Spotify Data Visualization

Elad Golan

2022-05-23

```
# install.packages(c("tidyverse", "ggmosaic", "glue", "tidymodels", "glmnet", "randomForest", "kernlab"
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggmosaic)
library(glue)
library(tidymodels)
```

```
## Registered S3 method overwritten by 'tune':
##   method                   from
##   required_pkgs.model_spec parsnip
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.1.4 --
```

```
## v broom        0.7.12     v rsample      0.1.1
## v dials        0.1.0      v tune         0.1.6
## v infer        1.0.0      v workflows    0.2.4
## v modeldata    0.1.1      v workflowsets 0.1.0
## v parsnip      0.2.0      v yardstick    0.0.9
## v recipes      0.2.0
```

```
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## x tune::tune()      masks parsnip::tune()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
```

```
## The following object is masked from 'package:scales':
```

```
##
##     alpha
```

```
## The following object is masked from 'package:purrr':
##
##     cross
```

```
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(ggradar)
library(ragg)
library(showtext)
```

```
## Loading required package: sysfonts
```

```
## Loading required package: showtextdb
```

```
library(tvthemes)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(ggwordcloud)
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(stringi)
```

https://github.com/rfordatascience/tidytuesday/tree/master/data/2020/2020-01-21

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

```
## Rows: 32833 Columns: 23
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (10): track_id, track_name, track_artist, track_album_id, track_album_na...
## dbl (13): track_popularity, danceability, energy, key, loudness, mode, speec...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
spotify_songs %>% count(playlist_genre)
```

```
## # A tibble: 6 x 2
##   playlist_genre     n
##   <chr>          <int>
## 1 edm             6043
## 2 latin           5155
## 3 pop             5507
## 4 r&b             5431
## 5 rap             5746
## 6 rock            4951
```

Attention: Apparently each song can repeat a few times with a few genres!

```
spotify_songs %>% count(track_id, sort = TRUE)
```

