top-spotify-tracks (Timos part)

Ila, Timo, Yun, Axel

12/25/2019

Environment preparation

```
# install pacman if not already installed
if(!require("pacman")) install.packages("pacman")
# check all required packages, install if necessary and then load them
pacman::p_load(ggplot2, GGally, corrplot, scales, gridExtra, tidyr,
               lubridate, grid, ggpubr, RColorBrewer, dplyr,
               ggforce, waffle, reshape2, cluster, fpc, NbClust,
               data.table, network, ggraph, factoextra)
```

Data input

```
# setting file path once (easier to change for testing on multiple machines)
csv_path = "data/"
# image_path = "images/"

# dataset imports into a data frame
spotify_data <- fread(paste(csv_path, "featuresdf.csv", sep=""))
daily_spotify <- fread(paste(csv_path, "data.csv", sep=""))
```

Timos analysis [TODO: great title here]

We will limit all analyses in the next sections to global streams and rankings.

```
# crop the dataset and make columns easier to read and use
ranks <- copy(daily_spotify)
features <- copy(spotify_data)

setnames(ranks,"Track Name", "Trackname")
ranks[,Date := as.Date(Date)]
setnames(features,"name", "Trackname")

ranks_glob <- ranks[Region=="global"]
ranks_glob[,c("Region","URL"):=NULL]
ranks_glob[,Trackname:=ifelse(grepl('Thomas Rhett', Artist), 'Unforgettable_2', Trackname)]
```

Here we add some helper functions for later use.

```
# returns all songs that were at a position <= pos at some point
getSongsAtPos <- function(dt, pos) {
  dt[Position<=pos,.N, by=Trackname][order(-N)][,Trackname]
}

# returns a data.table with days at or below the specified position for all songs
getDaysAtPos <- function(dt, pos)  {
```

```
  dt[Position<=pos,.N,by=Trackname][order(-N)][,.(Trackname, days_top = N)]
}

# for a vector of positions, get a data.table with the number of songs that were at least at
# these positions at some point
getCumSongs <- function(dt, pos_vec) {
  cum_songs <- data.table(Position = integer(), N_Songs = numeric())
  for(pos in pos_vec) {
    n_songs <- length(getSongsAtPos(dt, pos))
    cum_songs <- rbind(cum_songs,list(pos, n_songs))
  }
  return(cum_songs)
}

# perform a backward-selection algorithm to choose which variables are best
# used for the linear regression of the dep_var;
# you can set a critical p_value (crit_p) which all vars have to suffice
backwardSelection <- function(data, factors, dep_var, crit_p) {
  biggest <- 1
  while(biggest > crit_p && length(factors) > 0) {
    # recalculate formula with remaining factors
    formula <- as.formula(paste(paste0(dep_var, "~"), paste(factors, collapse="+")))
    model <- lm(formula, data)

    # Calculate var with highest p-value
    p_vals <- summary(model)$coefficients[,4]
    biggest <- 0
    for(i in 1:length(summary(model)$coefficients[,4])) {
      tmp_p_val <- p_vals[i]
      if(tmp_p_val > biggest) {
        biggest <- tmp_p_val
      }
    }

    factors <- factors[!factors %in% names(biggest)]
  }
  return(model)
}
```

Exploratory analysis of the data

First, we do some exploratory analysis of the dataset to get a feel how it looks like and maybe discover interesting trends to further evaluate.

