BrazilSpeaks

Gabriel Saruhashi

3/17/2019

Intro

Copy from BrazilSpeaks report

DATA

Data scraping

The following Python script was used

```
from spotipy.oauth2 import SpotifyClientCredentials
import spotipy
import json
import requests
from bs4 import BeautifulSoup
import pandas as pd
import pprint
import time
from nltk.stem import RSLPStemmer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import os
import re

def setEnvironmentVariables():
    os.environ['SPOTIPY_CLIENT_ID'] = 'c894a126681b4d97a8ccb0cd4a1e0de1'
    os.environ['SPOTIPY_CLIENT_SECRET'] = 'ebf185aaf47e40ab841246986fc7483d'
    os.environ['SPOTIPY_REDIRECT_URI'] = 'https://localhost:8080'
    print('Successfully set the environment variables')

def requestSongInfo(song_title, artist_name):
    base_url = 'https://api.genius.com'
    headers = {'Authorization': 'Bearer ' + '0RIKjAuJB6gohq-1r-w7FzG7W3FcgsL2ZwSRWjUdLLH0E31lUt6T8otW-J
    search_url = base_url + '/search'
    data = {'q': song_title + ' ' + artist_name}
    response = requests.get(search_url, data=data, headers=headers)

    return response

def scrapeSongURL(url):
    print("scraping {}".format(url))
    page = requests.get(url)
    html = BeautifulSoup(page.text, 'html.parser')
    lyrics = html.find('div', class_='lyrics').get_text()
```

```python
    # try:
    # release_date = html.find('span', class_='metadata_unit-info metadata_unit-info--text_only').get_t
    # print(release_date)

    return lyrics


# Preprocessing of the Lyrics
def preprocessLyrics(sentence):
    # stemmer=RSLPStemmer()

    sentence = sentence.lower()
    # remove all the annotations (e.g '[refrão 1] Bla bla')
    sentence = re.sub(r'[\(\[].*?[\)\]]', "", str(sentence))

    # get Portuguese stopwords
    file_stop = open("pt_stopwords.txt")
    body_stop = file_stop.read()
    stop = body_stop.split()

    token_words = word_tokenize(sentence)
    processed_sentence=[]

    for word in token_words:
        if word not in stop:
            processed_sentence.append(word)
            # stem_sentence.append(stemmer.stem(word))
            processed_sentence.append(" ")

    # remove all the annotations within [] and ()

    return "".join(processed_sentence)

def extractLyrics(song_title, artist_name):
    # Search for matches in request response
    response = requestSongInfo(song_title, artist_name)
    json = response.json()
    remote_song_info = None


    for hit in json['response']['hits']:
        if artist_name.lower() in hit['result']['primary_artist']['name'].lower():
            remote_song_info = hit
            break

    # Extract lyrics from URL if song was found
    if remote_song_info:
        song_url = remote_song_info['result']['url']
        lyrics = scrapeSongURL(song_url)
        lyrics = lyrics.replace('\n', ' ')
        lyrics = preprocessLyrics(lyrics)

        return lyrics
    else:
        print("Could not find lyrics for given artist and song title")
        return ""

def getSpotifySongFeatures(uri):
    song_features = sp.audio_features(uri)
    song_features = song_features[0]

    extra_fields = ["track_href", "uri", "analysis_url", "type"]

    for field in extra_fields:
        song_features.pop(field)

    return song_features

def getSpotifyArtistInfo(artist_id):
    artist = {}

    info = sp.artist(artist_id)

    artist["artist_genres"] = info["genres"][0]
    artist["artist_name"] = info["name"]
    if info["images"]:
        artist["artist_photo"] = info["images"][0]["url"]
    else:
        artist["artist_photo"] = ""
    artist["artist_popularity"] = info["popularity"]
    artist["artist_sp_followers"] = info["followers"]["total"]

    return artist

def processSpotifyPlaylistCSV(uri, csv_filepath, song_class):
    start_time = time.time()

    username = uri.split(':')[2]
    playlist_id = uri.split(':')[4]

    # get the relevant playlist
    results = sp.user_playlist(username, playlist_id)

    tracks = results["tracks"]["items"]

    # define main data frame that will store
    df = pd.DataFrame()
    index = 0
    for obj in tracks:
        track = obj["track"]
        song = {}

        # preprocessed song name
        song_name = re.split(r' -| \(', track["name"])[0]

        # song["artist"] = artist
        song["song_sp_uri"] = track["uri"]
        song["song_name"] = song_name
        song["song_isrc"] = track["external_ids"]["isrc"]
        song["song_popularity"] = track["popularity"]
        song_features = getSpotifySongFeatures(track["uri"])

        artist_info = getSpotifyArtistInfo(track["artists"][0]["id"])
        song["song_lyrics"] = extractLyrics(song["song_name"], artist_info["artist_name"])
        song["class"] = song_class

        # concatenating all dictionaries
        song = {**song, **song_features, **artist_info}

        # TODO incorporate this somehow
        # song_analysis = sp.audio_analysis(track["uri"])

        df = pd.concat([df, pd.DataFrame(song, index=[index])])
        index += 1

    print("Scraping process took {} s. Now storing intermediate results for this class of music".format
    df.to_csv(csv_filepath)

    return df

# Uncomment this section if you'd like to start the datascraping script
'''
PROTEST_URI = 'spotify:user:gabriel_saruhashi:playlist:4Tp4QcTk9rNikjmaDg5VxJ'
JOVEM_GUARDA_URI = 'spotify:user:gabriel_saruhashi:playlist:1JZoMCGiAKcXrgBzbKW931'
PROTEST_CLASSNAME = "Protest"
JOVEM_GUARDA_CLASSNAME = "Jovem Guarda"
setEnvironmentVariables()

client_credentials_manager = SpotifyClientCredentials()
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)

# create csv with data from spotify
protest_df = processSpotifyPlaylistCSV(PROTEST_URI, "protest.csv", "Protest")
jovem_guarda_df = processSpotifyPlaylistCSV(JOVEM_GUARDA_URI, "jovem_guarda.csv", "Jovem Guarda")

# store final output
res_df = pd.concat([protest_df, jovem_guarda_df])
res_df.to_csv("brz_dictatorship.csv")
'''
```

