0573
##  2 Doomsday     Architects         0.856  0.841     6    -7.74     0      0.335
##  3 Reptile      Periphery          0.277  0.91      7    -5.72     1      0.117
##  4 Hereafter    Architects         0.513  0.973     6    -4.12     0      0.0849
##  5 Blood Eagle  Periphery          0.492  0.982     1    -5.25     1      0.147
##  6 Ludens       BringMeThe~        0.295  0.679     1    -6.18     1      0.15
##  7 Satellites   Periphery          0.669  0.9      10    -3.53     0      0.178
##  8 Stranded     Gojira             0.513  0.88      7    -4.69     0      0.0309
```

```
##   9 Take Me Out  FranzFerdi~        0.277  0.663     4    -8.82     0      0.0377
##  10 The Hand Th~ NineInchNa~        0.587  0.99      0    -4.50     1      0.0783
## # ... with 90 more rows, and 8 more variables: acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   time_signature <int>, plays <int>, hoursPlayed <dbl>
```

```r
top_100_songs(clean_data[[4]])
```

```
## `summarise()` has grouped output by 'trackName', 'artistName', 'danceability', 'energy', 'key', 'lou

## # A tibble: 100 x 16
## # Groups:   trackName, artistName, danceability, energy, key, loudness, mode,
## #   speechiness, acousticness, instrumentalness, liveness, valence, tempo [100]
##    trackName     artistName danceability energy   key loudness  mode speechiness
##    <chr>         <chr>             <dbl>  <dbl> <int>    <dbl> <int>       <dbl>
##  1 Summer 99     Tchami            0.81   0.968     2    -3.98     1      0.103
##  2 Red Light Gr~ DukeDumont        0.811  0.825     2    -5.48     1      0.0654
##  3 Deceiver      ChrisLake         0.881  0.883    11    -5.49     1      0.0633
##  4 Brothers In ~ DireStrai~        0.22   0.44      8    -9.46     0      0.0352
##  5 Conjure Drea~ MaceoPlex         0.687  0.744     2    -8.09     1      0.0545
##  6 Lies, Decept~ ChrisLake         0.809  0.935     1    -4.81     1      0.06
##  7 IM GONE       JOYRYDE           0.253  0.676    10    -6.13     1      0.0455
##  8 Walk          Pantera           0.871  0.701     5    -5.59     0      0.0458
##  9 San Frandisc~ DomDolla          0.762  0.826     1    -4.87     1      0.0406
## 10 Never Let Yo~ SNBRN             0.728  0.94      4    -5.03     1      0.0368
## # ... with 90 more rows, and 8 more variables: acousticness <dbl>,
## #   instrumentalness <dbl>, liveness <dbl>, valence <dbl>, tempo <dbl>,
## #   time_signature <int>, plays <int>, hoursPlayed <dbl>
```

```r
top_100_genres(clean_data[[1]])
```

```
## # A tibble: 100 x 3
##    genres             instances `%occurence`
##    <chr>                  <int>        <dbl>
##  1 alternative metal      1859         30.4
##  2 rock                   1712         28
##  3 nu metal               1616         26.4
##  4 modern rock            1443         23.6
##  5 pop punk               1120         18.3
##  6 metalcore               958         15.7
##  7 post-grunge             751         12.3
##  8 screamo                 727         11.9
##  9 melodic metalcore       672         11.0
## 10 alternative rock        658         10.8
## # ... with 90 more rows
```