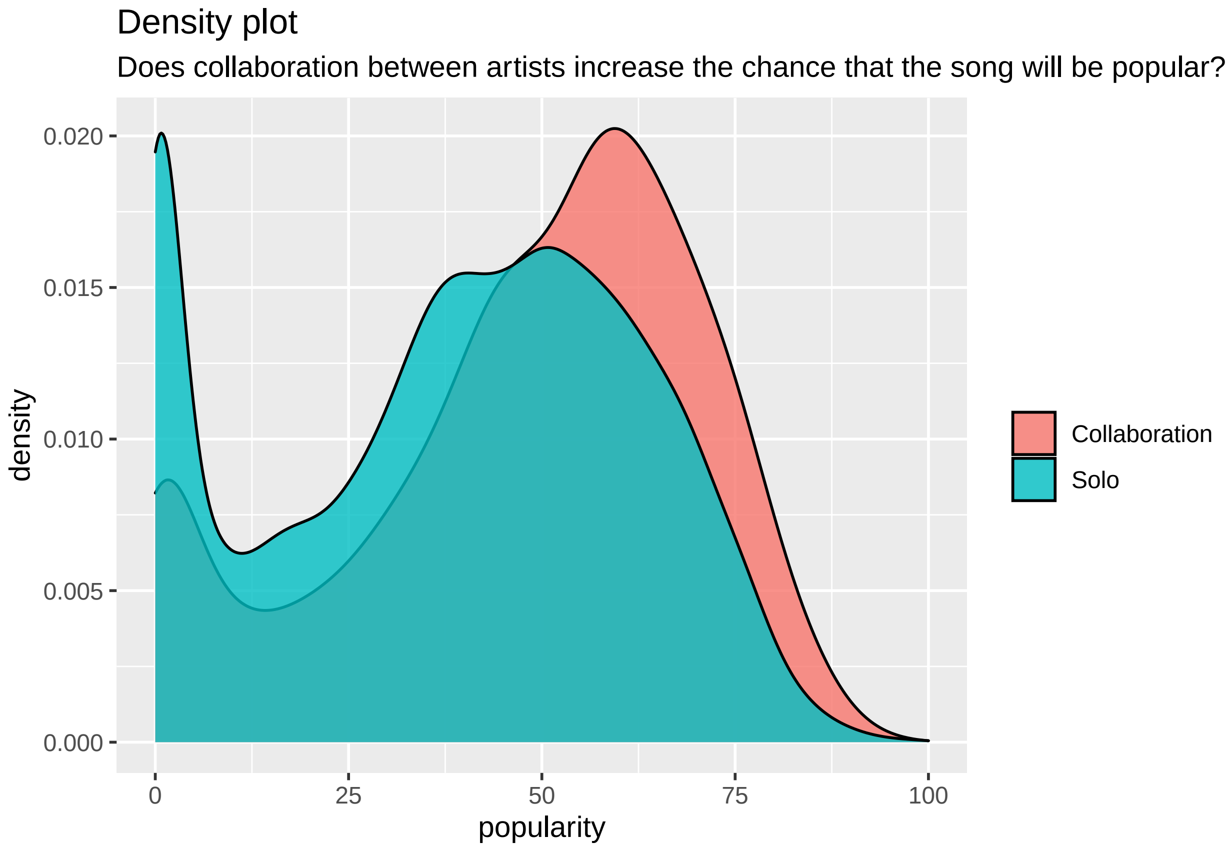
```
## # A tibble: 28,356 x 2
##    track_id                   n
##    <chr>                  <int>
##  1 7BKLCZ1jbUBVqRi2FVlTVw    10
##  2 14sOS5L36385FJ3OL8hew4     9
##  3 3eekarcy7kvN4yt5ZFzltW     9
##  4 0nbXyq5TXYPCO7pr3N8S4I     8
##  5 0qaWEvPkts34WF68r8Dzx9     8
##  6 0rIAC4PXANcKmitJfoqmVm     8
##  7 0sf12qNH5qcw8qpgymFOqD     8
##  8 2b8fOow8UzyDFAE27YhOZM     8
##  9 2Fxmhks0bxGSBdJ92vM42m     8
## 10 2tnVG71enUj33Ic2nFN6kZ     8
## # ... with 28,346 more rows
```

You can either ignore this for now or sample a genre for each song, e.g. with:

```
spotify_songs <- spotify_songs %>%
  group_by(track_id) %>%
  slice_sample(n = 1) %>%
  ungroup()
```