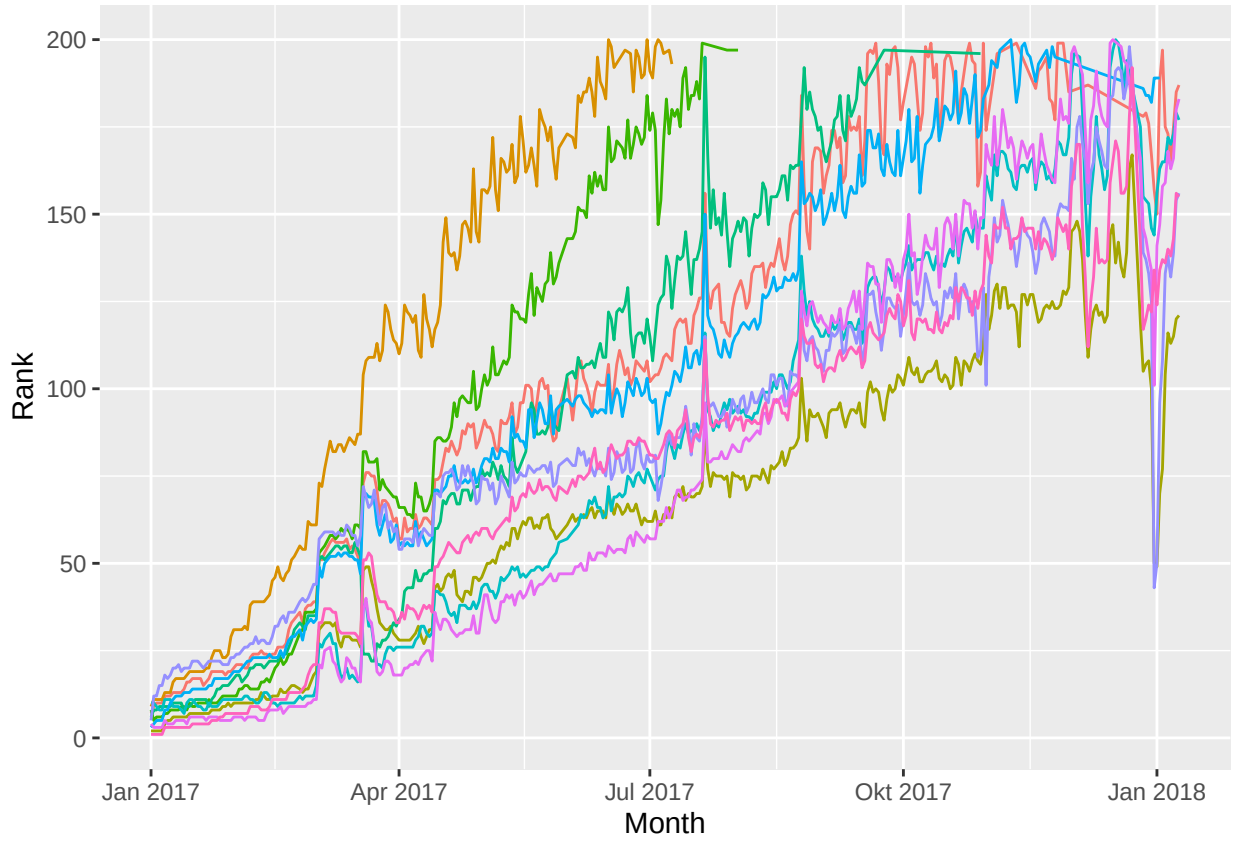
```
# get all songs that reached the top10
most_days_top10 <- getSongsAtPos(ranks_glob, 10)
# there are a total of 1307 songs in the data set
ranks_glob[,.N,by=Trackname][,.N]
```

```
## [1] 1307
```