```r
top_100_genres(clean_data[[3]])
```

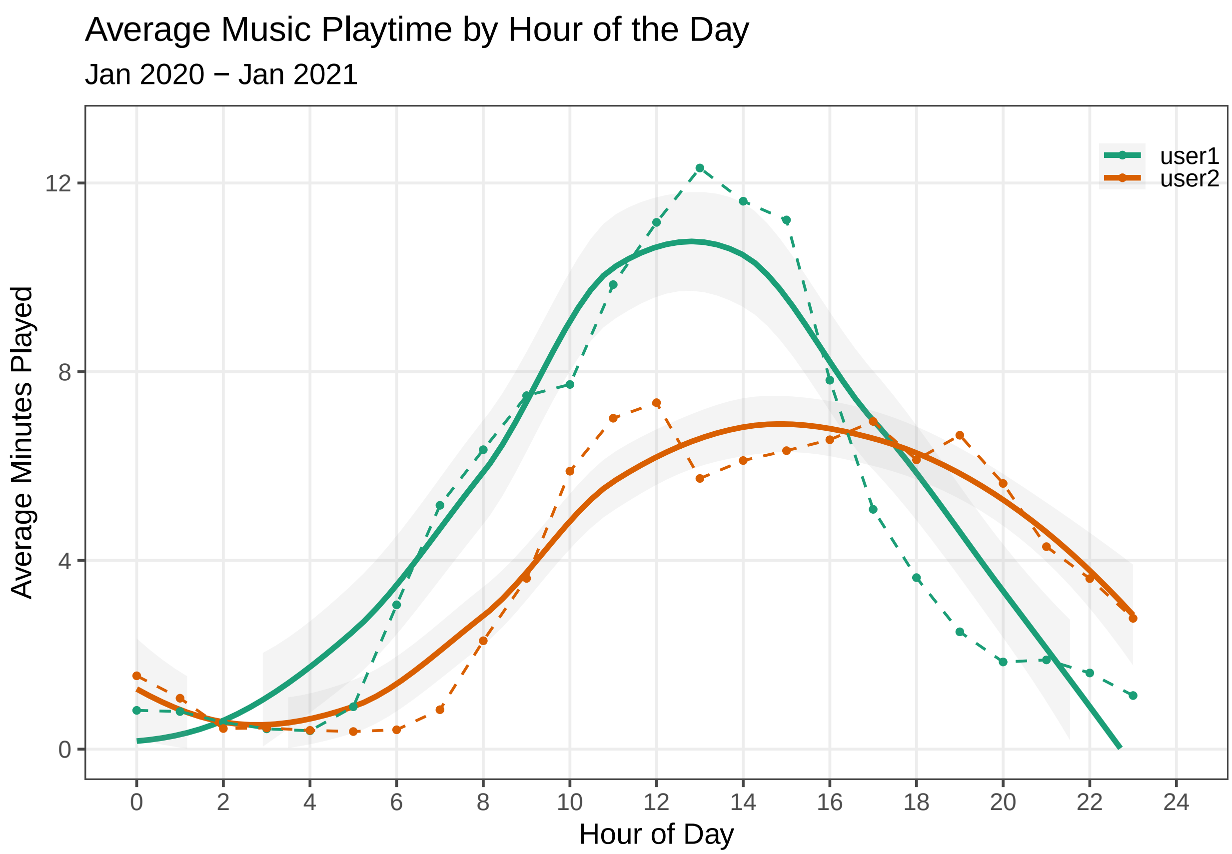
```
## # A tibble: 100 x 3
##    genres          instances `%occurence`
##    <chr>               <int>        <dbl>
##  1 edm                   337        19.9
##  2 electro house         313        18.5
##  3 pop dance             218        12.9
##  4 rap                   166         9.82
##  5 hip hop               151         8.93
##  6 Unknown               142         8.4
```

```
##   7 pop                  140         8.28
##   8 tropical house       139         8.22
##   9 house                121         7.16
##  10 pop rap              108         6.39
## # ... with 90 more rows
```

Visualizing Average Daily Play Time

```
# Function for binding user streaming data
stream_together <- function(frames) {
  n <- length(frames)
  for (i in 1:n) {
    frames[[i]] <- frames[[i]] %>%
      group_by(endTime, trackName, artistName, msPlayed, # main vars
               danceability, energy, key, loudness, mode,
               speechiness, acousticness, instrumentalness,
               liveness, valence, tempo, time_signature) %>%
      summarise()

    frames[[i]]$minPlayed <- frames[[i]]$msPlayed/(1000*60)
    frames[[i]] <- frames[[i]][-4]
    frames[[i]]$user <- as.factor(glue('user{i}'))
  }
  frames <- rbindlist(frames)
  frames
}

all_streams_UTC <- stream_together(clean_data[c(2,4)]) # Universal Standard Time

# Manual adjustment for user timezones
# This can only be done if one knows where the user has been when
all_streams <- rbind(
  # User 2 Ontario
  all_streams_UTC %>% filter(user == "user2" & month(endTime) < 9) %>%
    mutate(endHour = (hour(endTime)-5) %% 24, endDate = date(endTime), week_of_year = week(date(endTime
  # User 2 British Columbia
  all_streams_UTC %>% filter(user == "user2" & month(endTime) >= 9) %>%
    mutate(endHour = (hour(endTime)-8) %% 24, endDate = date(endTime), week_of_year = week(date(endTime
  # User 1 Ontario
  all_streams_UTC %>% filter(user == "user1" & month(endTime) < 5) %>%
    mutate(endHour = (hour(endTime)-5) %% 24, endDate = date(endTime), week_of_year = week(date(endTime
  # User 1 British Columbia
  all_streams_UTC %>% filter(user == "user1" & month(endTime) >= 5) %>%
    mutate(endHour = (hour(endTime)-8) %% 24, endDate = date(endTime), week_of_year = week(date(endTime
  )