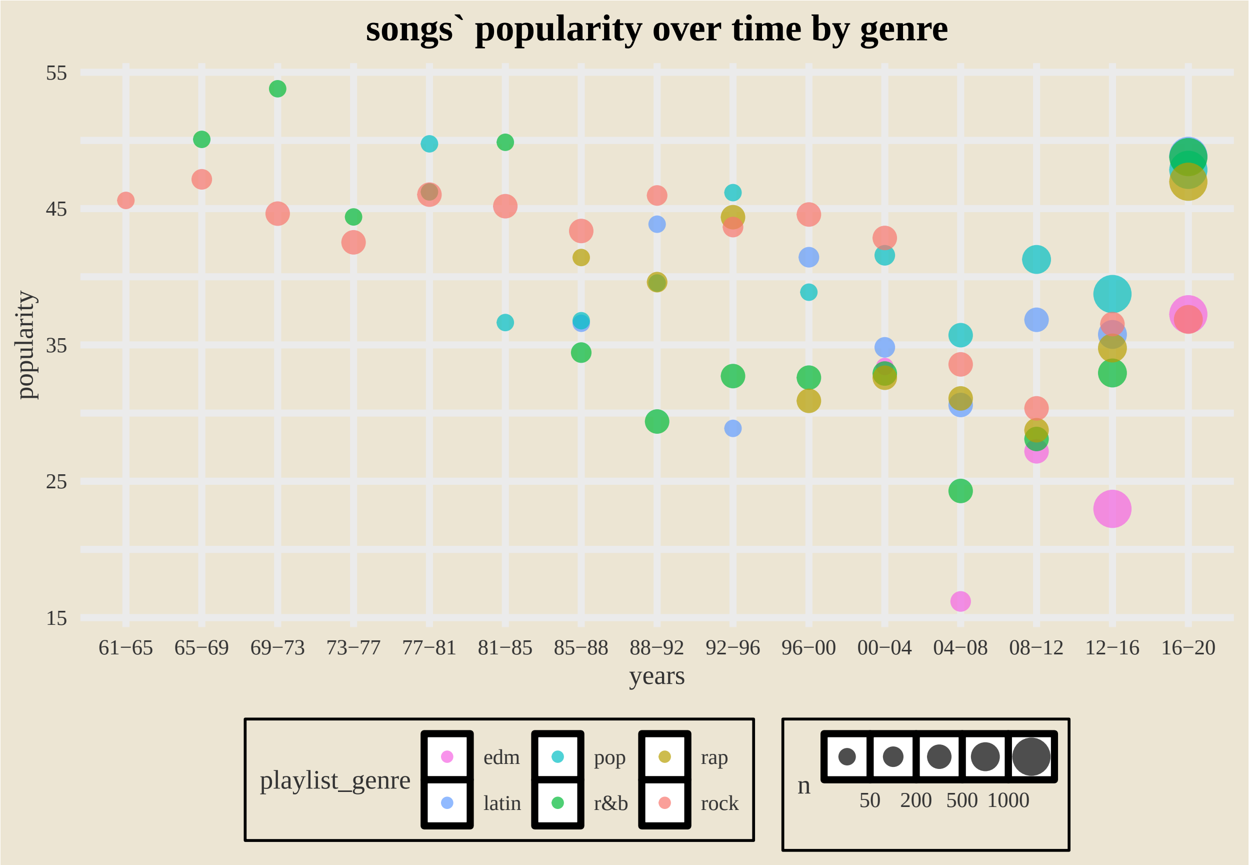
```
p4 <- ggplot(spotify_songs %>% select(track_name,track_album_name, track_popularity) %>% filter(!(grepl


p4 + geom_density(aes(fill=feat), alpha=0.8) + labs(title="Density plot",
         subtitle="Does collaboration between artists increase the chance that the song will be popular
         x="popularity",
         fill="")
```



