# Case Study II: Top Spotify Tracks

*Ilaria Battiston, Timo Angerer, Yun Fei Xsu, Axel Vargas*

*05/02/2020*

## Introduction

Every year, Spotify publishes a report of the most popular songs. This is included in two datasets:

- Top 100 tracks of all time along with features;
- Top 100 daily for country.

The goal of analysis is first of all to explore the available information, highlighting relevant details and cleansing any mistaken values, to then perform an accurate study on how features relate to popularity and how much influence the local rankings have on the global one.

Machine learning algorithms are essential for data analytics, since they allow to calculate correlation, time series and clusters, which are source of important information for songs features.

## 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, magrittr,
               tidyverse, janitor, ggthemes, tools, DescTools)
```

## Data input

```
# set 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=""))
```

## Data preprocessing

A first approach of data preprocessing is performed on input, to make it clearer and easier to use. All analyses in the next sections are limited 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]

# parsing a song that appears twice
ranks_glob[, Trackname :=
             ifelse(grepl('Thomas Rhett', Artist), 'Unforgettable_2', Trackname)]
```

Some helper functions are added for later use.

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

# return 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;
# a critical p_value (crit_p) can be set 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

After obtaining a general idea of the available information, some exploratory analysis of the dataset is relevant to get a feel how it looks like and maybe discover interesting trends to further evaluate.

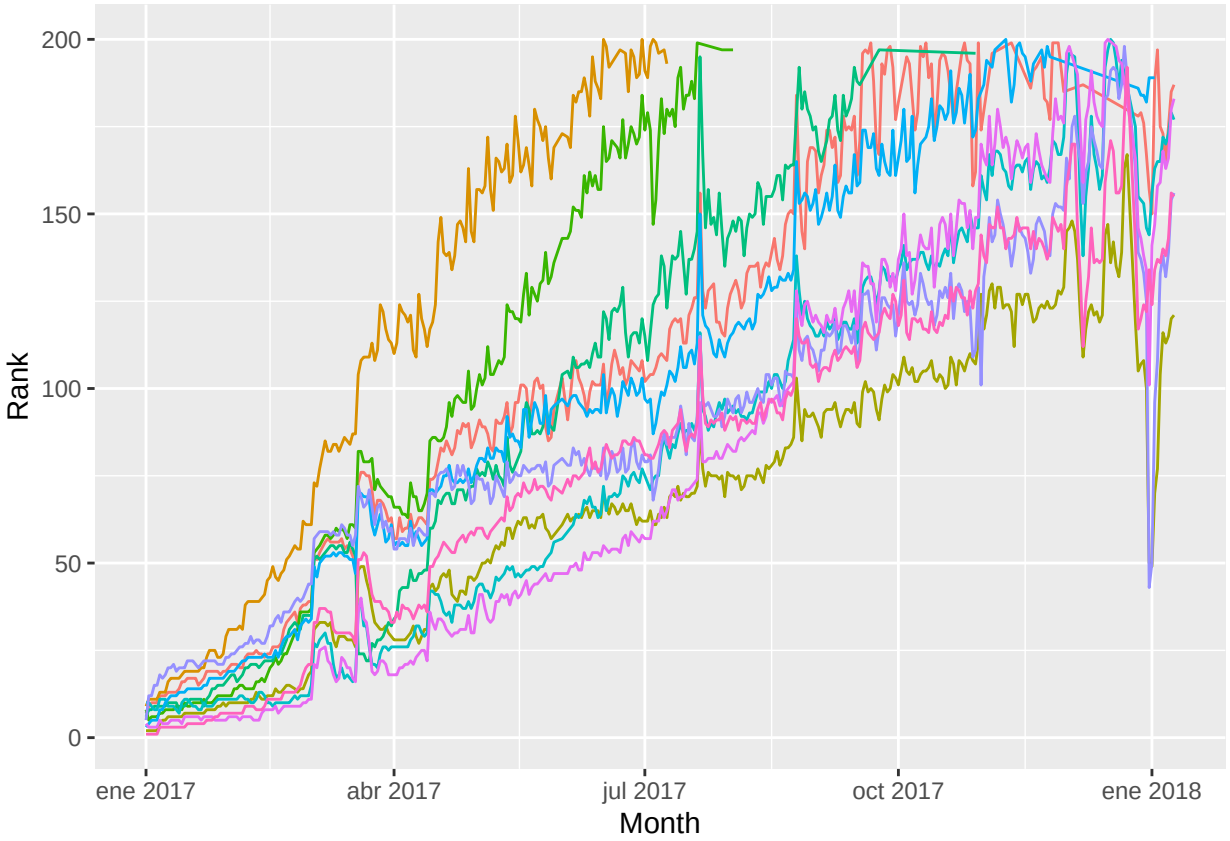
```
# get all songs that reached the top 10
most_days_top10 <- getSongsAtPos(ranks_glob, 10)
# there are a total of 1307 songs in the data set
ranks_glob[ , .N, by = Trackname][ , .N]
```

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

```
# but only 94 reach the top 10 at some point
length(most_days_top10)
```

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

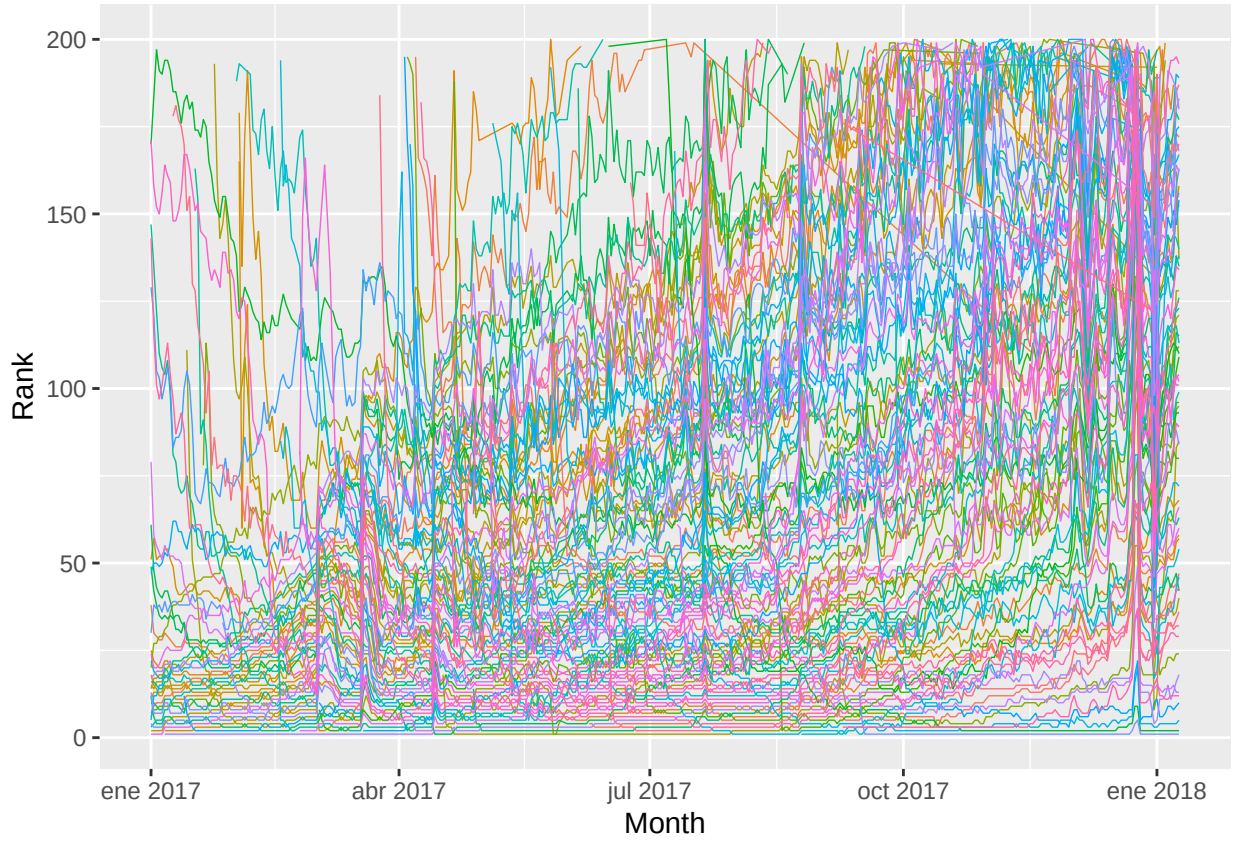
```
# plot how the top 10 songs on the 1st of January 2017 behave over time
top_start17 <- ranks_glob[Date=='2017-01-01' & Position < 11, Trackname]
ranks_glob_top <- ranks_glob[Trackname %in% top_start17]
ggplot(ranks_glob_top,aes(x=Date,y=Position,color=Trackname, group=Trackname)) +
  geom_line()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




Most songs follow a pattern of slowly earning popularity over time and then suddenly dropping after their peak, usually after some months.

Additional information about patterns might be obtained using the whole dataset.