# Plotting
tot_days <- as.numeric(max(all_streams$endDate) - min(all_streams$endDate))

all_streams %>%
  group_by(endHour, endDate, user) %>%
  summarise(hourdayPlayTime = sum(minPlayed)) %>% # total minutes played by hour by day
  group_by(endHour, user) %>%
```

```
summarise(avg = sum(hourdayPlayTime)/tot_days) %>% # avg minutes played by hour over all days
          #n = n())  # days played at given hour
          #q1 = quantile(c(hourdayPlayTime, rep(0, tot_days-n())), 0.5),  # quantiles including days
          #q3 = quantile(c(hourdayPlayTime, rep(0, tot_days-n())), 0.8)) %>%

ggplot(aes(y = avg, x = endHour, color = user)) +

geom_smooth(level = 0.95, method = loess, alpha = 0.1) +
geom_line(alpha = 1, size = 0.5, linetype  = 'dashed') +
geom_point(alpha = 1.2, shape = 16, size=1.2) +

scale_x_continuous(name = 'Hour of Day', breaks=seq(0,24,2), limits = c(0,24)) +
scale_y_continuous(name = 'Average Minutes Played', breaks=seq(0, 20, 4), limits = c(0,13)) +
ggtitle('Average Music Playtime by Hour of the Day', subtitle = 'Jan 2020 - Jan 2021') +
scale_color_brewer(palette = 'Dark2') +
theme_article() +
theme(legend.title = element_blank(),
      legend.position = c(0.94,0.92),
      panel.grid.major = element_line(color = '#ededed'))
```

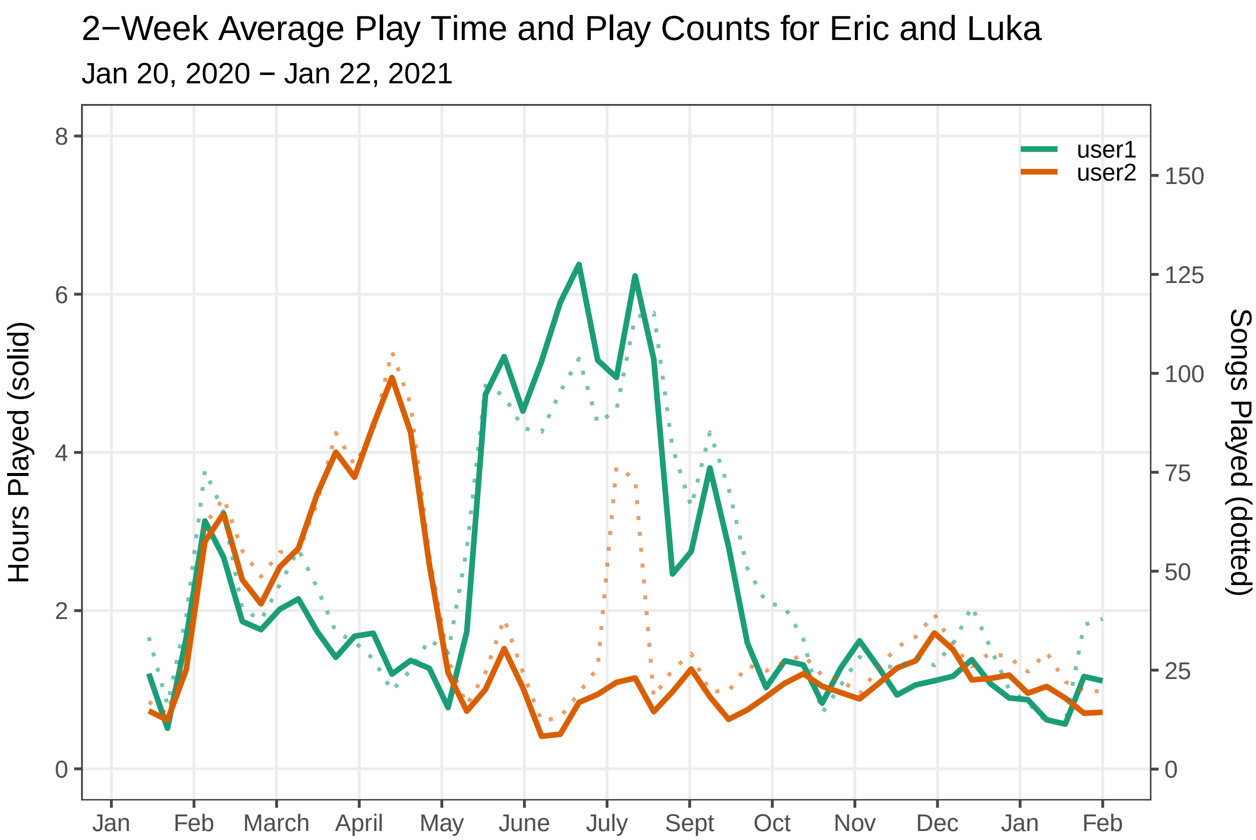


Visualizing Play Time through the Year