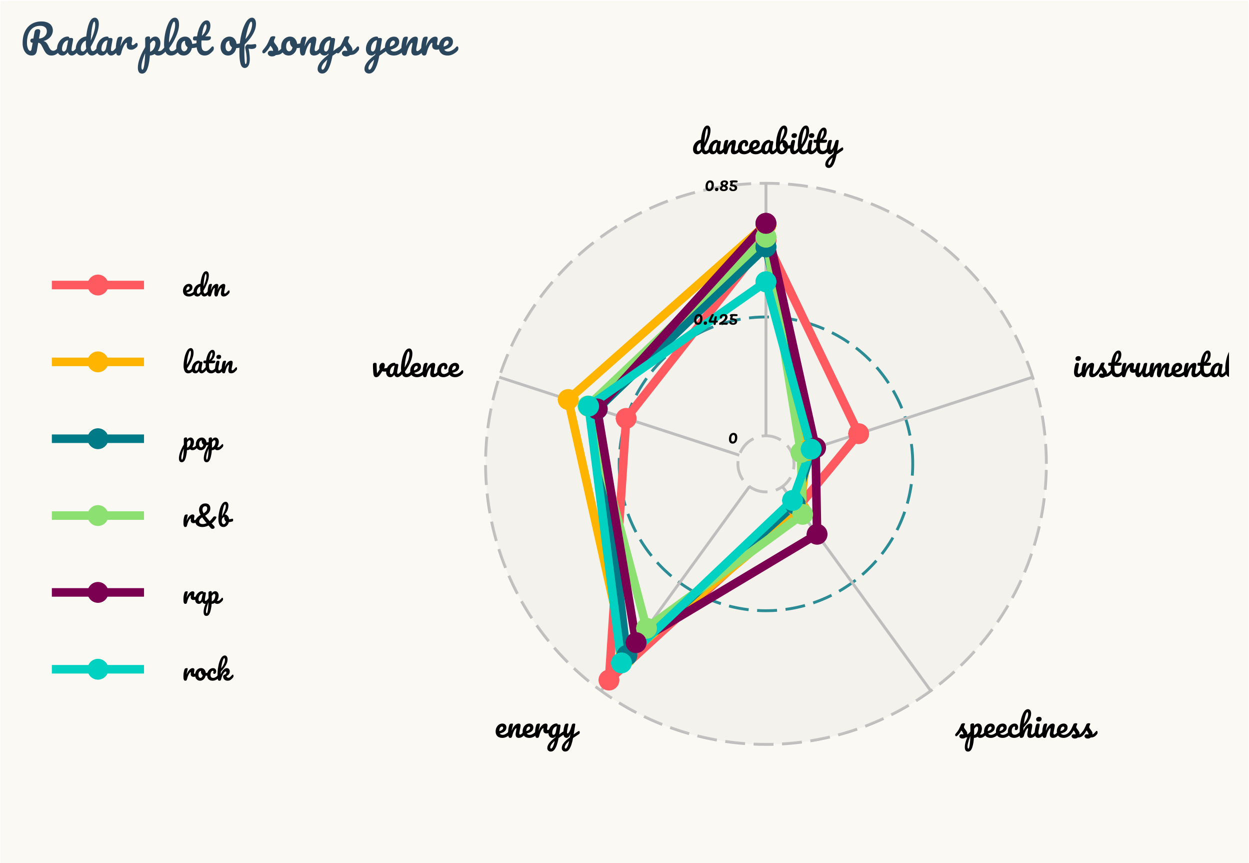
```r
# I cut the years into 4-year groups, except for 85-88 which represents a 3-year period.
ggplot(spotify_songs %>% select(playlist_genre, track_album_release_date, track_popularity) %>%
  mutate(track_album_release_year = cut(as.numeric(substr(track_album_release_date,1,4)), breaks = 16))
  group_by(playlist_genre, track_album_release_year) %>%
  summarise(mean_popularity = mean(track_popularity), n = n(),.groups = 'drop') %>% filter(n >=10),
  aes(track_album_release_year, mean_popularity, size = n, colour = playlist_genre)) + geom_point(alpha
   scale_x_discrete(labels = c("61-65","65-69","69-73", "73-77", "77-81", "81-85", "85-88","88-92", "92
                               "96-00", "00-04", "04-08", "08-12", "12-16", "16-20")) +
  labs(title = "songs` popularity over time by genre", y = "popularity", x="years") + theme_avatar() +
  theme(text= element_text(family  = "serif") ,plot.title = element_text(size=14, face="bold", hjust =
        legend.position= "bottom")
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```



```
font_add_google("pacifico")
showtext_auto()