```
# 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(size = 0.25) +
  labs(x = "Month", y = "Rank") +
  theme(legend.position = "none")
```




The pattern in this case is much more variable, leading to the assumption that only songs that get to the top follow the previous hypothesis.

## Cross analysis of the two datasets

This can be useful for error checking and to better understand the influence of days and position of each song in the final top 100, trying to understand how the rankings are related.

```
# only extracting data from 2017
daily_spotify_2017 <- daily_spotify[Date < '2018-01-01', ]

# how many times have those songs been streamed in total?
daily_spotify_2017_sum <- daily_spotify_2017[, sum(Streams),
                                             by = "Track Name"][order(-V1)]
head(daily_spotify_2017_sum)
```

```
##                          Track Name         V1
## 1:                     Shape of You 2954314942
## 2:                 Despacito - Remix 1808988398
## 3: Despacito (Featuring Daddy Yankee) 1449083311
## 4:          Something Just Like This 1366414897
## 5:                     Unforgettable 1341220672
## 6:                          HUMBLE. 1288359986
```

```
# there are some discrepancies between the two rankings
# now let's take the top 100 of all time
daily_100 <- daily_spotify_2017_sum[1:100, ]
```

```r
# how many days were they in the top 100?
daily_100_days <- daily_spotify_2017[daily_spotify_2017$`Track Name`
                                     %in% daily_100$`Track Name`]
daily_100_group <- daily_100_days[, .N, by = "Track Name"][order(-N)]
head(daily_100_group, 5)
```

```
##                                Track Name     N
## 1:                            Shape of You 18901
## 2:                               Believer 16393
## 3:      Despacito (Featuring Daddy Yankee) 15985
## 4: Rockabye (feat. Sean Paul & Anne-Marie) 15908
## 5:                Something Just Like This 15765
```

```r
# this is still different than the top 100 worldwide
# Spotify in 2017 was available in 60-61 countries, the other one has 53 (+ global)
unique(daily_spotify_2017$Region)
```

```
##  [1] "ec"     "fr"     "ar"     "fi"     "no"     "it"     "lt"
##  [8] "ph"     "tw"     "nz"     "ee"     "tr"     "us"     "sv"
## [15] "cr"     "de"     "cl"     "jp"     "br"     "hn"     "gt"
## [22] "ch"     "hu"     "ca"     "pe"     "be"     "my"     "dk"
## [29] "bo"     "pl"     "at"     "pt"     "se"     "mx"     "pa"
## [36] "uy"     "is"     "es"     "cz"     "ie"     "nl"     "sk"
## [43] "co"     "sg"     "id"     "do"     "lu"     "gb"     "global"
## [50] "py"     "au"     "lv"     "gr"     "hk"
```

```r
# those are the largest countries

# verifying 'global' data
daily_spotify_2017_sum_global <-
  daily_spotify_2017[Region == "global", sum(Streams), by = "Track Name"][order(-V1)]
# still different

# how many days have the most popular songs stayed at the top?
top_1_days <- daily_spotify_2017[Position == 1 & Region == "global", .N,
                                 by = "Track Name"][order(-N)]
head(top_1_days)
```

```
##                  Track Name   N
## 1:                 rockstar 103
## 2:             Shape of You  96
## 3:        Despacito - Remix  96
## 4:                  Mi Gente  29
## 5: Look What You Made Me Do  15
## 6:                   HUMBLE.   7
```

```r
top_10_days <- daily_spotify_2017[Position <= 10 & Region == "global",
                                  .N, by = "Track Name"][order(-N)]
head(top_10_days)
```

```
##           Track Name   N
## 1:      Shape of You 231
## 2:         New Rules 138
## 3: Despacito - Remix 135
## 4:            Havana 120
## 5:     Unforgettable 114
```

```
## 6:       I'm the One 108
```

There are some discrepancies between the top 100 globally and the calculated top 100 from the daily Spotify data. This might be due to other external factors not present in the datasets influencing the final rankings, or other missing countries with different preferences.

Further analysis and cleansing is performed using features.

```
# 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 top 10
# 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]

# 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% tracknames_ranks)])]

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

## Linear modelling

Linear modelling aims to calculate some metrics to represent how successful a song was in 2017.

These indicators are then used to analyse which features influence that the most by using backward-selection.

The two chosen metrics are:

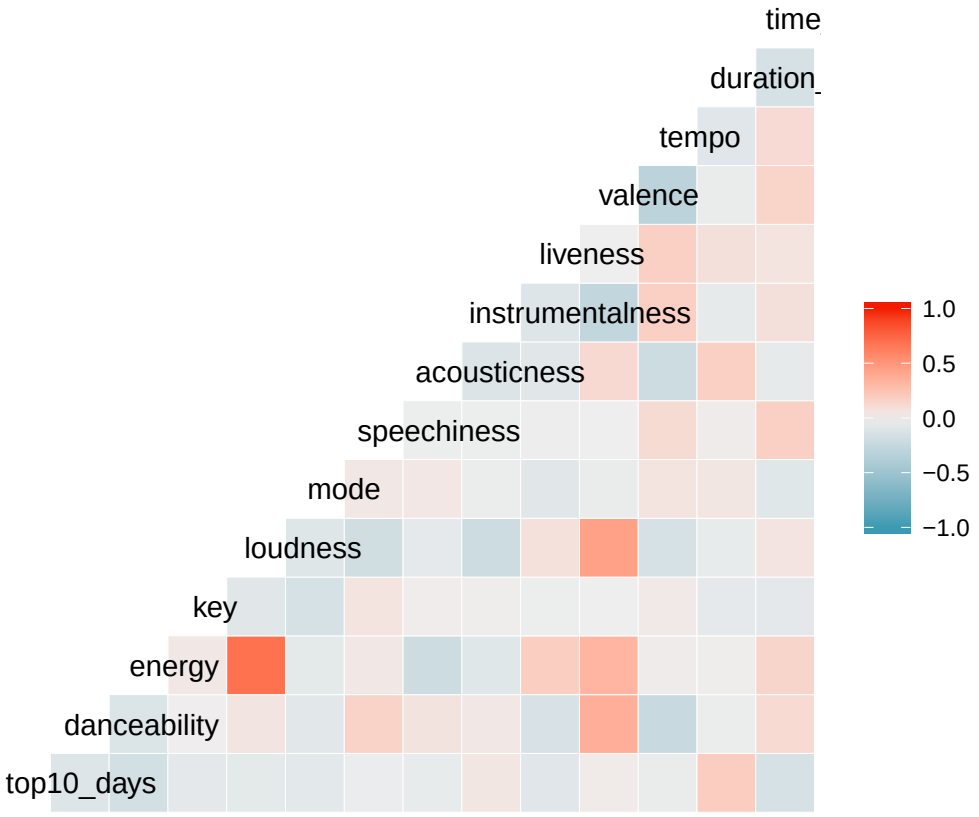
- Number of days in the top 10;
- Sum of (200-Rank) for all days the song is in the dataset.

```
# create data.table with all songs there are features of
# and add their days in the top 10
# filter for only songs that are in the top 100 songs of 2017
ranks_features <- ranks_glob[Trackname %in% features[,Trackname]]
# output is a complete dataset of features

# group by trackname and add column that counts how many days the song was in the top 10
top10_days <- ranks_features[Position < 11, .N,
                             by = Trackname][ , .(Trackname, top10_days = N)]
```

```r
# 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 top 10 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")]
```