```
months <- c('Jan', 'Feb', 'March', 'April', 'May', 'June', 'July', 'Sept',
            'Oct', 'Nov', 'Dec', 'Jan', 'Feb')
```

```r
tot_weeks <- length(unique(all_streams$week_of_year))

all_streams %>%
  group_by(week_of_year, endDate, user) %>%
  summarise(dailyPlayTime = sum(minPlayed)/60,
            dailyPlays = n()) %>%
  group_by(week_of_year, user) %>%
  summarise(avg_weeklyPlayTime = mean(dailyPlayTime),
            avg_weeklyPlays = mean(dailyPlays)) %>%

  ggplot(aes(x = week_of_year, color = user)) +

  #geom_line(aes(y=avg_weeklyPlayTime), linetype = 1, size = 1) +
  #geom_line(aes(y=avg_weeklyPlays/16), linetype = 3, size = 0.7) +

  geom_ma(aes(y=avg_weeklyPlayTime), n = 2, linetype = 1, size = 1, alpha = 1.0) +
  geom_ma(aes(y=avg_weeklyPlays/20), n = 2, linetype = 3, size = 0.7, alpha = 0.6) +

  scale_x_continuous(name = element_blank(), labels = months, breaks = seq(0, 57, 4.4167)) +

  scale_y_continuous(name = 'Hours Played (solid)', limits = c(0,8), breaks = seq(0,10,2),
                     sec.axis = sec_axis(name = 'Songs Played (dotted)', ~.*20, breaks = seq(0,200,25),

  ggtitle('2-Week Average Play Time and Play Counts for Eric and Luka',
          subtitle = 'Jan 20, 2020 - Jan 22, 2021') +

  scale_color_brewer(palette = 'Dark2') +

  theme_article() +

  theme(legend.title = element_blank(),
        legend.position = c(0.93,0.93),
        axis.title.y.left = element_text(vjust = 3),
        axis.title.y.right = element_text(vjust = 3, angle = -90),
        panel.grid.major = element_line(color = '#ededed'))
```

```
## `summarise()` has grouped output by 'week_of_year', 'endDate'. You can override using the `.groups` a