ggradar(spotify_songs %>% group_by(playlist_genre) %>%
          summarise(across(c(danceability, instrumentalness, speechiness,energy , valence),mean))
        ,font.radar = "pacifico", group.point.size = 3,  axis.label.size = 4, grid.label.size = 3,
          values.radar = c("0", "0.425", "0.85"), grid.min = 0, grid.mid = 0.4, grid.max = 0.85,
        legend.text.size = 10) + labs(title = "Radar plot of songs genre") +
  theme(legend.background = element_blank(), legend.key = element_rect(fill = NA, color = NA),
      plot.background = element_rect(fill = "#fbf9f4", color = "#fbf9f4"),
      panel.background = element_rect(fill = "#fbf9f4", color = "#fbf9f4"),
      plot.title.position = "plot",
      text = element_text( family = "pacifico"),
      plot.title = element_text(
        size = 15,
        face = "bold",
        color = "#2a475e"
    )
  )
```



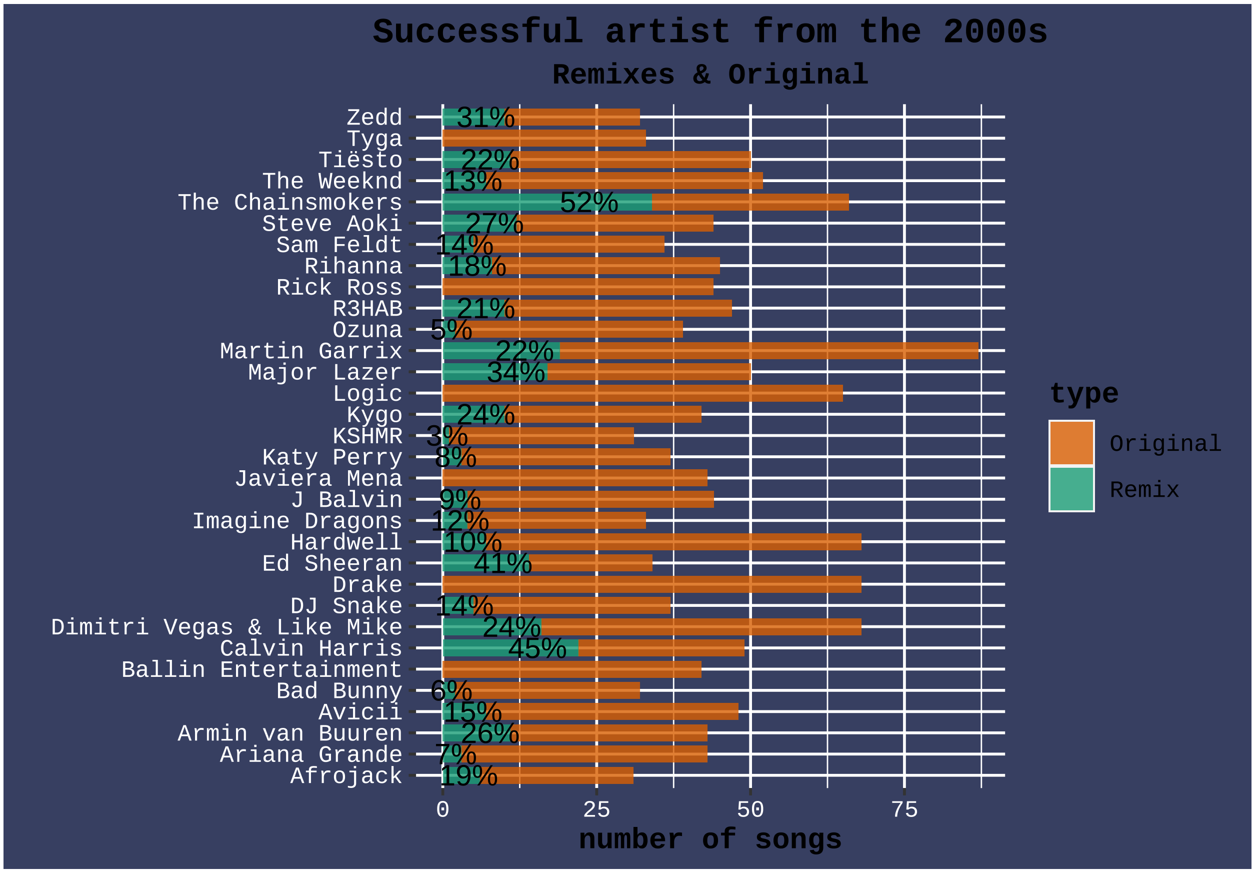
```
#filter the atrists' name only for those who their first album release after 2005.

names_filter <- spotify_songs %>% mutate(as.numeric(substr(track_album_release_date,1,4))) %>%
  filter(track_album_release_date <= 2005) %>% pull(unique(track_artist))