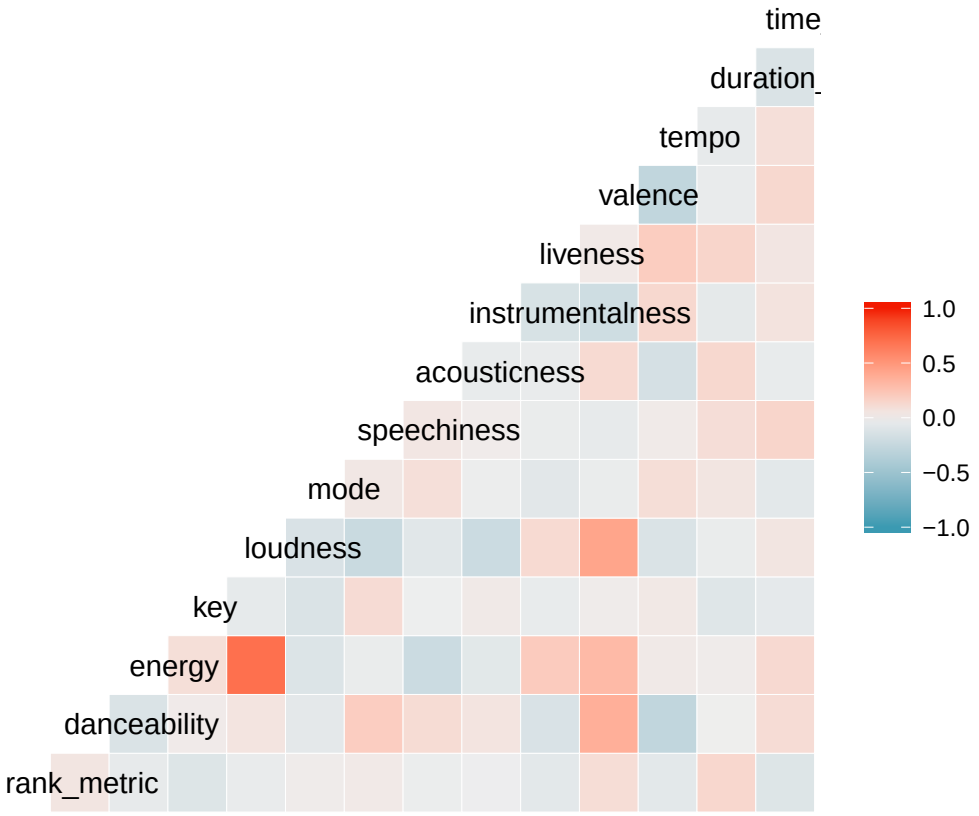
After obtaining a parsed dataset, the importance of different features is highlighted through plotting distances.

```r
# plot correlation of all variables against each other
GGally::ggcorr(top10_days[,!c("Trackname")], hjust=0.1, method=c("all.obs", "spearman"))
```




Energy is the most relevant feature according to the most popular songs from days in rankings. A further check is performed using the rankings.

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




Results are similar, which proves the strenght of features. After calculating correlation, the model can be trained.

```
# train the model - using backward-selection - for the top10_days metric
# not using '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
```

```
# 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(rank_metric_model)
```

```
##
## Call:
## lm(formula = formula, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -27431 -22379   3318  13859  42912
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  27430.9     2845.2   9.641   <2e-16 ***
## key           -561.4      425.1  -1.320    0.189
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18510 on 141 degrees of freedom
## Multiple R-squared:  0.01221,    Adjusted R-squared:  0.005209
## F-statistic: 1.744 on 1 and 141 DF,  p-value: 0.1888
```

It can be seen that for the `top10_days` metric only the features energy and loudness remain with a p-value below the threshold of 0.05 after using backward-selection. However, both p-values are pretty high, with 0.17 for energy and 0.03 for loudness. The model overall doesn't predict the `top10_days` metric very well, with a p-value of 0.048 and R-squared being at 0.62.

For the self-defined rank_metric the model is even worse, with no features with a p-value lower than 0.05 remaining. The p-value is way too high with 0.19 and the R-squared too low, which implies that this model didn't find any correlation between the `rank_metric` and the features. This might be due to the metric being chosen poorly and not reflecting the popularity of a song very well,

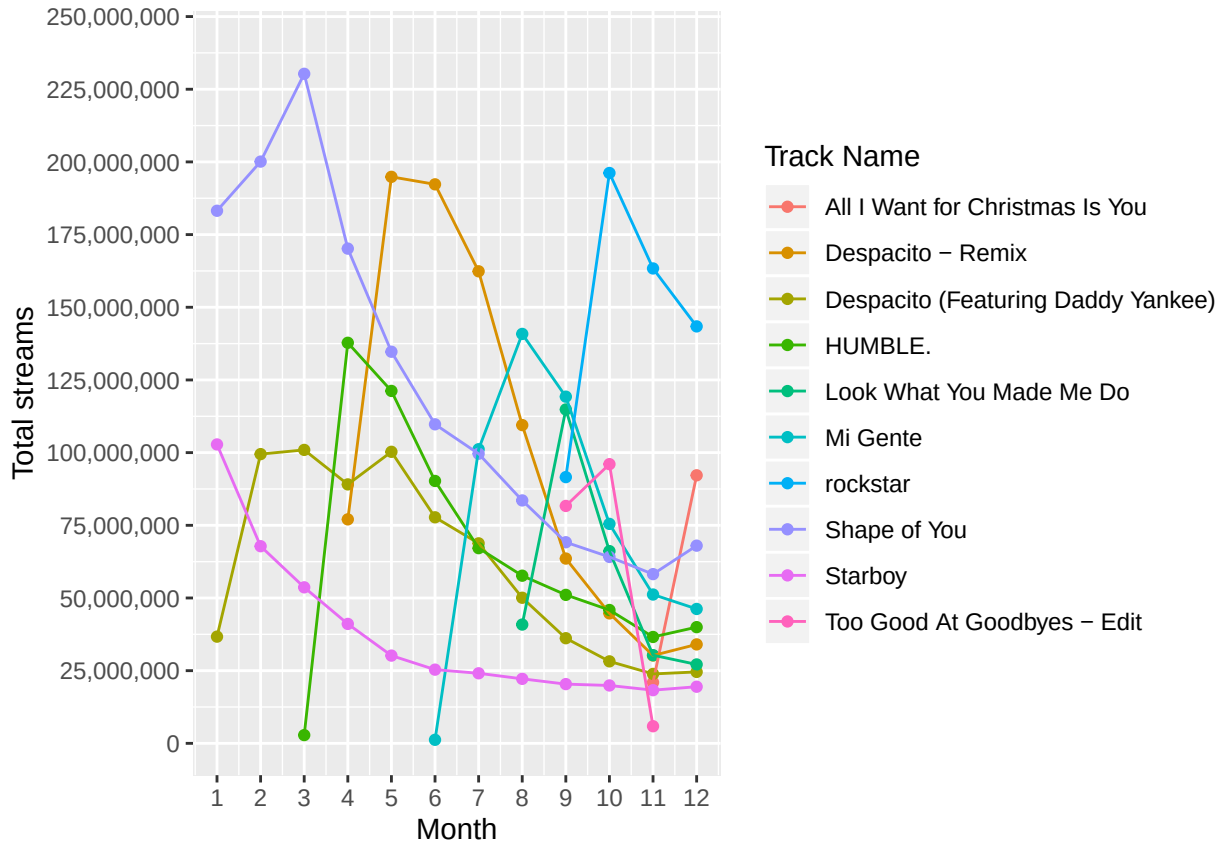
Further tuning might improve the first model for the `top10_days` metric but the focus of this part was on implementing and using the backward-selection algorithm.

## Analysing Christmas songs features

Since top songs have an interesting pattern regarding their behaviour in rankings, this can be an informative feature to analyse. After zooming on tracks by month, it can be seen that Christmas songs follow an unique trend, therefore analysis can be performed to get to know differences.