## `summarise()` has grouped output by 'week_of_year'. You can override using the `.groups` argument.
```

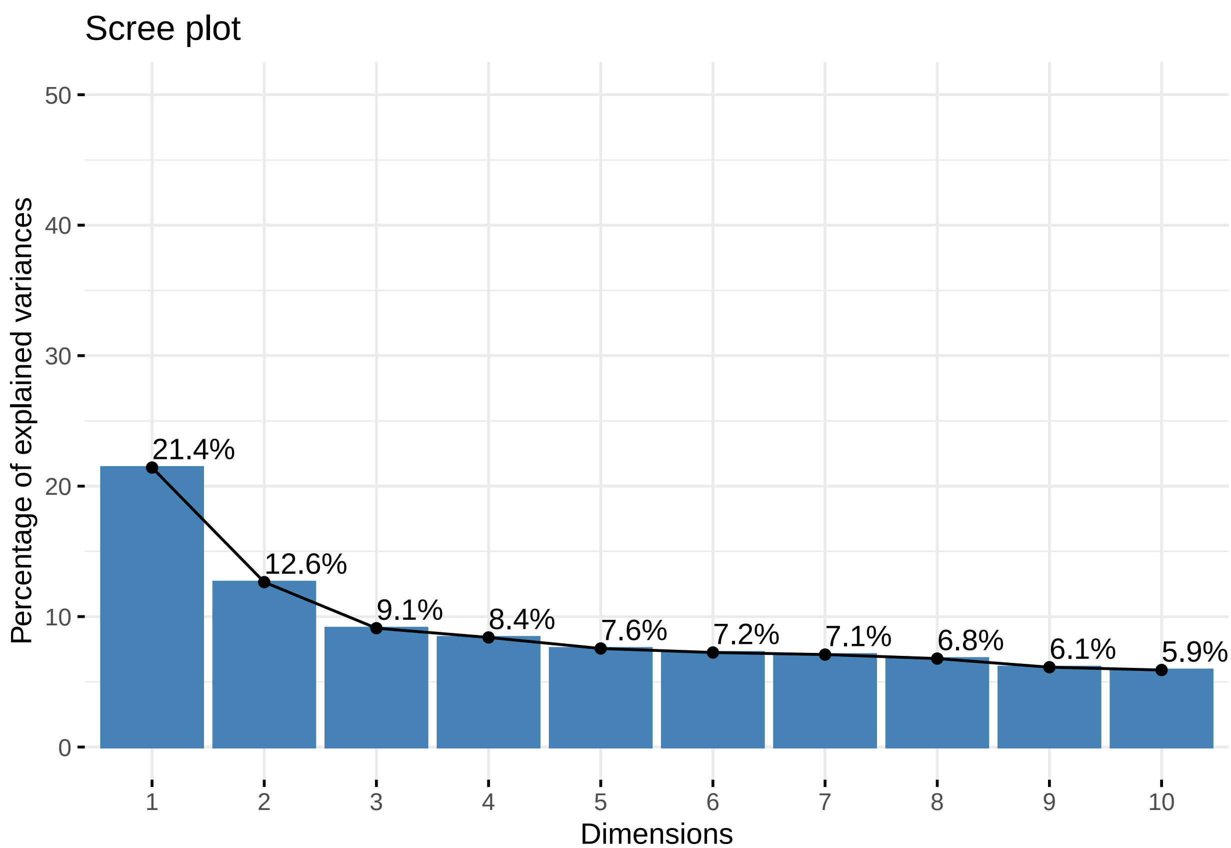


Unsupervised Exploration to find Emotional Groupings in Libraries

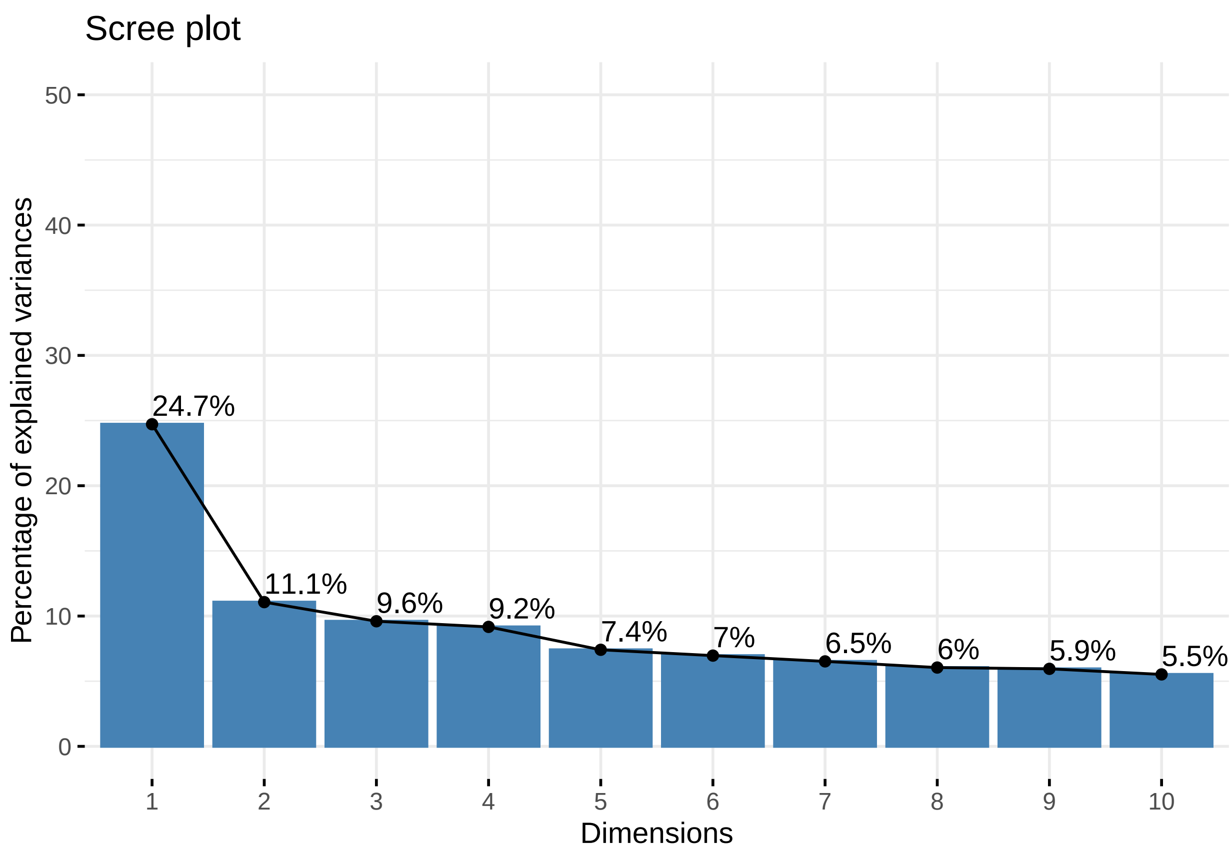
- Genres do not form clear clusters
- PC1 is related to energy/positivity/loudness in one direction and calmness in the other.
- PC2 is related to duration/energy in one direction and mode in the other.
- Ultimately, we won't be able to cluster music tastes into specific subgroups for either user.
- Perhaps clusters may appear when comparing user data but it's not clear if they exist within user data.
- Therefore, further clustering methods won't be performed.