The following Python script was used to create the WhoSampled dataset

```
# SOURCE: https://github.com/cpease00/Spotify-Samples/blob/master/data_science/spotify/package/whosampl
import requests
from bs4 import BeautifulSoup
import urllib3
import pandas as pd
import json
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
http = urllib3.PoolManager()
```

```python
def retrieve_song_link(song_name, artist_name=None):
    query = song_name.replace(' ', '%20')
    url = 'https://www.whosampled.com/search/tracks/?q={}'.format(query)
    r = http.request('GET', url)
    content = r.data
    soup = BeautifulSoup(content, 'html.parser')
    stuff = soup.findAll('li', attrs={'class': "listEntry"})
    if stuff:
        link = [i.a for i in stuff][0].get('href')
        return link
    # else:
#         print('{} not found on whosampled'.format(song_name))

def retrieve_samples_v2(song_name, link):
    samples = []
    sampled_by = []
    s = http.request('GET', 'https://www.whosampled.com'+link)
    content1 = s.data
    soup = BeautifulSoup(content1, 'html.parser')
    listed = [i.text for i in soup.findAll('div', attrs={'class':'list bordered-list'})]
    if len(listed) == 2:
        there_in = [i.split('\n') for i in list(filter(None, listed[0].split('\t')))][:-1]
        there_out = [i.split('\n') for i in list(filter(None, listed[1].split('\t')))][:-1]
        for j in there_out:
            sampled_by.append({'query':song_name, 'type':j[-7], 'genre':j[-6], 'title':j[-3], 'artist':j
    else:
        there_in = [i.split('\n') for i in list(filter(None, listed[0].split('\t')))][:-1]
    for i in there_in:
        samples.append({'query':song_name, 'type':i[-7], 'genre':i[-6], 'title':i[-3], 'artist':i[-2].re
    return samples, sampled_by

def getme_thesamples(song_name, artist_name):
    link = retrieve_song_link(song_name, artist_name)
    if link:
        samples, sampled_by = retrieve_samples_v2(song_name, link)
        return samples, sampled_by
    else:
        return None, None

def get_whosampled_playlist(loaded_playlist):
    samples = []
    new_playlist = []
    df = pd.DataFrame()
    index = 0
    print('SPOTIFY PLAYLIST DISCOVERED: \n')
    # for i in loaded_playlist:
        #print(i['track']+' by '+i['artist'][0])
    for i in loaded_playlist:
        samples, sampled_by = getme_thesamples(i['track'], i['artist'][0])
        if samples:
            for sample in samples:
                df = pd.concat([df, pd.DataFrame(sample, index=[index])])
                index += 1
            print("Samples for " + i['track']+' by '+i['artist'][0] + json.dumps(samples))
            print('\n')
            new_playlist.append(samples)

    lst = [i for j in new_playlist for i in j]

    #df.to_csv("whosampled_list.csv")
    return lst
```

Overview of the data

Make a LIST of all variables you use –describe units, anything I should know.

```
music = read.csv("brz_dictatorship.csv", as.is=TRUE)
dim(music)
```

```
## [1] 200  26
```

```
str(music)
```

```
## 'data.frame':    200 obs. of  26 variables:
##  $ X                 : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ song_sp_uri       : chr  "spotify:track:0VUgbCK0k8QWGpLiEV8YYZ" "spotify:track:2GAFZG9Z7UGS1iMm4I
##  $ song_name         : chr  "Cálice" "Apesar De Você" "Roda-Viva" "Como nossos pais" ...
##  $ song_isrc         : chr  "BRPGD7800015" "BRPGD7800024" "BRSGL6800006" "BRWMB9705419" ...
##  $ song_popularity   : int  50 57 53 27 45 47 49 23 58 54 ...
##  $ song_lyrics       : chr  "pai , afasta mim calice pai , afasta mim calice pai , afasta mim calic
##  $ class             : chr  "Protest" "Protest" "Protest" "Protest" ...
##  $ danceability      : num  0.596 0.568 0.502 0.463 0.609 0.442 0.67 0.373 0.439 0.567 ...
##  $ energy            : num  0.372 0.574 0.512 0.337 0.76 0.417 0.669 0.366 0.716 0.333 ...
##  $ key               : int  4 4 10 8 2 4 2 0 0 4 ...
##  $ loudness          : num  -9.39 -8.99 -8.1 -13.18 -10.17 ...
##  $ mode              : int  1 0 0 1 1 1 1 1 1 1 ...
##  $ speechiness       : num  0.0606 0.0683 0.0337 0.154 0.087 0.047 0.0476 0.0343 0.0341 0.0425 ...
##  $ acousticness      : num  8.48e-01 4.68e-01 7.93e-01 8.87e-01 3.32e-01 7.73e-01 8.31e-01 8.82e-01
##  $ instrumentalness  : num  0.00 0.00 1.80e-05 0.00 4.86e-05 3.54e-06 6.89e-05 7.70e-02 1.41e-01 0.0
##  $ liveness          : num  0.331 0.362 0.235 0.203 0.161 0.229 0.101 0.198 0.0912 0.0736 ...
##  $ valence           : num  0.293 0.68 0.651 0.269 0.88 0.276 0.927 0.179 0.4 0.53 ...
##  $ tempo             : num  123.1 107.8 133.2 96.6 92 ...
##  $ id                : chr  "0VUgbCK0k8QWGpLiEV8YYZ" "2GAFZG9Z7UGS1iMm4Idrnr" "06ND7qqsmIRCuWdQNQIl
##  $ duration_ms       : int  241867 235547 233400 280627 239187 229733 131000 185507 178733 258027 .
##  $ time_signature    : int  4 4 4 4 4 4 4 4 4 4 ...
##  $ artist_genres     : chr  "bossa nova" "bossa nova" "bossa nova" "bossa nova" ...
##  $ artist_name       : chr  "Chico Buarque" "Chico Buarque" "Chico Buarque" "Belchior" ...
##  $ artist_photo      : chr  "https://i.scdn.co/image/c1a6ae9e79561abeb0bd97507cf7a26696469df0" "http
##  $ artist_popularity : int  63 63 63 57 63 64 63 29 69 57 ...
##  $ artist_sp_followers: int  542214 542214 542214 299597 829961 532021 542214 16490 2440436 299597 .
```

Create corpus for text mining

```
library(tm)
```

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

```
jg <- paste(music$song_lyrics[music$class=="Jovem Guarda"], collapse = '')
protest <- paste(music$song_lyrics[music$class=="Protest"], collapse = '')
```

```
docs <- Corpus(VectorSource(c(jg, protest)))
```

Data Cleaning

describe the cleaning process you used on your data. Talk about what issues you encountered.

```
# Remove numbers
docs <- tm_map(docs, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(docs, removeNumbers): transformation drops
## documents
```

```
# Remove english common stopwords
docs <- tm_map(docs, removeWords, stopwords("portuguese"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, stopwords("portuguese")):
## transformation drops documents
```

```
# Remove punctuations
docs <- tm_map(docs, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(docs, removePunctuation): transformation
## drops documents
```

```
# Eliminate extra white spaces
docs <- tm_map(docs, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(docs, stripWhitespace): transformation drops
## documents
```

```
# Remove your own stop word
# specify your stopwords as a character vector
docs <- tm_map(docs, removeWords, c("mim", "pra", "vai"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, c("mim", "pra", "vai")):
## transformation drops documents
```

Descriptive Plots & Summary Information

(Plots should be clearly labeled, well formatted,and display an aesthetic sense.) Examine correlations between continuous variables