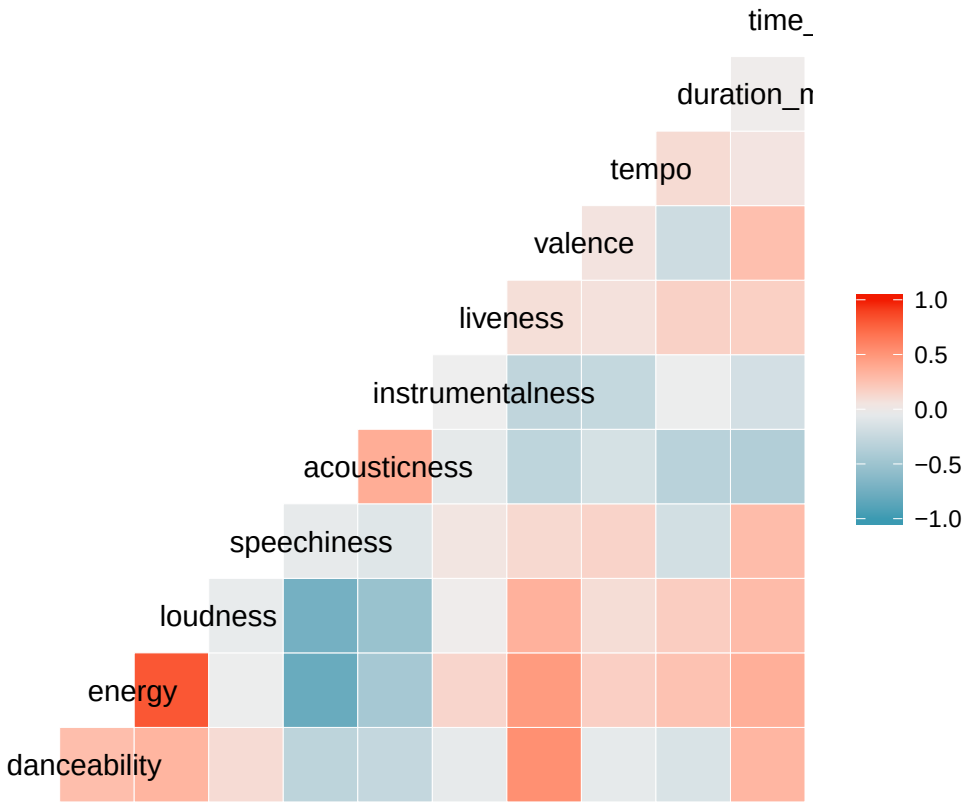
```
# plotting number of streams of top 10 by month
streams_day <- daily_spotify_2017[Region == "global" & `Track Name`
                                  %in% top_1_days$`Track Name`] %>%
  group_by(`Track Name`, month(Date)) %>%
  summarise(Streams = sum(Streams))

ggplot(streams_day, aes(x=`month(Date)`, y=Streams, color=`Track Name`)) +
  geom_point() + geom_line() +
  scale_y_continuous(breaks = seq(0, 250000000, by=25000000), labels=comma,
```

```
                    limits=c(1000000, 240000000)) +
  scale_x_continuous(breaks=c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)) +
  labs(x="Month", y="Total streams")
```




This plot clearly highlights that the only Christmas song has a completely different pattern, only gaining popularity in December. Further examination on Christmas songs are performed, also using an external dataset.

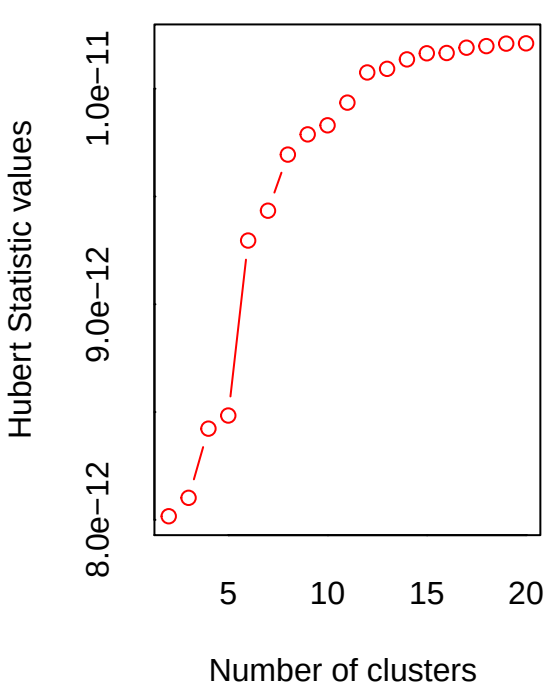
```
# extracting the top 50 worldwide in December
top_songs_december = daily_spotify_2017[month(Date) == 12 & Region == "global",
                                        sum(Streams), by = "Track Name"]

# recognizing Christmas songs
# https://data.world/promptcloud/spotify-musical-features-of-160-holiday-songs
# assuming no different songs with same title
christmas_dataset <- fread(paste(csv_path, "holiday_songs_spotify.csv", sep=""))

christmas_songs_spotify <- christmas_dataset[trimws(toupper(track_name))
                                             %in% trimws(toupper(
                                               top_songs_december$`Track Name`)), ]

christmas_songs_spotify_subset <- christmas_songs_spotify[, 3:15][, -3][, -4]

# are Christmas songs related?
GGally::ggcorr(christmas_songs_spotify_subset, hjust=0.1, method=c("all.obs", "spearman"))
```

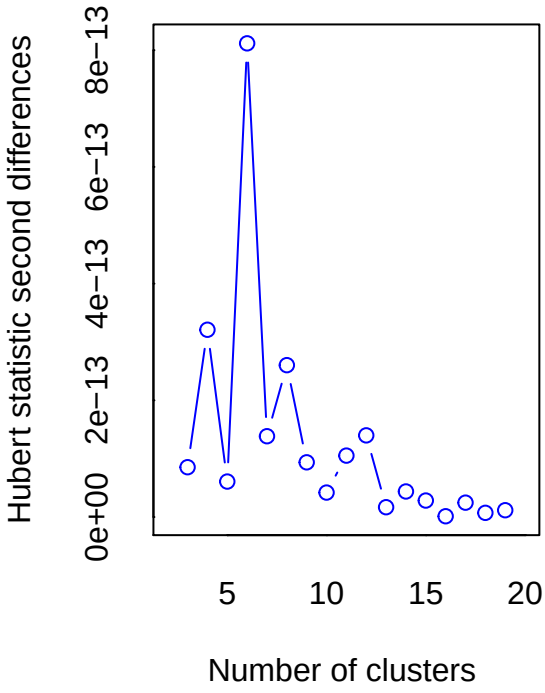



Similarly to the general result previously obtained, energy is the most correlated feature of Christmas songs, yet also danceability, liveness and speechiness. It can be assumed that those features are the most common in Christmas songs.

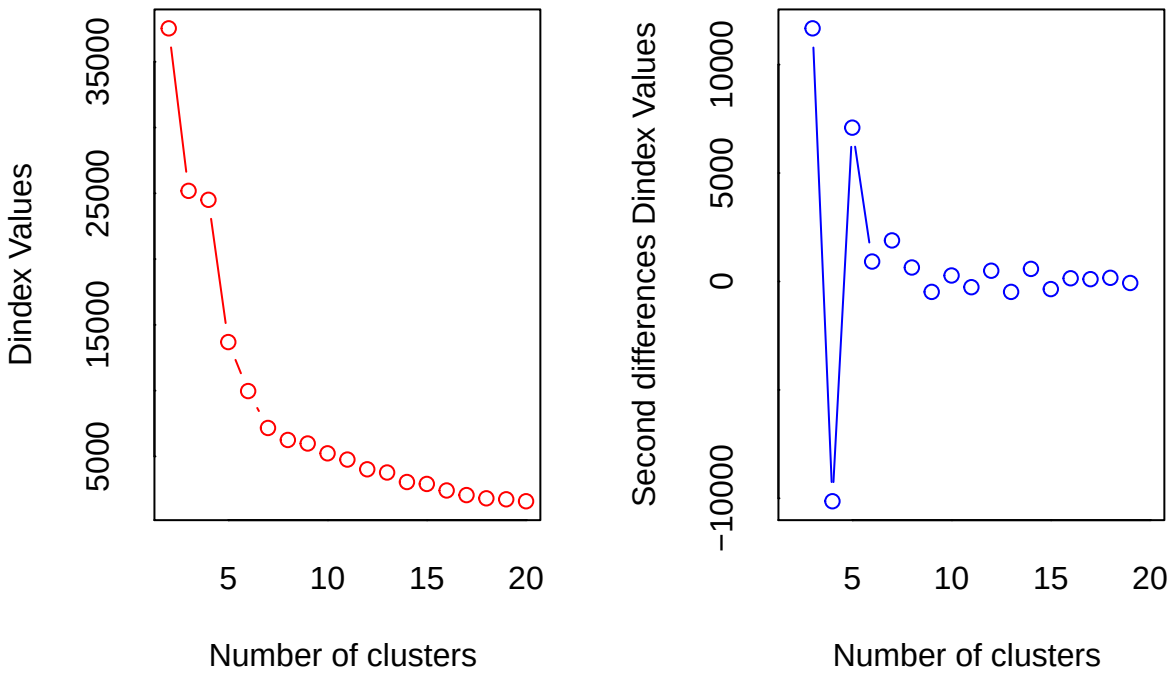
Cluster might give additional information on how they relate to each other.

```
ncluster <- NbClust(christmas_songs_spotify_subset, min.nc=2, max.nc=20, method="centroid")
```




```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```




```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 2 proposed 2 as the best number of clusters
## * 8 proposed 3 as the best number of clusters
## * 1 proposed 4 as the best number of clusters
## * 4 proposed 5 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 1 proposed 17 as the best number of clusters
## * 2 proposed 18 as the best number of clusters
## * 1 proposed 19 as the best number of clusters
## * 3 proposed 20 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  3
##
##
## *******************************************************************
```