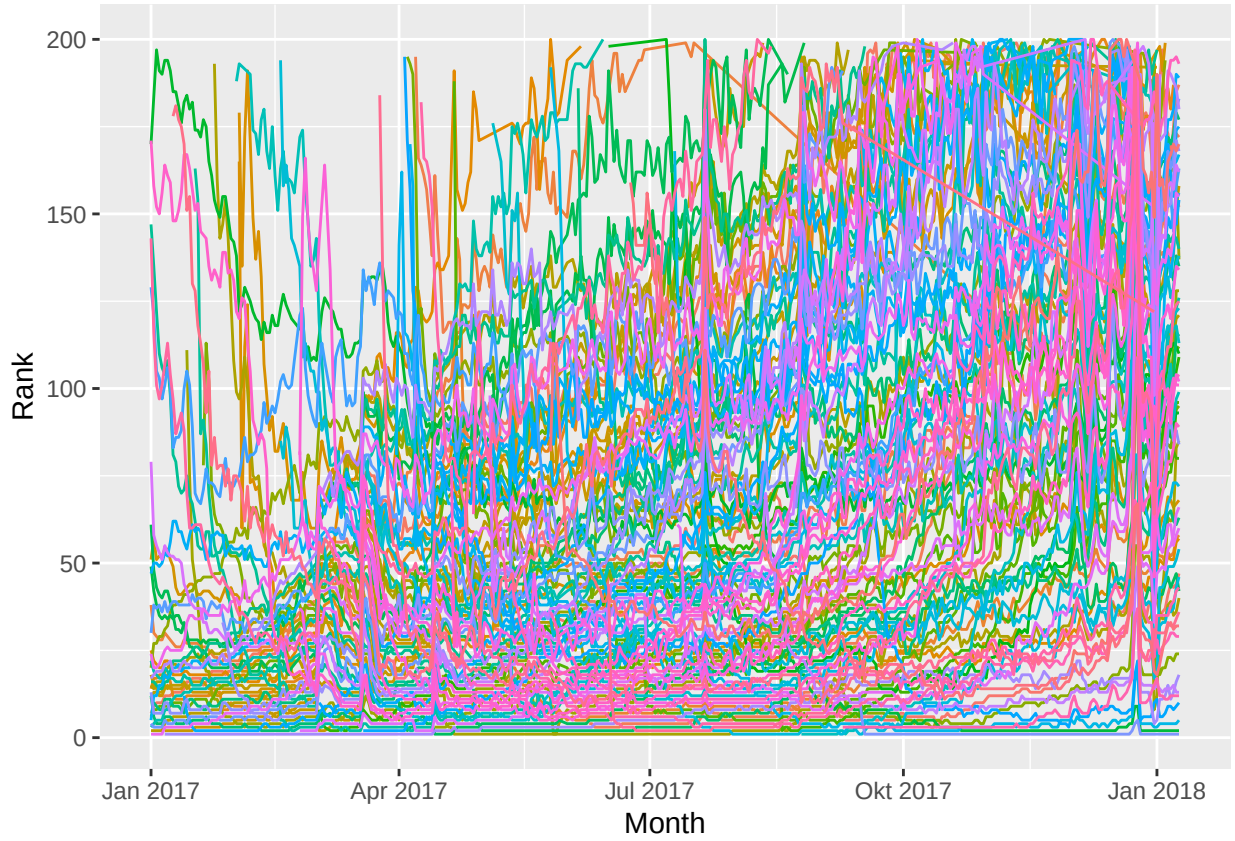
```
# but only 94 reach the top10 at some point
length(most_days_top10)
```

```
## [1] 94
```

```
# plot how the top 10 songs on the 1st of january 2017 behave over time
top_start17 <- ranks_glob[Date=='2017-01-01' & Position<11, Trackname]
ranks_glob_top <- ranks_glob[Trackname %in% top_start17]
ggplot(ranks_glob_top,aes(x=Date,y=Position,color=Trackname, group=Trackname)) +
  geom_line() +
  labs(x="Month", y="Rank") +
  theme(legend.position = "none")
```




```
# plot all songs that were in the top 100 in 2017 (and of which we have features)
ranks_glob_features <- ranks_glob[Trackname %in% features[,Trackname]]
ggplot(ranks_glob_features,aes(x=Date,y=Position,color=Trackname, group=Trackname)) +
  geom_line() +
  labs(x="Month", y="Rank") +
  theme(legend.position = "none")
```



Some analysis on which songs are in both datasets and creation of lists with those tracknames for later use

```
# get all tracknames that are in the top 100 of 2017 and which were in the top 10 at some point
# 53 tracks remain
tracknames_features <- features[,Trackname]
tracknames_top10 <- getSongsAtPos(ranks_glob, 10)
top10_features <- tracknames_top10[tracknames_features %in% tracknames_top10]


# all tracks in features that never were in top10
# 45 tracks remain [ => 2 tracks from features are not in the daily rankings data set]
tracknames_ranks <- ranks_glob[,.N, by=Trackname][,Trackname]
tracknames_not10 <- tracknames_ranks[!(tracknames_ranks %in% tracknames_top10)]
not10_features <- tracknames_features[tracknames_features %in% tracknames_not10]

features_top10 <- features[, top10 := Trackname %in% tracknames_top10]

# the names of the two songs that are not in the daily rankings data set
tracknames_features[!(tracknames_features %in% tracknames_ranks)]
```

```
## [1] "Don't Wanna Know (feat. Kendrick Lamar)"
## [2] "Cold (feat. Future)"
```

```
# exclude all songs that aren't in the daily rankings data set
features_top10 <- features_top10[!(Trackname %in% tracknames_features[!(tracknames_features %in% trackna
```