Basic tests with the different classes

Visualizing Correlations

Your first task is to visually examine the correlations with the `corrplot.mixed`.

```
#Load the corrplot package
library(corrplot)
```

```
## corrplot 0.84 loaded
```

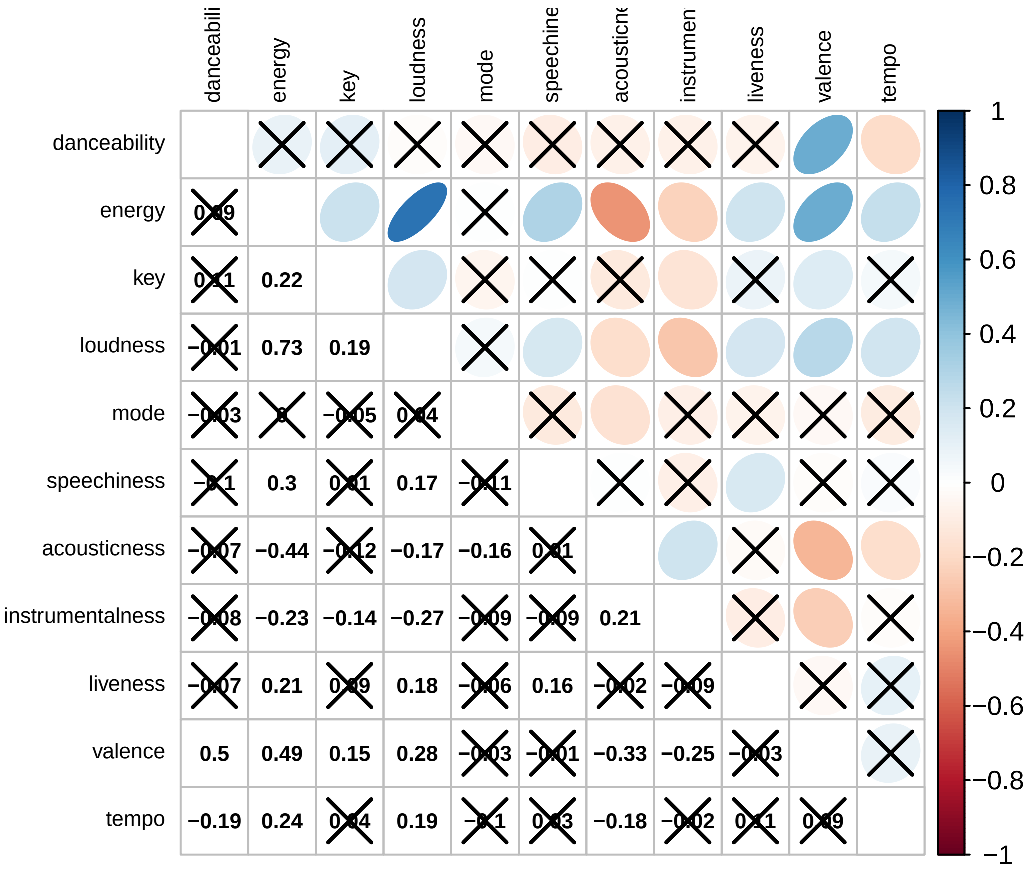
```
df <- na.omit(music[,8:18])
cor1 <- cor(df, use="pairwise.complete.obs")

#round cor1 to 2 decimal places and display the result.
(round(cor1, 2))
```

```
##                  danceability energy   key loudness  mode speechiness
## danceability             1.00   0.09  0.11    -0.01 -0.03       -0.10
## energy                   0.09   1.00  0.22     0.73  0.00        0.30
## key                      0.11   0.22  1.00     0.19 -0.05        0.01
## loudness                -0.01   0.73  0.19     1.00  0.04        0.17
## mode                    -0.03   0.00 -0.05     0.04  1.00       -0.11
## speechiness             -0.10   0.30  0.01     0.17 -0.11        1.00
## acousticness            -0.07  -0.44 -0.12    -0.17 -0.16        0.01
## instrumentalness        -0.08  -0.23 -0.14    -0.27 -0.09       -0.09
## liveness                -0.07   0.21  0.09     0.18 -0.06        0.16
## valence                  0.50   0.49  0.15     0.28 -0.03       -0.01
## tempo                   -0.19   0.24  0.04     0.19 -0.10        0.03
##                  acousticness instrumentalness liveness valence tempo
## danceability            -0.07            -0.08    -0.07    0.50 -0.19
## energy                  -0.44            -0.23     0.21    0.49  0.24
## key                     -0.12            -0.14     0.09    0.15  0.04
## loudness                -0.17            -0.27     0.18    0.28  0.19
## mode                    -0.16            -0.09    -0.06   -0.03 -0.10
## speechiness              0.01            -0.09     0.16   -0.01  0.03
## acousticness             1.00             0.21    -0.02   -0.33 -0.18
## instrumentalness         0.21             1.00    -0.09   -0.25 -0.02
## liveness                -0.02            -0.09     1.00   -0.03  0.11
## valence                 -0.33            -0.25    -0.03    1.00  0.09
## tempo                   -0.18            -0.02     0.11    0.09  1.00
```

```
#get the array index (row, col) for the predictors of maximum non-one correlation value
#By ignoring correlations cor1 == 1, you discard the matrix main diagonal
#(correlation of a variable with itself is always 1).
maxloc <- which(cor1 == max(cor1[cor1<1]), arr.ind = TRUE)

#get the column names of the two variables with highest correlation by index
#note that maxloc[2, ] is the same as maxloc[1, ], but flipped
names(df)[maxloc[1,]]
```