#build the plot based the name_filter
p <- ggplot(spotify_songs %>% mutate(type= case_when(grepl(c("Remix"),track_name) |
                                              grepl(c("Remix"), track_album_name) ~ "Remix",
                                            TRUE ~ "Original")) %>% group_by(track_artist, type) %>%
  summarise(total_type = n(), .group = 'drop')  %>%  inner_join(spotify_songs %>% group_by(track_artist
                                                 summarise(total_songs = n()), by = "track_artist") %>%
  select(track_artist, type, total_type, total_songs) %>%
  mutate(ratio = case_when(type == "Remix" ~ paste0(round(total_type/total_songs, 2)*100, "%"), TRUE ~
  filter(total_songs > 30, !track_artist %in% names_filter),
  aes(x = track_artist, y = total_type ,fill = type))
```

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

```
p + geom_col(alpha=0.8 , width=.8) + coord_flip() +
  geom_text(aes(label=ratio), position =position_stack(vjust = .7)) +
  scale_fill_brewer(palette = "Dark2", direction = -1) + labs( title = "Successful artist from the 2000s
                                                               subtitle = "Remixes & Original" ,
                                                               x = "", y = "number of songs") +
  theme(text = element_text(family = "mono"), plot.title = element_text(face = "bold", hjust = 0.5),
        title = element_text(face = "bold"), plot.subtitle = element_text(hjust = 0.5) ,
        plot.background = element_rect(fill = "#373F61", color = "#373F61"),
        panel.background = element_rect(fill = "#373F61", color = "#373F61"),
        legend.background = element_rect(fill = "#373F61", color = "#373F61"),
        axis.text.x = element_text(color = "white"), axis.text.y = element_text(color = "white"))
```



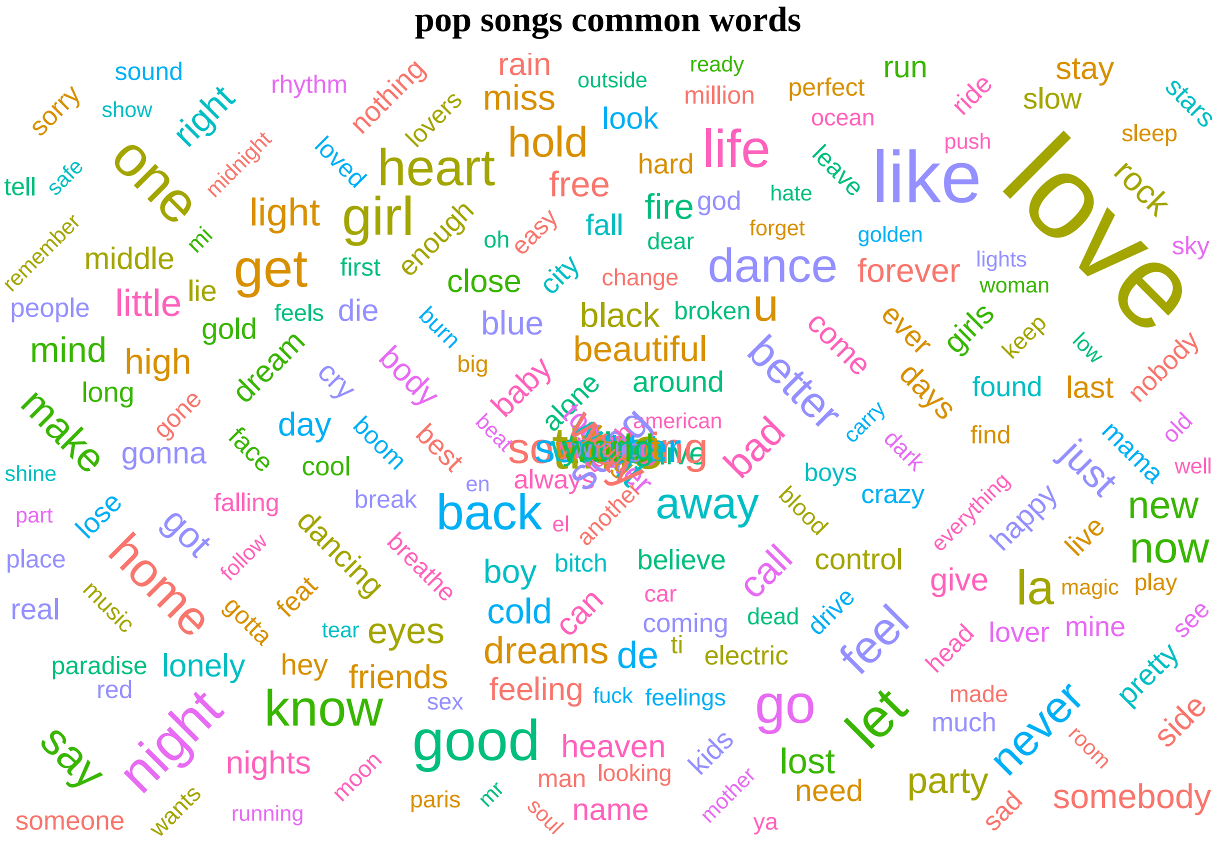
```
#function for cleaning the songs' names from punctuation, stop words, digits and irrelevant words as Re
clean_fun <- function(name){
  name <- removeWords(stri_replace_all_regex(str_to_lower(name) ,
                                     c("\\(feat..*","\\feat..*", "\\[feat..*","\\-.*","\\(.*"), "",
                                     vectorize=FALSE), stopwords())
  name <- str_remove_all(name, "[[:punct:]]")
  return(str_remove_all(name, "[[:digit:]]"))
}