```r
# 3 clusters
set.seed(1234)
christmas_clusters <- kmeans(christmas_songs_spotify_subset, centers=3, iter.max=100, nstart=25)
```

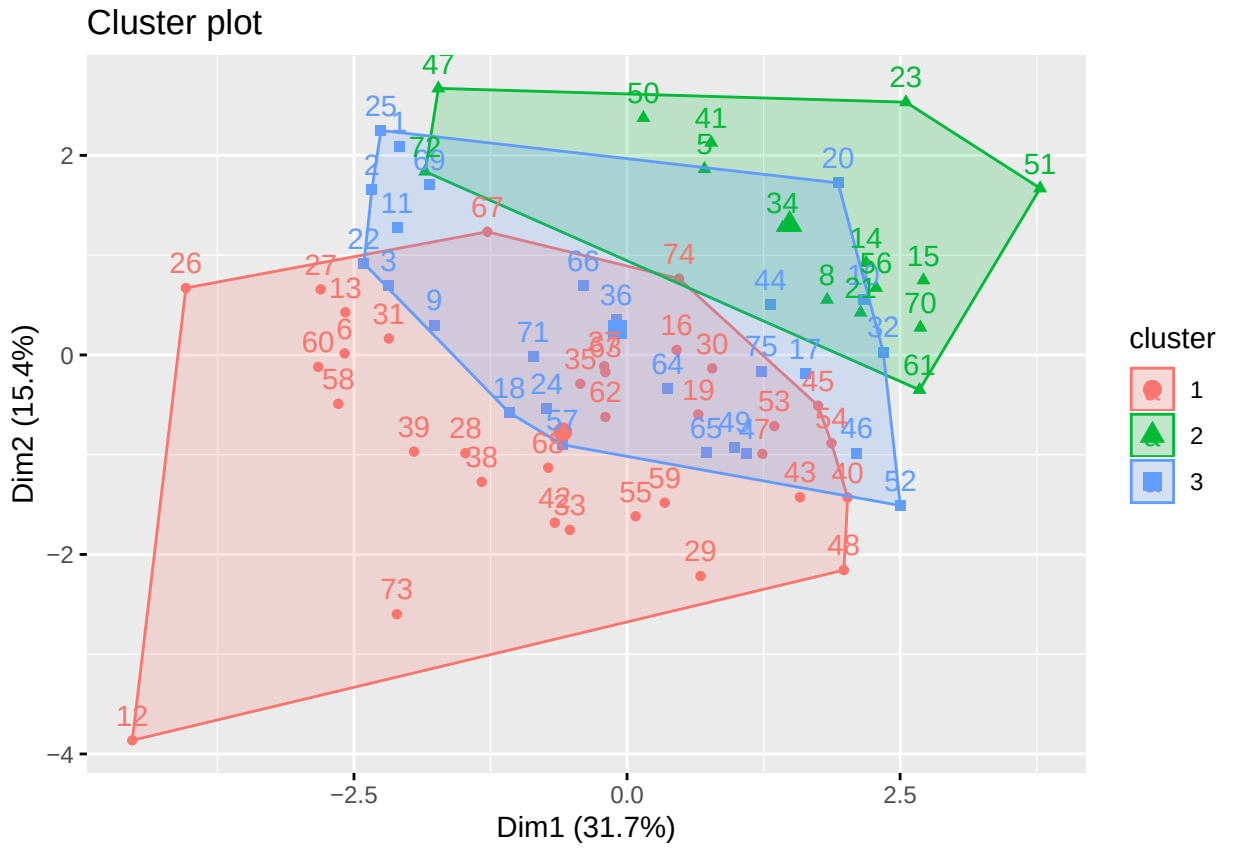
```
christmas_clusters$centers
```

```
##   danceability    energy  loudness speechiness acousticness
## 1    0.5356176 0.4360206 -10.54874  0.04700000    0.6623588
## 2    0.5086667 0.6863333  -6.89280  0.03639333    0.2836873
## 3    0.5358077 0.4535769 -10.09008  0.05198077    0.5756385
##   instrumentalness  liveness   valence    tempo duration_ms time_signature
## 1     0.0042489459 0.1578882 0.5812382 114.2085    138885.0       3.764706
## 2     0.0000106040 0.2643533 0.5824000 118.6799    256401.8       3.933333
## 3     0.0001516635 0.1990615 0.5308077 123.5463    197047.5       3.769231
```

```
christmas_clusters$size
```

```
## [1] 34 15 26
```

```
fviz_cluster(christmas_clusters, data = christmas_songs_spotify_subset)
```




There are 3 clusters of size 34, 15 and 26, with the following dominant features:

- One has much higher energy, high loudness and low speechiness;
- One has particularly high instrumentalness and acousticness;
- One has more balanced values, with highest tempo and speechiness.

## Trend Exploration on top songs

The information provided may help to identify trends, the next code examine the behavior of the top 1 worldwide songs:

```r
# preprocessing function for cleaning names
clean_rows_name <- function(char){
  partial_clean_names <- function(.data, unique = FALSE) {
    n <- if (is.data.frame(.data)) colnames(.data) else .data

    n <- gsub("%+", "_pct_", n)
    n <- gsub("\\$+", "_dollars_", n)
    n <- gsub("\\++", "_plus_", n)
    n <- gsub("-+", "_Minus_", n)
    n <- gsub("\\*+", "_star_", n)
    n <- gsub("#+", "_cnt_", n)
    n <- gsub("&+", "_and_", n)
    n <- gsub("@+", "_at_", n)

    n <- gsub("[^a-zA-Z0-9_]+", "_", n)
    n <- gsub("([A-Z][a-z])", "_\\1", n)
    n <- tolower(trimws(n))

    n <- gsub("(^_+|_+$)", "", n)

    n <- gsub("_+", "_", n)

    if (unique) n <- make.unique(n, sep = "_")

    if (is.data.frame(.data)) {
      colnames(.data) <- n
      .data
    } else {
      n
    }
  }

  char %>%
    partial_clean_names() %>%
    str_replace_all('_', ' ') %>%
    toTitleCase() %>%
    str_replace_all(' i ', ' I ') %>%
    str_replace_all(' Minus ', '- ') %>%
    StrCap() %>%
    return()
}
```

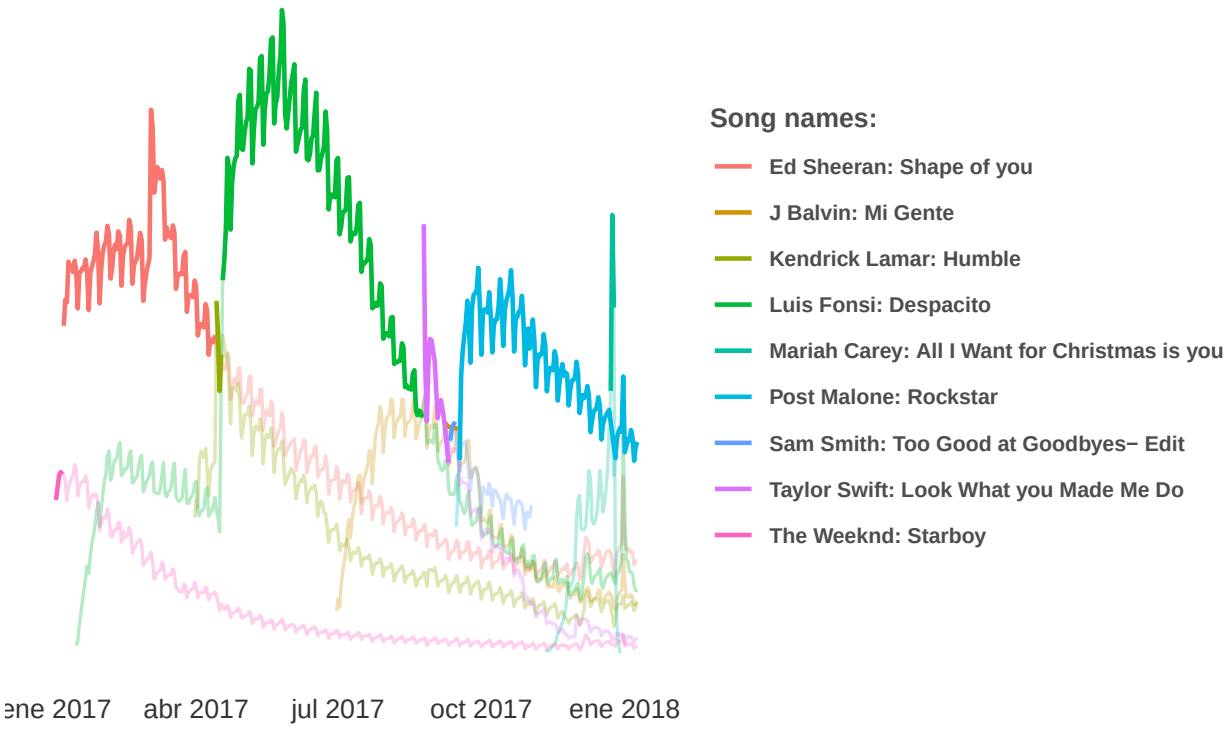
During the cleaning processed we standarized the names of the songs employing our function and clean_names() contained in the packages 'janitor'; we noticed that 'Despacito' had two 'ids', for the purpose of this studies we summed the songs streams and created a new register for this song. Likewise, one hypothesis was that studying the top songs worldwide can be a good proxy for knowing the structural behavior of the series.