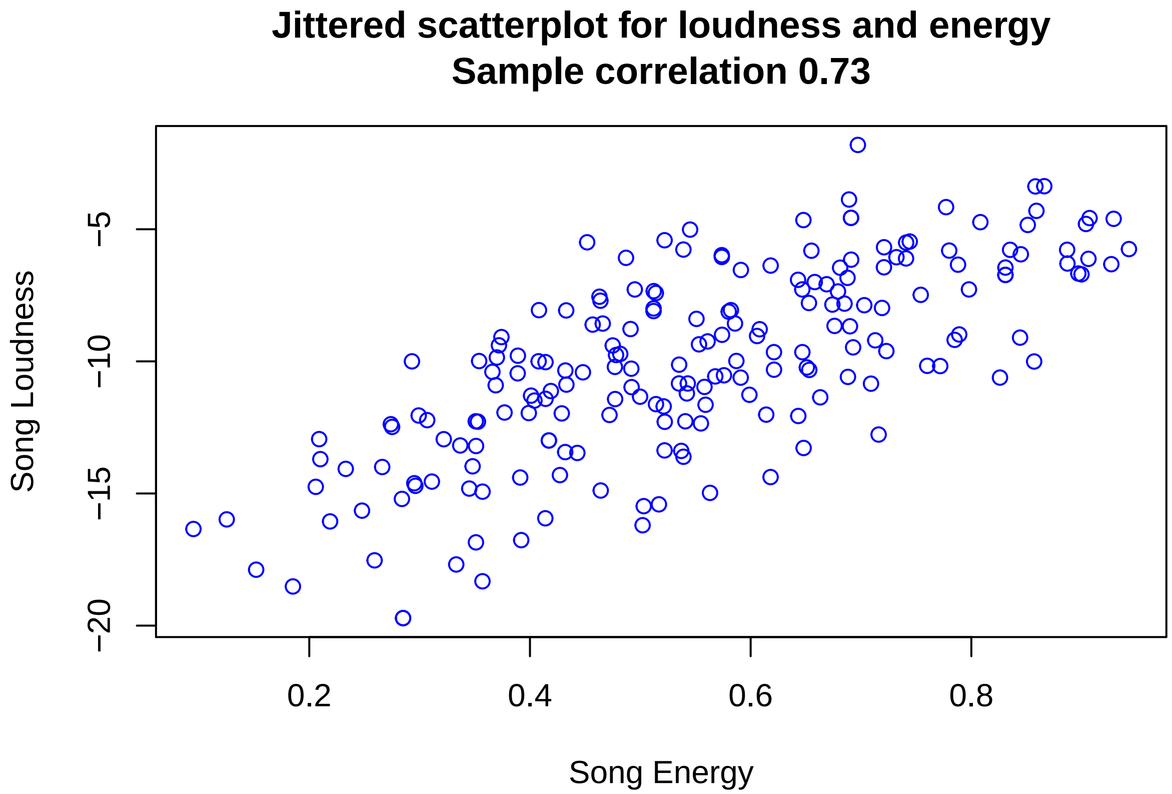
```
## [1] "loudness" "energy"
```

```
#Create an object called sigcorr that has the results of cor.mtest for columns 10-23 of the crime data.
sigcorr <- cor.mtest(df, conf.level = .95)

#Use corrplot.mixed to display confidence ellipses, pairwise correlation values, and put on 'X' over no
corrplot.mixed(cor1,
               lower.col="black",
               upper = "ellipse",
               tl.col = "black",
               number.cex=.7,
               tl.pos = "lt",
               tl.cex=.7,
               p.mat = sigcorr$p,
               sig.level = .05)
```



Now let's examine more closely the correlation between the two variables with highest correlation.

```
plot(jitter(loudness) ~ jitter(energy),
     data=df,
     xlab="Song Energy",
     ylab="Song Loudness",
     main=paste("Jittered scatterplot for loudness and energy\nSample correlation", round(cor1[maxloc[1
     col="blue")
```



By adding a small amount of random normally distributed noise, we can see observations and their densities more clearly, and now it looks like there is a strongcorrelation between the two questions (as demonstrated by the slighlty linear concentration in density).

Stepwise Regression

We are now going to proceed with performing stepwise regression. In particular, we're going to fit a model that looks at possible predictors of the class of the song. To do this, I'm making a new dataset called `music2` which contains the relevant columns (notice I'm putting the response variable FIRST). Be sure to remove the option `eval = F`.

```
music2 <- music[,c(5, 8:18)]
music2 <- na.omit(music2)

#TODO why are factors not allowed for this
#music2$class <- as.factor(music2$class)

names(music2)
```

```
##  [1] "song_popularity"  "danceability"     "energy"
##  [4] "key"              "loudness"         "mode"
##  [7] "speechiness"      "acousticness"     "instrumentalness"
## [10] "liveness"         "valence"          "tempo"
```

```
dim(music2)
```

```
## [1] 200  12
```

```
str(music2)
```

```
## 'data.frame':    200 obs. of  12 variables:
```

```
##  $ song_popularity : int  50 57 53 27 45 47 49 23 58 54 ...
##  $ danceability    : num  0.596 0.568 0.502 0.463 0.609 0.442 0.67 0.373 0.439 0.567 ...
##  $ energy          : num  0.372 0.574 0.512 0.337 0.76 0.417 0.669 0.366 0.716 0.333 ...
##  $ key             : int  4 4 10 8 2 4 2 0 0 4 ...
##  $ loudness        : num  -9.39 -8.99 -8.1 -13.18 -10.17 ...
##  $ mode            : int  1 0 0 1 1 1 1 1 1 1 ...
##  $ speechiness     : num  0.0606 0.0683 0.0337 0.154 0.087 0.047 0.0476 0.0343 0.0341 0.0425 ...
##  $ acousticness    : num  8.48e-01 4.68e-01 7.93e-01 8.87e-01 3.32e-01 7.73e-01 8.31e-01 8.82e-01 2.
##  $ instrumentalness: num  0.00 0.00 1.80e-05 0.00 4.86e-05 3.54e-06 6.89e-05 7.70e-02 1.41e-01 0.00
##  $ liveness        : num  0.331 0.362 0.235 0.203 0.161 0.229 0.101 0.198 0.0912 0.0736 ...
##  $ valence         : num  0.293 0.68 0.651 0.269 0.88 0.276 0.927 0.179 0.4 0.53 ...
##  $ tempo           : num  123.1 107.8 133.2 96.6 92 ...
```

```
total_vars <- dim(music2)[2]
```

Perform best subsets regression using the `regsubsets` function in the `leaps` package. Save the results in an object called `mod2`. Get the summary of `mod2` and save the results in an object called `mod2sum`. Display `mod2sum$which` to get a sense of which variables are included at each step of best subsets.

```
library('leaps')

#use all variables in crime2 (20 variables)
mod2 <- regsubsets(song_popularity ~ ., data=music2, nvmax=total_vars)
mod2sum <- summary(mod2)
mod2sum$which
```

```
##    (Intercept) danceability energy   key loudness  mode speechiness
## 1         TRUE        FALSE  FALSE FALSE    FALSE FALSE       FALSE
## 2         TRUE        FALSE  FALSE FALSE    FALSE  TRUE       FALSE
## 3         TRUE        FALSE  FALSE FALSE    FALSE  TRUE       FALSE
## 4         TRUE        FALSE  FALSE FALSE    FALSE  TRUE       FALSE
## 5         TRUE        FALSE  FALSE FALSE    FALSE  TRUE        TRUE
## 6         TRUE         TRUE  FALSE FALSE    FALSE  TRUE        TRUE
## 7         TRUE         TRUE  FALSE FALSE     TRUE  TRUE        TRUE
## 8         TRUE         TRUE  FALSE FALSE     TRUE  TRUE        TRUE
## 9         TRUE         TRUE  FALSE FALSE     TRUE  TRUE        TRUE
## 10        TRUE         TRUE   TRUE FALSE     TRUE  TRUE        TRUE
## 11        TRUE         TRUE   TRUE  TRUE     TRUE  TRUE        TRUE
##    acousticness instrumentalness liveness valence tempo
## 1          TRUE            FALSE    FALSE   FALSE FALSE
## 2          TRUE            FALSE    FALSE   FALSE FALSE
## 3          TRUE            FALSE    FALSE    TRUE FALSE
## 4          TRUE             TRUE    FALSE    TRUE FALSE
## 5          TRUE             TRUE    FALSE    TRUE FALSE
## 6          TRUE             TRUE    FALSE    TRUE FALSE
## 7          TRUE             TRUE    FALSE    TRUE FALSE
## 8          TRUE             TRUE    FALSE    TRUE  TRUE
## 9          TRUE             TRUE     TRUE    TRUE  TRUE
## 10         TRUE             TRUE     TRUE    TRUE  TRUE
## 11         TRUE             TRUE     TRUE    TRUE  TRUE
```