```
# Function to scale and remove non-numeric rows
convenient_scale <- function(df) {
  # Save genre classes
  genres <- as.factor(df$genres)
  # Scale numeric columns
  df <- df %>% mutate_if(is.numeric, scale) %>% select_if(is.numeric)
  # Readd genre classes
  df$genres <- cbind(genres, df)
}

# Scale data
usr1_lib_scaled <- convenient_scale(clean_data[[1]])
usr2_lib_scaled <- convenient_scale(clean_data[[3]])

# PCA
usr1_pca <- PCA(usr1_lib_scaled[,-1], graph = FALSE)
usr2_pca <- PCA(usr2_lib_scaled[,-1], graph = FALSE)
```

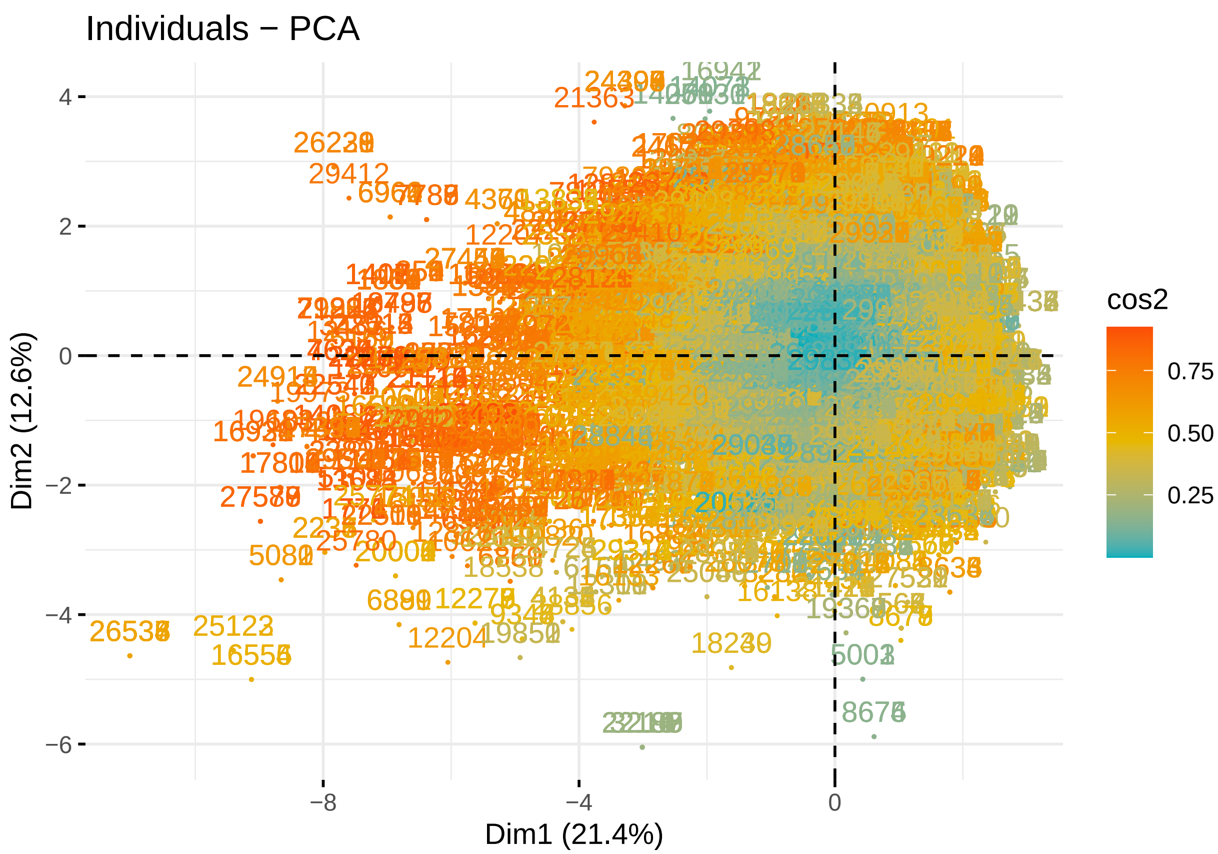
```r
# Skree plot
fviz_eig(usr1_pca, addlabels = TRUE, ylim = c(0, 50))
```



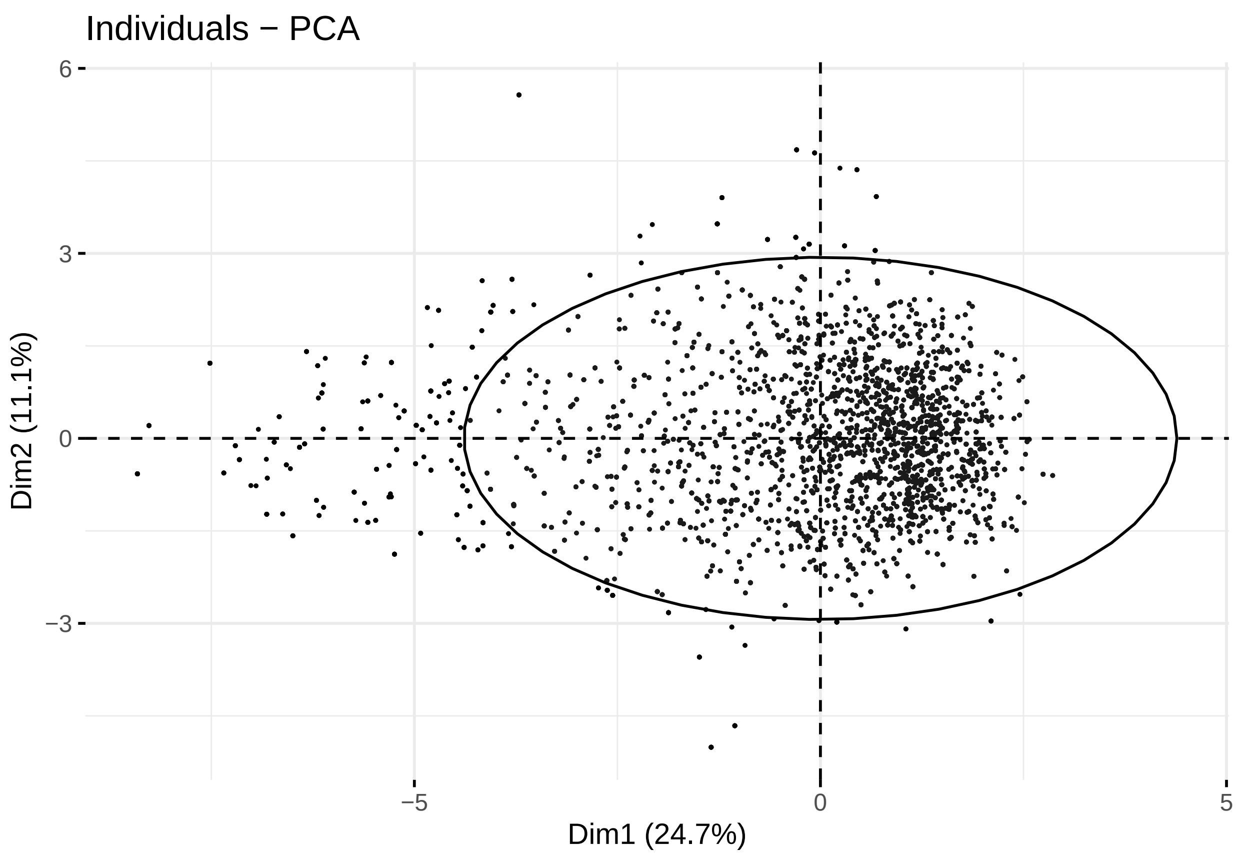