#applying the clean_fun on pop songs.
pop_songs_clean <- spotify_songs %>% filter(playlist_genre == "pop") %>%
  select(track_name) %>%
  mutate(track_name = clean_fun(track_name)) %>% pull()

#making a DF of the freq of the words in the pop songs' names.
counter_words <- as.data.frame(table(unlist(strsplit(pop_songs_clean, " "))))


set.seed(42) # The way the words appear in the chart is random and we would like to get identical resul

#I filtered for words that appear more than 5 times so that the chart does not become too cluttered.
counter_words %>% filter(Freq > 5) %>%
  mutate(angle = 45 * sample(-1:1, n(), replace = TRUE, prob = c(1, 4, 1))) %>%
  ggplot(aes(label = Var1, size = Freq,
             color = factor(sample.int(10, nrow(counter_words %>% filter(Freq > 5)), replace = TRUE)),
             angle = angle)) +
  geom_text_wordcloud(area_corr_power = 1, eccentricity = 1) +
  scale_size_area(max_size = 65) +
  theme_minimal() + labs(title = "pop songs common words") + theme(plot.title = element_text(family = "
```


pop songs common words
sound
sorry
show
right
rhythm
nothing
rain
miss
outside
ready
million
perfect
ocean
run
stay
slow
stars
ride
push
sleep
lovers
look
tell
safe
one
midnight
loved
heart
hold
hard
life
leave
like
love
rock
free
hate
fire
god
light
girl
enough
forget
remember
mi
middle
oh
easy
fall
dear
golden
sky
get
first
close
city
change
dance
forever
lights
woman
people
little
lie
feels
die
burn
blue
black
broken
u
gold
come
ever
girls
keep
low
mind
high
dream
cry
body
big
beautiful
better
nobody
long
baby
around
days
found
last
make
gone
face
day
boom
best
beat
alone
american
carry
dark
find
mama
old
gonna
cool
bad
boys
crazy
just
well
shine
en
always
another
el
break
back
away
blood
everything
happy
new
part
lose
falling
live
got
dancing
breathe
follow
place
home
boy
bitch
believe
call
control
give
la
magic
play
now
real
music
feat
car
cold
can
coming
dead
drive
feel
head
lover
mine
see
gotta
tear
eyes
ti
paradise
lonely
hey
dreams
de
electric
red
friends
feeling
fuck
feelings
made
pretty
sex
know
go
much
say
night
let
never
side
heaven
nights
good
kids
room
man
looking
lost
party
moon
mother
need
paris
mr
soul
name
sad
somebody
someone
wants
running
ya