Now, let's examine the best model according to highest r-squared, etc.

```
modnum = which.max(mod2sum$rsq)

#Which variables are in model 12
```

```
names(music2)[mod2sum$which[modnum,]][-1]
```

```
##  [1] "danceability"     "energy"           "key"
##  [4] "loudness"         "mode"             "speechiness"
##  [7] "acousticness"     "instrumentalness" "liveness"
## [10] "valence"          "tempo"
```

```
#Fit this model and show results
musictemp <- music2[,mod2sum$which[modnum,]]

summary(lm(song_popularity ~ .,data=musictemp))
```

```
##
## Call:
## lm(formula = song_popularity ~ ., data = musictemp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.770  -9.199   1.369  10.531  32.899
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       48.20764   13.68885   3.522 0.000538 ***
## danceability       7.72240   11.08497   0.697 0.486879
## energy             2.28472   11.61569   0.197 0.844281
## key                0.02144    0.35644   0.060 0.952097
## loudness           0.19516    0.49409   0.395 0.693295
## mode              -6.39425    2.56830  -2.490 0.013654 *
## speechiness      -36.29120   26.05897  -1.393 0.165369
## acousticness     -12.91296    5.16885  -2.498 0.013340 *
## instrumentalness -23.67305   12.17040  -1.945 0.053250 .
## liveness           1.53966    5.76663   0.267 0.789766
## valence          -15.67833    6.80853  -2.303 0.022388 *
## tempo             -0.02327    0.04651  -0.500 0.617425
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.88 on 188 degrees of freedom
## Multiple R-squared:  0.09774,    Adjusted R-squared:  0.04495
## F-statistic: 1.851 on 11 and 188 DF,  p-value: 0.04826
```

```
modnum <- which.max(mod2sum$adjr2)

#Which variables are in model 12
names(music2)[mod2sum$which[modnum,]][-1]
```

```
## [1] "mode"             "speechiness"      "acousticness"
## [4] "instrumentalness" "valence"
```

```
#Fit this model and show results
musictemp <- music2[,mod2sum$which[modnum,]]
summary(lm(song_popularity ~ .,data=musictemp))
```

```
##
## Call:
## lm(formula = song_popularity ~ ., data = musictemp)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.391  -8.639   1.188  10.007  32.930
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         46.770      5.416   8.635 2.14e-15 ***
## mode                -6.233      2.499  -2.494  0.01346 *
## speechiness        -32.152     23.503  -1.368  0.17290
## acousticness       -12.832      4.548  -2.822  0.00527 **
## instrumentalness  -25.443     11.605  -2.192  0.02954 *
## valence            -12.281      5.158  -2.381  0.01824 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.7 on 194 degrees of freedom
## Multiple R-squared:  0.09072,    Adjusted R-squared:  0.06729
## F-statistic: 3.871 on 5 and 194 DF,  p-value: 0.00229
```

BIC

```
modnum = which.min(mod2sum$bic)

#Which variables are in model 12
names(music2)[mod2sum$which[modnum,]][-1]
```

```
## [1] "acousticness"
```

```
#Fit this model and show results
musictemp <- music2[,mod2sum$which[modnum,]]

summary(lm(song_popularity ~ .,data=musictemp))
```

```
##
## Call:
## lm(formula = song_popularity ~ ., data = musictemp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.775 -11.039   1.713  10.730  35.189
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    30.538      2.141   14.26   <2e-16 ***
## acousticness   -9.624      4.297   -2.24   0.0262 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.09 on 198 degrees of freedom
## Multiple R-squared:  0.02471,    Adjusted R-squared:  0.01979
## F-statistic: 5.017 on 1 and 198 DF,  p-value: 0.02621
```

CP

```
(modCP <- min(c(1:length(mod2sum$cp))[mod2sum$cp < c(1:length(mod2sum$cp))+1]))
```

```
## [1] 3
```

```r
#Which variables are in model 2
names(music2)[mod2sum$which[modCP,]][-1]
```

```
## [1] "mode"         "acousticness" "valence"
```

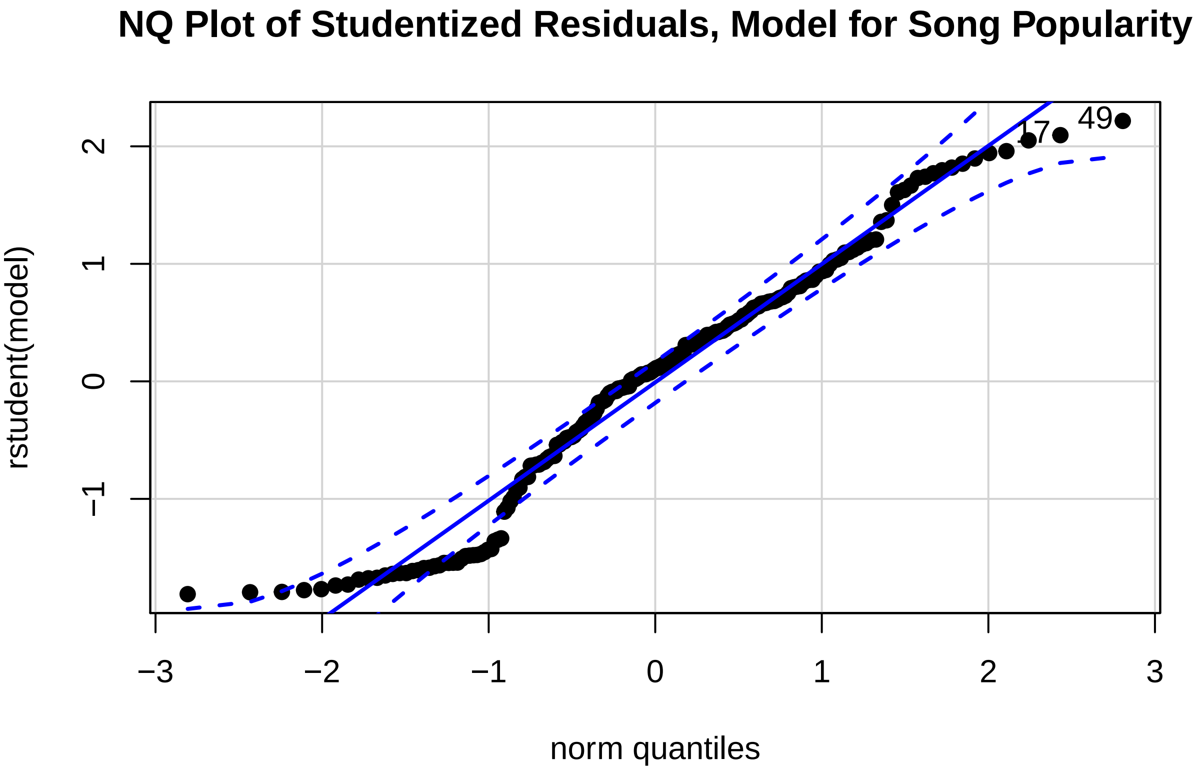
```r
#Fit this model and show results
musictemp <- music2[,mod2sum$which[modCP,]]
summary(lm(song_popularity ~ .,data=musictemp))
```

```
##
## Call:
## lm(formula = song_popularity ~ ., data = musictemp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.733  -8.423   1.979  10.724  32.436
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    42.783      5.129   8.341 1.29e-14 ***
## mode           -5.437      2.500  -2.175  0.03081 *
## acousticness  -13.974      4.560  -3.064  0.00249 **
## valence        -9.891      5.104  -1.938  0.05408 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.86 on 196 degrees of freedom
## Multiple R-squared:  0.062,  Adjusted R-squared:  0.04765
## F-statistic: 4.319 on 3 and 196 DF,  p-value: 0.005646
```