```r
fviz_eig(usr2_pca, addlabels = TRUE, ylim = c(0, 50))
```



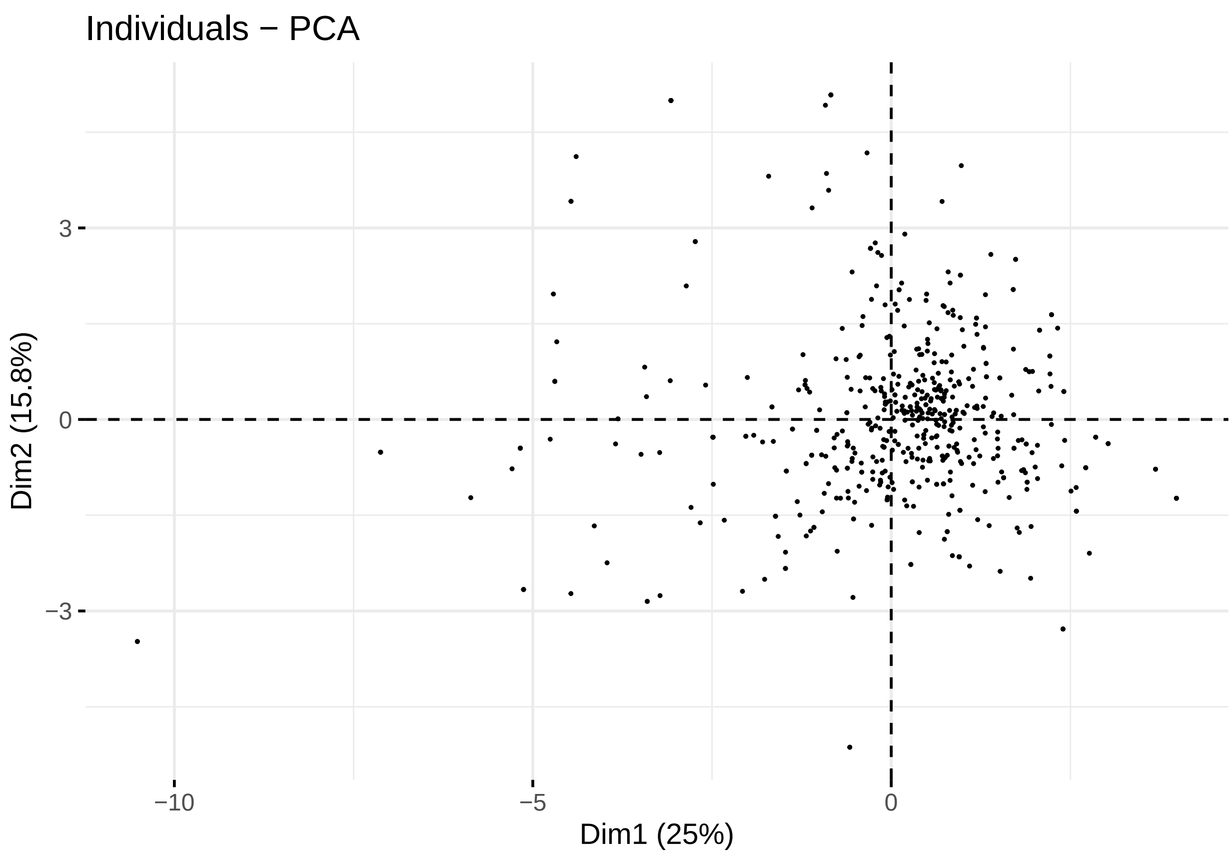

```r
fviz_pca_ind(usr1_pca, col.ind = "cos2", pointsize = 0.2,
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = FALSE) # Avoid text overlapping (slow if many points)
```




```r
fviz_pca_ind(usr2_pca, geom.ind = "point", pointsize = 0.2, addEllipses = TRUE,)
```




```
# Check if average genre features alone cluster based on audio features
# (summarize average audio features by genre)
gens <- usr1_lib_scaled %>%
  group_by(genres) %>%
  summarise_all(mean)

# PCA on average genre feature
gens_pca <- PCA(gens[,-1], ncp = 10, graph = FALSE)
fviz_pca_ind(gens_pca, geom.ind = "point", pointsize = 0.2)
```




```r
library(teigen)
```

```
## Warning: package 'teigen' was built under R version 4.0.3
```

```r
#obj1 <- teigen(gens_pca$ind$cos2[,1:2], Gs = 1:5, model = 'UCCU', scale = FALSE, verbose = FALSE)
#obj2 <- teigen(usr1_pca$ind$cos2[,1:2], Gs = 1:5, model = 'UCCU', scale = FALSE, verbose = FALSE)
#obj3 <- teigen(usr2_pca$ind$cos2[,1:2], Gs = 1:5, model = 'UCCU', scale = FALSE, verbose = FALSE)
#plot(obj1, xmarg=1, ymarg=7, what="contour")
#plot(obj2, xmarg=1, ymarg=7, what="contour")
#plot(obj3, xmarg=1, ymarg=7, what="contour")
```

Supervised Learning

Predicting genres on audio features

Modeling daily, weekly, and monthly listening habits

Modeling play time, and specific audio feature play time