```r
# removing rows with null data and processing names
spoti_clean <-
  daily_spotify %>%
  filter(Region == 'global') %>%
  clean_names() %>%
  mutate(
    date = as.Date(date, '%Y-%m-%d'),
```

```
    year = year(date),
    month = month(date),
    track_name = clean_rows_name(track_name),
    artist = clean_rows_name(artist),
    track_name = ifelse(str_detect(track_name, 'Despacito') == T, 'Despacito', track_name),  # Summaris
    general_name = paste0(artist, ': ', track_name)
    ) %>%
  filter(track_name != '' & artist != '')

# We select the top 1 position songs
tops_prev <-
  spoti_clean %>%
  filter(position == 1) %>%
  select(general_name) %>%
  unique()

tops <-
  tops_prev %>%
  bind_cols(
    colors = rainbow(nrow(tops_prev))
  )

# Sum all the objects with the same features
general_data <-
  spoti_clean %>%
  group_by(general_name, artist, track_name,  date, region, year, month) %>%
  summarise(streams = sum(streams)) %>%
  ungroup()



# Plot
top_tracks <-
  general_data %>%
  filter(general_name %in% (tops %>% select(general_name) %>% pull())) %>%
  inner_join(tops)

top_positions <-
  general_data %>%
  group_by(date) %>%
  mutate(
    max = ifelse(max(streams) == streams, 1, 0)) %>%
  ungroup() %>%
  filter(max ==1) %>%
  inner_join(tops) %>%
  arrange(date) %>%
  mutate(
    general_name = factor(general_name, levels = unique(top_tracks$general_name))
  )
```

The plot beneath shows breaks in the time series trends, suggesting that a linear approach won't be the best one in certain cases due to a different stream increasing patterns.

```r
top_tracks %>%
  arrange(date) %>%
  mutate(
    general_name = factor(general_name, levels = unique(top_tracks$general_name))
  ) %>%
  ggplot(aes(x = date, y = streams, color = general_name, group = general_name)) +
  geom_line(show.legend = FALSE, size = 0.6, alpha = 0.3) +
  geom_line(data = top_positions,
            aes(
              x = date,
              y = streams
              ),
            size = 0.8
            ) +
  labs(title = "Top songs in Spotify",
       subtitle = 'jan.2017 - jan.2018') +
  theme(
    axis.title = element_blank(),

    axis.text.x = element_text(size = 10, color = "gray20", angle = 0),
    axis.text.y = element_blank(),

    axis.ticks.x = element_blank(),
    axis.ticks.y = element_blank(),

    panel.background = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),

    legend.title = element_text(size=10, color = "gray30", face="bold"),
    legend.text = element_text(size=8, color = "gray30", face="bold"),
    legend.background = element_blank(),
    legend.key = element_blank()
  ) +
  scale_color_discrete(name = "Song names:")
```



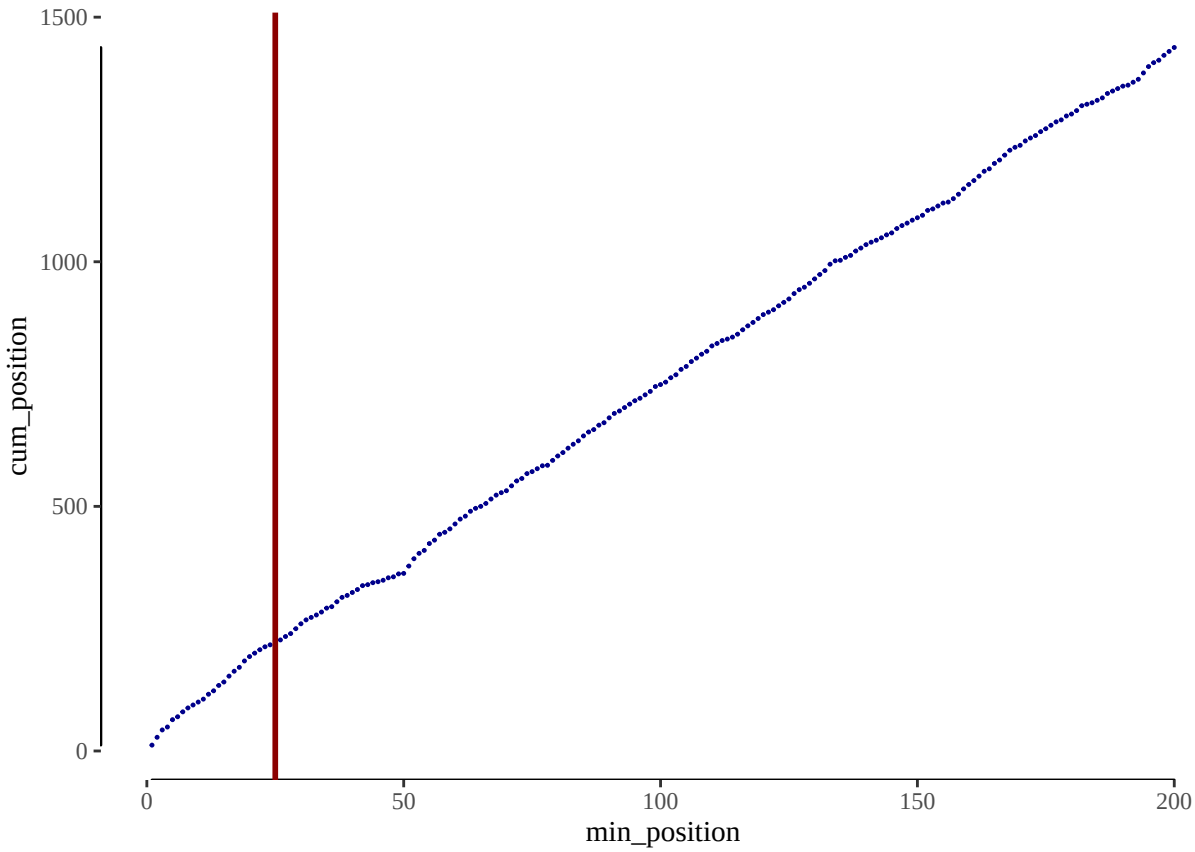

## Logistic Model

In this section we propose another approach for estimating the probability of a song to become a 'top' one in all the regions. So far we have studied linear/trends behavior, however this path analyse the data via a logistic regression model employing the feautres as regressors. For this we cleaned similarly our dataset as before considering a sample of 70% due to time processing issues.