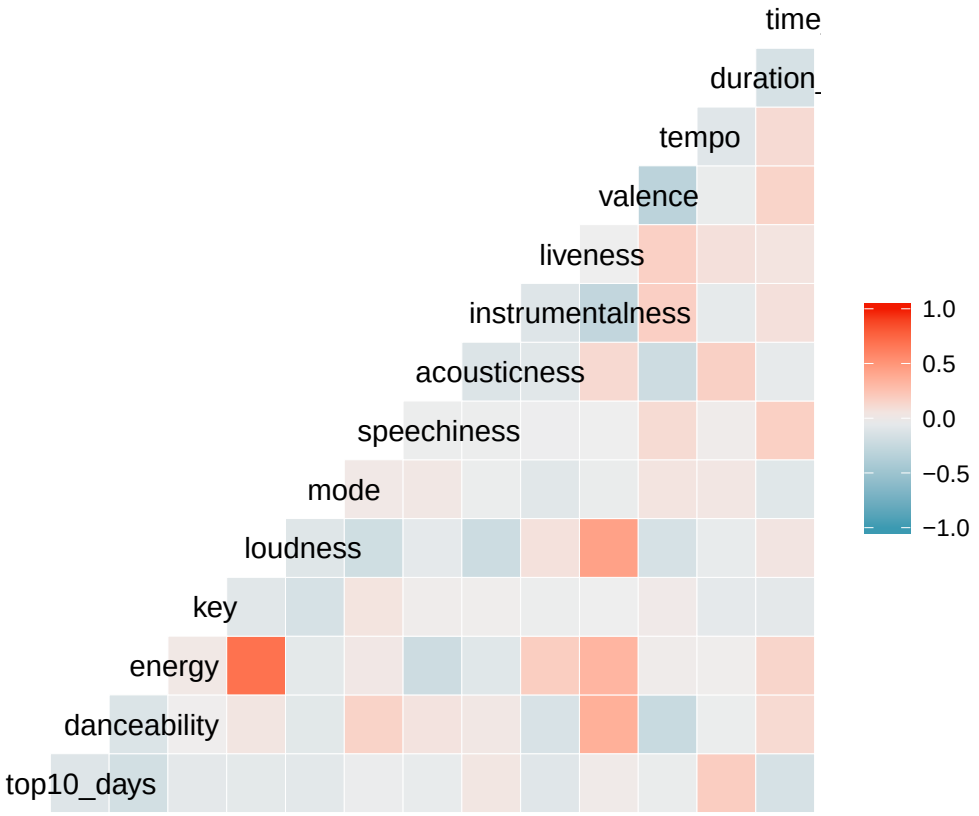
```r
# convert TRUE/FALSE to 1/0
features_top10[,top10:=as.numeric(top10)]

#GGally::ggcorr(features_top10[,!c("id","Trackname","artists")], geom = 'circle', params=c(corMethod="s

#songs_top1to50 <- getCumSongs(ranks_glob, 1:200)

#ggplot(songs_top1to50, aes(Position, N_Songs)) +
#  geom_point()
```

Linear Models: Calculate two metrics for how successfull a song was in 2017 and analyse which features influence that the most by using backward-selection

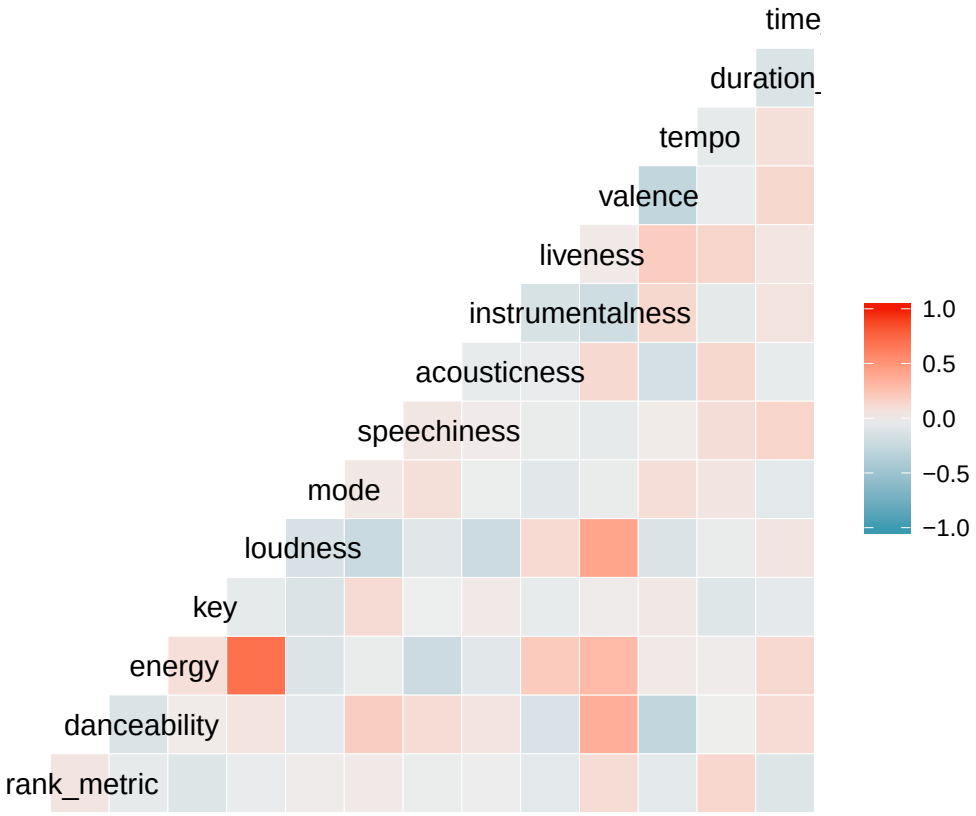
The two metrics are:

- Number of days in the top10
- Sum of (200-Rank) for all days the song is in the dataset

```r
### Create data.table with all songs that we have features for and add their days in the top10 ###
# filter for only songs that are in the top100 songs of 2017 and therefore we know their features
ranks_features <- ranks_glob[Trackname %in% features[,Trackname]]

# group by trackname and add column that counts how many days the song was in the top10
top10_days <- ranks_features[Position<11,.N, by=Trackname][,.(Trackname, top10_days = N)]
# add column that contains rank metric
rank_metric <- ranks_features[,sum(200-Position), by=Trackname][,.(Trackname, rank_metric = V1)]

# add the 45 tracks that are never in top10 with top10_days = 0
not10_days <- data.table(Trackname = not10_features, top10_days=c(0))
top10_days <- rbind(top10_days, not10_days)

not10_days_2 <- data.table(Trackname = not10_features, rank_metric=c(0))
rank_metric <- rbind(rank_metric, not10_days_2)

# add the features of the 98 songs and delete unnecessary columns
top10_days <- merge(top10_days, features, all.x=TRUE)[,!c("id", "top10","artists")]

rank_metric <- merge(rank_metric, features, all.x=TRUE)[,!c("id", "top10","artists")]

### Analyse how important the different features are ###
# plot correlation of all variables against each other
GGally::ggcorr(top10_days[,!c("Trackname")], hjust=0.1, method=c("all.obs", "spearman"))
```




```
GGally::ggcorr(rank_metric[,!c("Trackname")], hjust=0.1, method=c("all.obs", "spearman"))
```




```
# train the model - using backward-selection - for the top10_days metric
# don't use 'Trackname' and 'top10_days' as factor
factors <- names(top10_days)[-(1:2)]
top10_days_model <- backwardSelection(top10_days, factors, "top10_days", 0.05)
summary(top10_days_model)
```

```
##
## Call:
## lm(formula = formula, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -50.74 -29.95 -12.79  24.42 179.99
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  146.456     45.749   3.201  0.00186 **
## energy      -107.823     44.749  -2.410  0.01790 *
## loudness       7.584      3.456   2.195  0.03064 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.51 on 95 degrees of freedom
## Multiple R-squared:  0.06194,    Adjusted R-squared:  0.04219
## F-statistic: 3.137 on 2 and 95 DF,  p-value: 0.04796
```

```r
# train the model - using backward-selection - for the rank_metric
factors <- names(rank_metric)[-(1:2)]
rank_metric_model <- backwardSelection(rank_metric, factors, "rank_metric", 0.05)
summary(top10_days_model)
```

```
##
## Call:
## lm(formula = formula, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -50.74 -29.95 -12.79  24.42 179.99
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  146.456     45.749   3.201  0.00186 **
## energy      -107.823     44.749  -2.410  0.01790 *
## loudness       7.584      3.456   2.195  0.03064 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.51 on 95 degrees of freedom
## Multiple R-squared:  0.06194,    Adjusted R-squared:  0.04219
## F-statistic: 3.137 on 2 and 95 DF,  p-value: 0.04796
```

Prediction: Use the sum of the differences between the ranks of the last 7 days to predict the rank of the next day