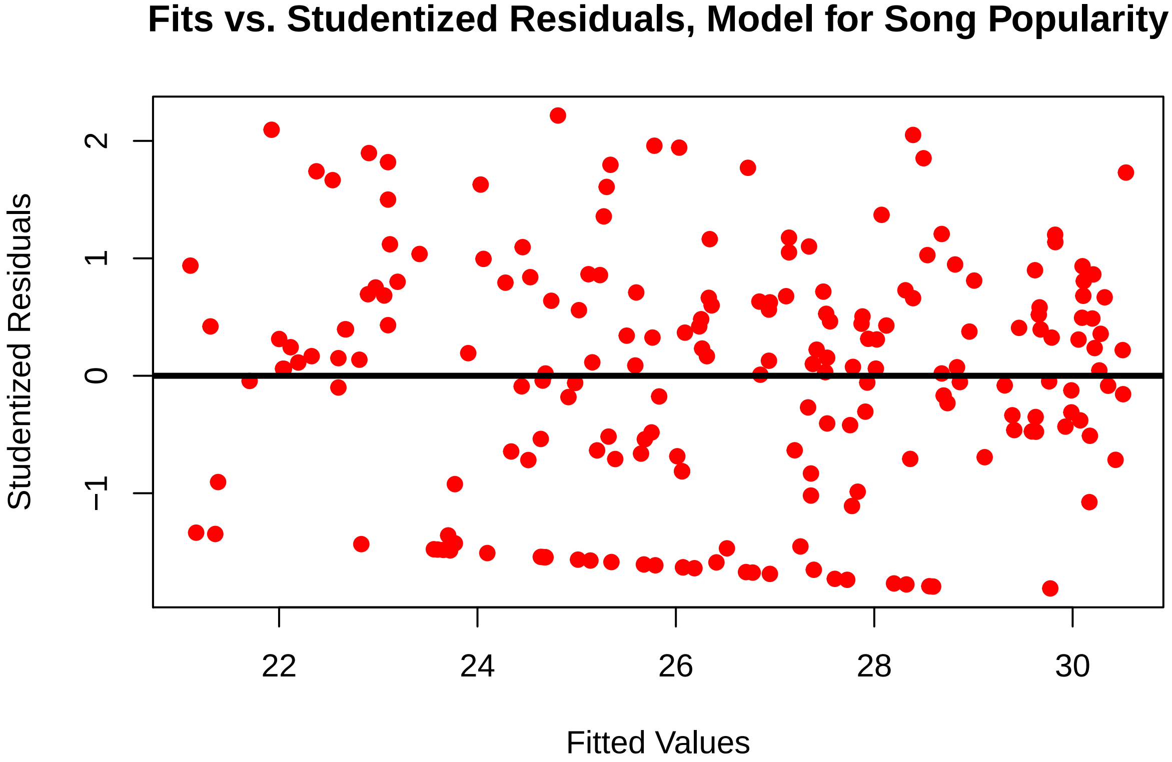
```r
musicfinal <- music2[,mod2sum$which[1,]]
modfin <- lm(song_popularity ~ .,data=musicfinal)

#get new function for pairs plotn AND get myResPlots function
source("http://www.reuningscherer.net/s&ds230/Rfuncs/regJDRS.txt")
```

```
##
## Attaching package: 'olsrr'

## The following object is masked from 'package:datasets':
##
##     rivers

## Loading required package: carData
```

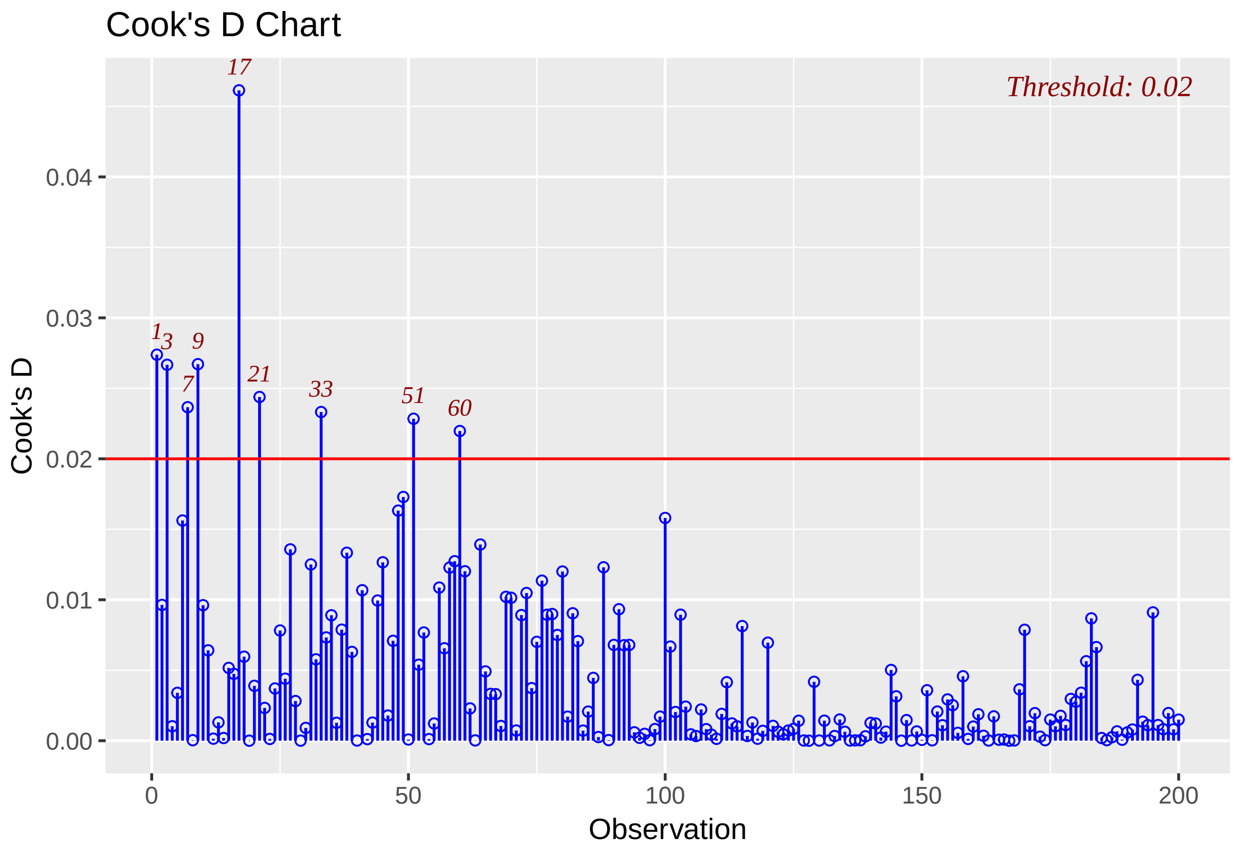
```r
myResPlots(modfin,"Model for Song Popularity")
```