```
# We create a seed for replation purpose
set.seed(123)
# Data preparation ---------------------------------------------
spoti_clean <-
  daily_spotify %>%
  sample_n(0.7*nrow(daily_spotify), replace = FALSE) %>%   # Sample of 70%
  clean_names() %>%
  mutate(
    date = as.Date(date, '%Y-%m-%d'),
    year = year(date),
    month = month(date),
    track_name = clean_rows_name(track_name),
    artist = clean_rows_name(artist),
    track_name = ifelse(str_detect(track_name, 'Despacito') == T, 'Despacito', track_name), # Problems
    general_name = paste0(artist, ': ', track_name),
    id = substr(url, 32, nchar(url)-1)                    # Id for joining features vs position
  ) %>%
  filter(track_name != '' & artist != '')
```

```r
# Cleaning features
features_clean <-
  spotify_data %>%
  clean_names() %>%
  mutate(
    name = clean_rows_name(name),
    artists = clean_rows_name(artists),
    name = ifelse(str_detect(name, 'Despacito') == T, 'Despacito', name),
    general_name = paste0(artists, ': ', name)
  ) %>%
  filter(name != '' & artists != '')                    # Rows without a name/artist
```

```r
# Songs best position achieved code
tops_achieved <-
  spoti_clean %>%
  filter(region == 'global') %>%
  select(id, position) %>%
  unique() %>%
  group_by(id) %>%
  summarise(min_position = min(position)) %>%
  ungroup() %>%
  arrange(min_position)

# Counting per position
data_plot <-
  tops_achieved %>%
  group_by(min_position) %>%
  summarise('Number of Tracks' = n()) %>%
  ungroup() %>%
  mutate(cum_position = cumsum(`Number of Tracks`))

# Plots the Number of songs rank by its position
data_plot %>%
  ggplot(aes(x = min_position, y = cum_position)) +
  geom_point(color = 'darkblue', size = 0.2) +
  geom_rangeframe() +
  theme_tufte() +
  geom_vline(xintercept = 25, color = 'darkred', size = 1)
```




```
# Logistic Model ------------------------------------------------------------
# Data preparation
position_test <-
  tops_achieved %>%
  inner_join(features_clean %>%
               select(danceability, energy, loudness, mode, speechiness,
                      acousticness, instrumentalness, liveness, valence,
                      key, id)) %>%
  mutate(
    state = ifelse(min_position < 25, 1, 0),                # Threshold for analysis, top 25 songs
    state = factor(state),
    mode = factor(mode)
    ) %>%
  as.data.table()

# Training set
model_train_test <-
  position_test %>%
  sample_n(0.6*nrow(position_test), replace = FALSE) %>%     # 60% used for training our data
  unique()

input_model <-
  model_train_test %>%
  select(-id, -min_position)
```

```r
# Logistic fitting
logit_complete_test <-
  glm(state ~ . -1,
      data = input_model,
      family = "binomial")
summary(logit_complete_test)
```

```
##
## Call:
## glm(formula = state ~ . - 1, family = "binomial", data = input_model)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.89174   0.00002   0.00008   0.61801   1.54551
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## danceability       -5.7200     4.4518  -1.285    0.199
## energy              1.7574     4.6515   0.378    0.706
## loudness            0.3088     0.4202   0.735    0.462
## mode0              27.4098  1948.8925   0.014    0.989
## mode1               7.0517     6.7436   1.046    0.296
## speechiness         0.4734     4.3854   0.108    0.914
## acousticness        2.7217     3.5244   0.772    0.440
## instrumentalness   49.8366    42.6809   1.168    0.243
## liveness            0.5403     4.4441   0.122    0.903
## valence            -2.7494     2.0347  -1.351    0.177
## key                -0.1994     0.1480  -1.348    0.178
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 80.405  on 58  degrees of freedom
## Residual deviance: 38.860  on 47  degrees of freedom
## AIC: 60.86
##
## Number of Fisher Scoring iterations: 18
```

```r
# Test Set
model_test_test <-
  position_test %>%
  filter(!(id %in% model_train_test$id))                    # 'IDs' didn't use during the training stage

# Prediction employing the test set
pred_test <- predict(logit_complete_test, model_test_test, type="response")


# Reviewing results obtained
output_test <-
  cbind(model_test_test, pred_test) %>%
  mutate(
    yhat_test = ifelse(pred_test > 0.5, 1, 0),
    acc = ifelse(state == yhat_test, 1, 0)
  ) %>%
  as.data.table()
```

```
data.table(
  accuracy = output_test$acc %>% sum / nrow(output_test)
  ) %>%
  print()
```

```
##    accuracy
## 1:      0.8
```

The results obtained exhibit the possibility of tuning this model for increasing the accuracy. So far we have only studied songs during 2017 in which the more important p-values for classification conrresponds to 'danceability' and 'speechiness'.

## Spread of songs with different language through countries

This part of the analysis concerns the impact of each language on songs rankings, according to countries: particularly, how does a new release song spread over the world.

Based on previous analysis, it is chosen song which gets into the ranking a few weeks after release. To compare the popularity between countries, it is used the weekly average position to represent the popularity of the song in a single country.

```
# copy datasets and rename columns
daily_data_2017 = daily_spotify_2017 %>% rename(track_name = 'Track Name',
                                                position = Position,
                                                artist = Artist,
                                                streams = Streams,
                                                url = URL,
                                                date = Date,
                                                region = Region)

daily_data_2017$date <- as.Date(daily_data_2017$date)
week_of_year <- week(daily_data_2017$date)
daily_data_2017 <- cbind(daily_data_2017, week_of_year)
country_code <- c("ar"= "Argentina",    "at"="Austria", "au"="Australia",   "be"="Belgium", "bo"="Bolivi

# select the datasets of Swalla
swalla_flow = daily_data_2017 %>%
                filter(artist == "Jason Derulo",
                  track_name == "Swalla (feat. Nicki Minaj & Ty Dolla $ign)",
                  region != "global",
                  position <= 50) %>%
                select(position, streams, date, week_of_year,  region) %>%
                group_by(week_of_year, region) %>%
                summarise(popularity = mean(position))

# sort region by popularity
region_order <- unique(swalla_flow$region)
region_order <- data.frame(region = c(unique(region_order)))
for (i in 53:8) {
  k <- swalla_flow %>% filter(week_of_year == i)
  k <- left_join(region_order, k, by = "region")
  k <- k %>% arrange(popularity)
  region_order$region <- k$region
}
```

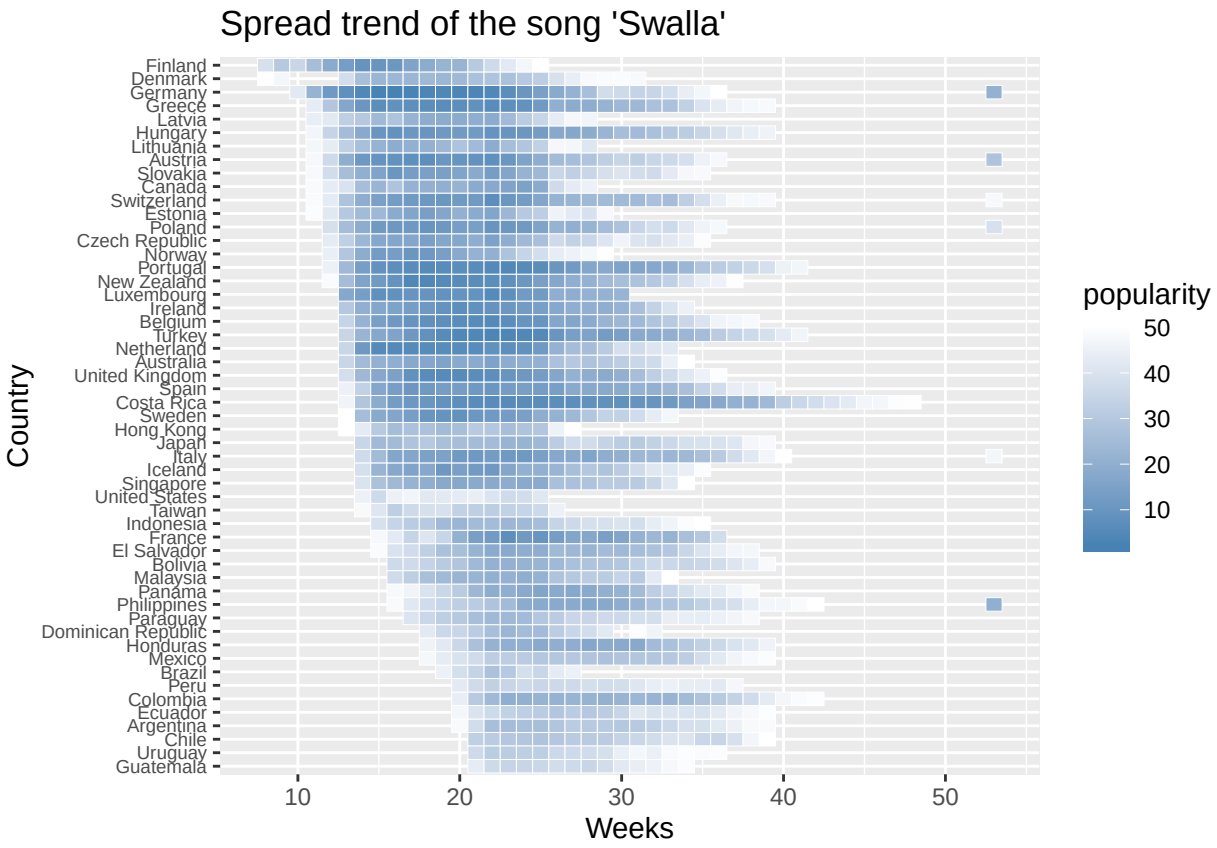
```
region_order$region
```