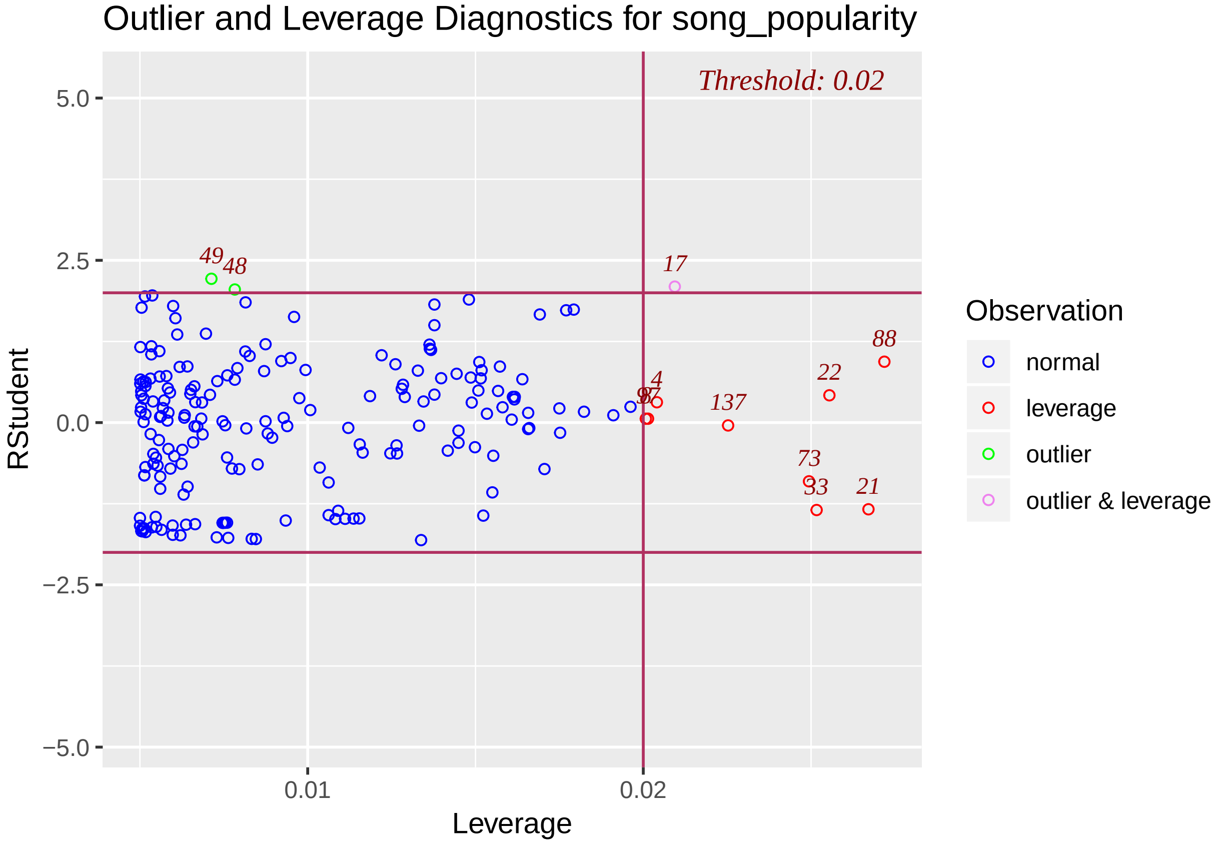


NQ Plot of Studentized Residuals, Model for Song Popularity



Fits vs. Studentized Residuals, Model for Song Popularity


Cook's D Chart
17
1
3
9
7
21
33
51
60
Threshold: 0.02
0.04
0.03
0.02
0.01
0.00
Cook's D
0
50
100
150
200
Observation


Outlier and Leverage Diagnostics for song_popularity
Threshold: 0.02
5.0
2.5
0.0
−2.5
−5.0
RStudent
49
48
17
88
4
97
22
137
73
33
21
0.01
0.02
Leverage
Observation
normal
leverage
outlier
outlier & leverage

Lyric Analysis

Inspired by the analysis conducted by (http://www.sthda.com/english/wiki/text-mining-and-word-cloud-fundamentals-in-r-5-si

```
# Load
library("tm")
library("SnowballC")
library("wordcloud")
```

```
## Loading required package: RColorBrewer
```

```
library("RColorBrewer")

# Remove numbers
docs <- tm_map(docs, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(docs, removeNumbers): transformation drops
## documents
```

```
# Remove english common stopwords
docs <- tm_map(docs, removeWords, stopwords("portuguese"))
```

```
## Warning in tm_map.SimpleCorpus(docs, removeWords, stopwords("portuguese")):
## transformation drops documents
```

```
# Remove punctuations
docs <- tm_map(docs, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(docs, removePunctuation): transformation
## drops documents
```

```
# Eliminate extra white spaces
docs <- tm_map(docs, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(docs, stripWhitespace): transformation drops
## documents
```

```
# Document matrix is a table containing the frequency of the words. Column names are words and row name
dtm_jg <- TermDocumentMatrix(docs[1])
m <- as.matrix(dtm_jg)
v <- sort(rowSums(m),decreasing=TRUE)
d_jg <- data.frame(word = names(v),freq=v)
head(d_jg, 10)
```

```
##            word freq
## nao         nao  268
## voce       voce  238
## amor       amor  146
## vou         vou   78
## bem         bem   73
## sei         sei   68
## quero     quero   63
## tao         tao   55
## coracao coracao   54
## tudo       tudo   54
```