```
##  [1] "fi" "dk" "de" "gr" "lv" "hu" "lt" "at" "sk" "ca" "ch" "ee" "pl" "cz"
## [15] "no" "pt" "nz" "lu" "ie" "be" "tr" "nl" "au" "gb" "es" "cr" "se" "hk"
## [29] "jp" "it" "is" "sg" "us" "tw" "id" "fr" "sv" "bo" "my" "pa" "ph" "py"
## [43] "do" "hn" "mx" "br" "pe" "co" "ec" "ar" "cl" "uy" "gt"
```

```
region_order$region = region_order$region[53:1]
```

```
swalla_flow$region<-country_code[swalla_flow$region]
swalla_flow$region <- factor(swalla_flow$region, levels=unique(country_code[region_order$region]))
```

```
ggplot(swalla_flow, aes(week_of_year, region)) +
    labs(x = "Weeks", y="Country", title = "Spread trend of the song 'Swalla'") +
    theme(axis.text.y = element_text(size=7)) +
    geom_tile(aes(fill = popularity), colour = "white")+
    scale_fill_gradient(low = "steelblue", high = "white")
```




As can be seen from the graph, the song first becomes popular in Europe, starting from Northern Europe and spread all over Europe. It then becomes popular in countries of Oceania such as New Zealand and Australia. Six weeks after release, the song gets into the top 50 in Asia and North America, then finally reaches the top 50 in South America after nine weeks. Note that it didn't hit a very top ranking in the US.

```r
# select the datasets of Despacito
despacito_flow = daily_data_2017 %>%
                    filter(artist == "Luis Fonsi",
                      track_name == "Despacito - Remix",
                      region != "global",
                      position <= 50) %>%
                    select(position, streams, date, week_of_year,  region) %>%
                    group_by(week_of_year, region) %>%
                    summarise(popularity = mean(position))


region_order <- unique(despacito_flow$region)
region_order <- data.frame(region = c(region_order))
for (i in 53:16) {
  k <- despacito_flow %>% filter(week_of_year == i)
  k <- left_join(region_order, k, by = "region")
  k <- k %>% arrange(popularity)
  region_order$region <- k$region
}
region_order$region
```

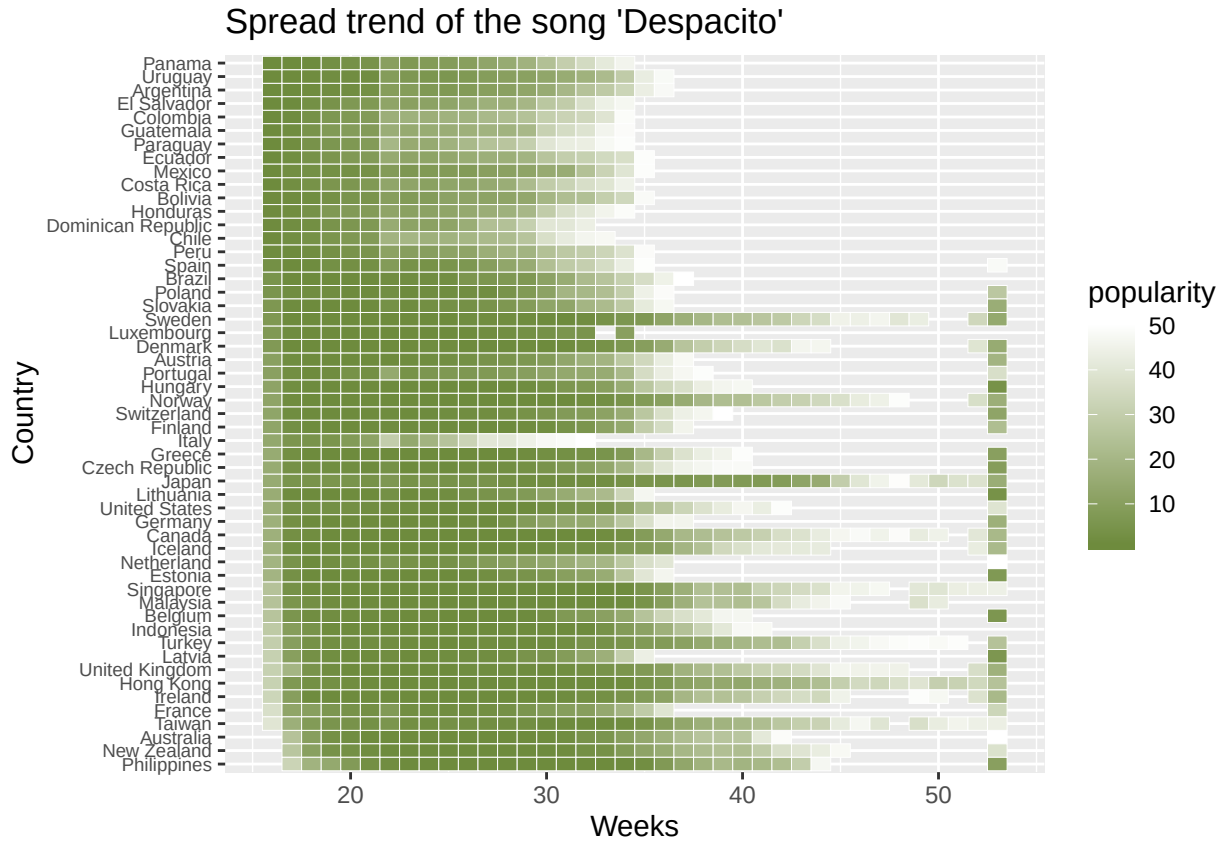
```
##  [1] "pa" "uy" "ar" "sv" "co" "gt" "py" "ec" "mx" "cr" "bo" "hn" "do" "cl"
## [15] "pe" "es" "br" "pl" "sk" "se" "lu" "dk" "at" "pt" "hu" "no" "ch" "fi"
## [29] "it" "gr" "cz" "jp" "lt" "us" "de" "ca" "is" "nl" "ee" "sg" "my" "be"
## [43] "id" "tr" "lv" "gb" "hk" "ie" "fr" "tw" "au" "nz" "ph"
```

```r
country_code[region_order$region]
```

```
##                   pa                   uy                   ar
##              "Panama"            "Uruguay"          "Argentina"
##                   sv                   co                   gt
##        "El Salvador"           "Colombia"          "Guatemala"
##                   py                   ec                   mx
##           "Paraguay"            "Ecuador"             "Mexico"
##                   cr                   bo                   hn
##         "Costa Rica"            "Bolivia"           "Honduras"
##                   do                   cl                   pe
## "Dominican Republic"              "Chile"               "Peru"
##                   es                   br                   pl
##              "Spain"             "Brazil"             "Poland"
##                   sk                   se                   lu
##           "Slovakia"            "Sweden"         "Luxembourg"
##                   dk                   at                   pt
##            "Denmark"            "Austria"           "Portugal"
##                   hu                   no                   ch
##            "Hungary"             "Norway"        "Switzerland"
##                   fi                   it                   gr
##            "Finland"              "Italy"             "Greece"
##                   cz                   jp                   lt
##     "Czech Republic"              "Japan"          "Lithuania"
##                   us                   de                   ca
##      "United States"            "Germany"             "Canada"
##                   is                   nl                   ee
##            "Iceland"         "Netherland"            "Estonia"
```

```
##                  sg                  my                  be
##           "Singapore"           "Malaysia"           "Belgium"
##                  id                  tr                  lv
##           "Indonesia"             "Turkey"            "Latvia"
##                  gb                  hk                  ie
##     "United Kingdom"          "Hong Kong"           "Ireland"
##                  fr                  tw                  au
##              "France"            "Taiwan"         "Australia"
##                  nz                  ph
##        "New Zealand"       "Philippines"
```

```
region_order$region = region_order$region[53:1]

despacito_flow$region<-country_code[despacito_flow$region]
despacito_flow$region <- factor(despacito_flow$region, levels=unique(country_code[region_order$region]))

ggplot(despacito_flow, aes(week_of_year, region)) +
    labs(x = "Weeks", y="Country", title = "Spread trend of the song 'Despacito'") +
    theme(axis.text.y = element_text(size=7)) +
    geom_tile(aes(fill = popularity),colour = "white")+
    scale_fill_gradient(low = "darkolivegreen4", high = "white")
```




In contrast, the Spanish song "Despacito" hit the top 1 ranking in countries of South America the first week after its releases. It then becomes popular in Europe starting from Spain, which is the only Spanish country in Europe. Four weeks later, the song gets to the top 1 in North America, and finally, reach the top 1 in Asia and Oceania after six weeks.