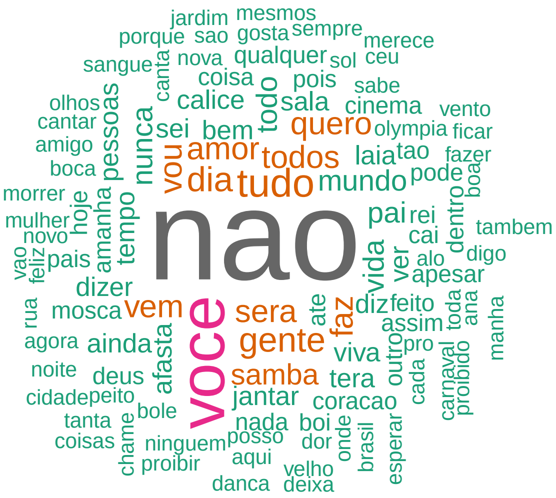
```
dtm_protest <- TermDocumentMatrix(docs[2])
m <- as.matrix(dtm_protest)
v <- sort(rowSums(m),decreasing=TRUE)
```

```
d_protest <- data.frame(word = names(v),freq=v)
head(d_protest, 10)
```

```
##          word freq
## nao       nao  416
## voce     voce  179
## tudo     tudo   94
## gente   gente   74
## dia       dia   71
## sera     sera   64
## todos   todos   64
## amor     amor   60
## faz       faz   59
## quero   quero   58
```

Generate the worcloud for protest songs

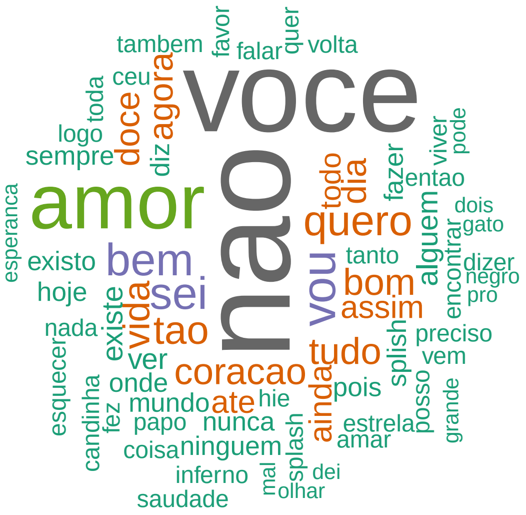
```
set.seed(1234)
wordcloud(words = d_protest$word, freq = d_protest$freq, min.freq = 15,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```



```
#findFreqTerms(dtm, lowfreq = 4)
#findAssocs(dtm, terms = "abusar", corlimit = 1.0)
```

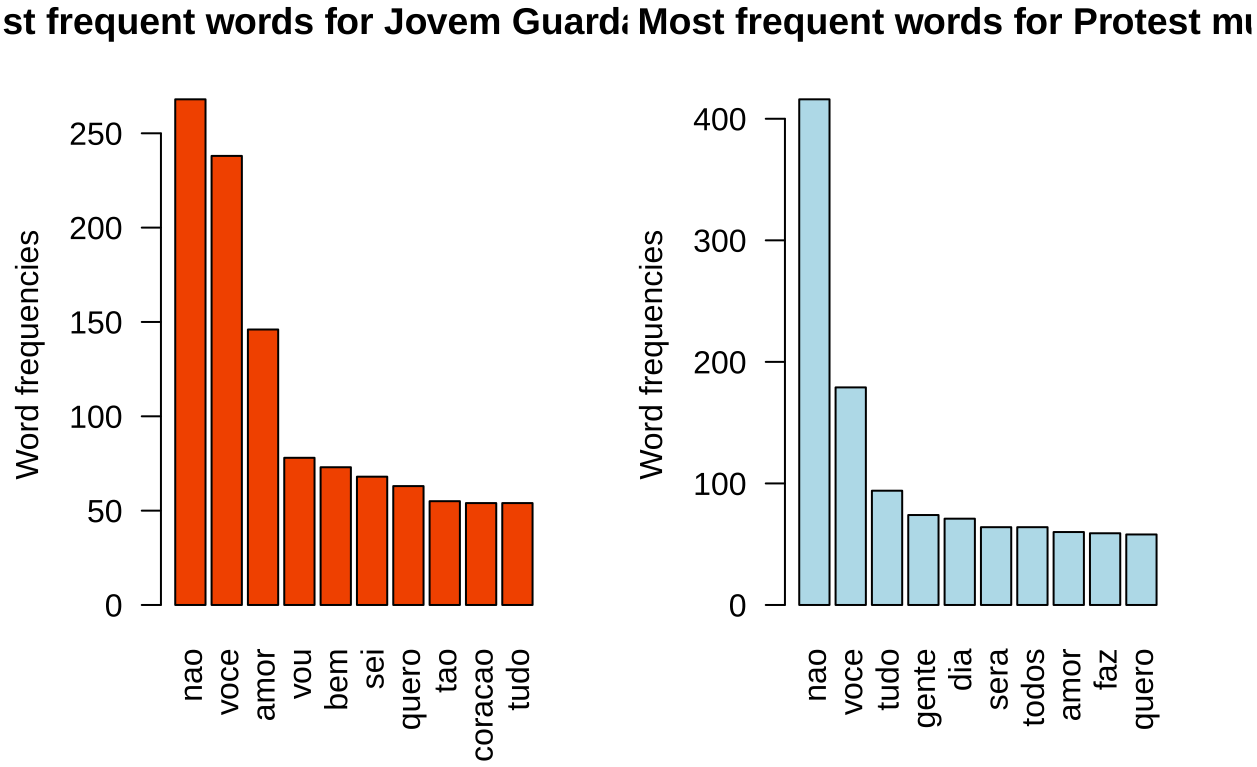
Generate wordclouds for Jovem Guarda

```
wordcloud(words = d_jg$word, freq = d_jg$freq, min.freq = 15,
          max.words=200, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```



Let's analyse now as barplots:

```
par(mfrow=c(1,2))

barplot(d_jg[1:10,]$freq, las = 2, names.arg = d_jg[1:10,]$word,
        col ="orangered2", main ="Most frequent words for Jovem Guarda music",
        ylab = "Word frequencies")

barplot(d_protest[1:10,]$freq, las = 2, names.arg = d_protest[1:10,]$word,
        col ="lightblue", main ="Most frequent words for Protest music",
        ylab = "Word frequencies")
```



Classification