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 ' + 'ORIKjAuJB6gohq-1r-w7FzG7W3FcgsL2ZwSRWjUdLLHOE31lUt6T8otW-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()
```

```
    # 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")
'''
```

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] 34 26
```

```r
str(music)
```

```
## 'data.frame':    34 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  49 57 53 26 44 47 48 22 58 54 ...
##  $ song_lyrics       : chr  "pai , afasta mim cálice pai , afasta mim cálice pai , afasta mim cálic
##  $ 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" "06ND7qqsmIRCuWdQNQIl7
##  $ 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  62 62 62 56 62 63 62 28 68 56 ...
##  $ artist_sp_followers: int  539002 539002 539002 296984 824113 527698 539002 16336 2415626 296985 .
```

Create corpus for text mining

```r
library(tm)
```

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

```r
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)))
inspect(docs)
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 2
##
## [1] terrível bom parar desse jeito provocar sabe onde venho terrível vou dizer ponho pra derreter ra
## [2] pai , afasta mim cálice pai , afasta mim cálice pai , afasta mim cálice vinho tinto sangue pai ,
```

```r
# 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

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  -0.08  0.12    -0.05 -0.28       -0.05
## energy                 -0.08   1.00  0.22     0.61  0.07       -0.07
## key                     0.12   0.22  1.00     0.38 -0.17        0.05
## loudness               -0.05   0.61  0.38     1.00  0.03       -0.01
## mode                   -0.28   0.07 -0.17     0.03  1.00       -0.12
## speechiness            -0.05  -0.07  0.05    -0.01 -0.12        1.00
## acousticness            0.26  -0.54 -0.05    -0.03 -0.21        0.21
```

```
## instrumentalness        -0.25  -0.01 -0.44    -0.24  0.16       -0.14
## liveness                -0.26   0.20  0.27     0.23 -0.19        0.07
## valence                  0.56   0.39  0.21     0.23 -0.20       -0.13
## tempo                   -0.32   0.47  0.20     0.28  0.04       -0.25
##                  acousticness instrumentalness liveness valence tempo
## danceability             0.26            -0.25    -0.26    0.56 -0.32
## energy                  -0.54            -0.01     0.20    0.39  0.47
## key                     -0.05            -0.44     0.27    0.21  0.20
## loudness                -0.03            -0.24     0.23    0.23  0.28
## mode                    -0.21             0.16    -0.19   -0.20  0.04
## speechiness              0.21            -0.14     0.07   -0.13 -0.25
## acousticness             1.00            -0.12    -0.16   -0.03 -0.37
## instrumentalness        -0.12             1.00    -0.19   -0.31 -0.01
## liveness                -0.16            -0.19     1.00   -0.13  0.21
## valence                 -0.03            -0.31    -0.13    1.00  0.16
## tempo                   -0.37            -0.01     0.21    0.16  1.00
```

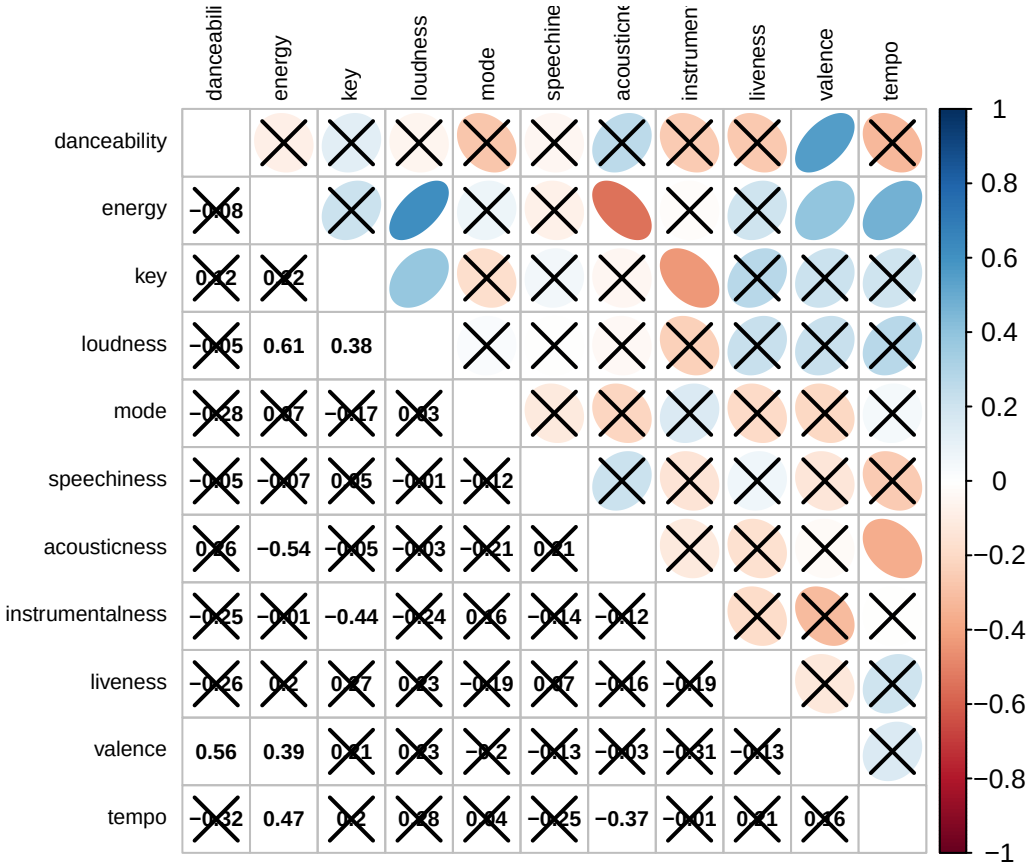
```r
#get the array index (row, col) for the predictors of maximum non-one correlation value
#By ignoring correlations cor1 == 1, you discard the matrix main diagonal
#(correlation of a variable with itself is always 1).
maxloc <- which(cor1 == max(cor1[cor1<1]), arr.ind = TRUE)

#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"
```

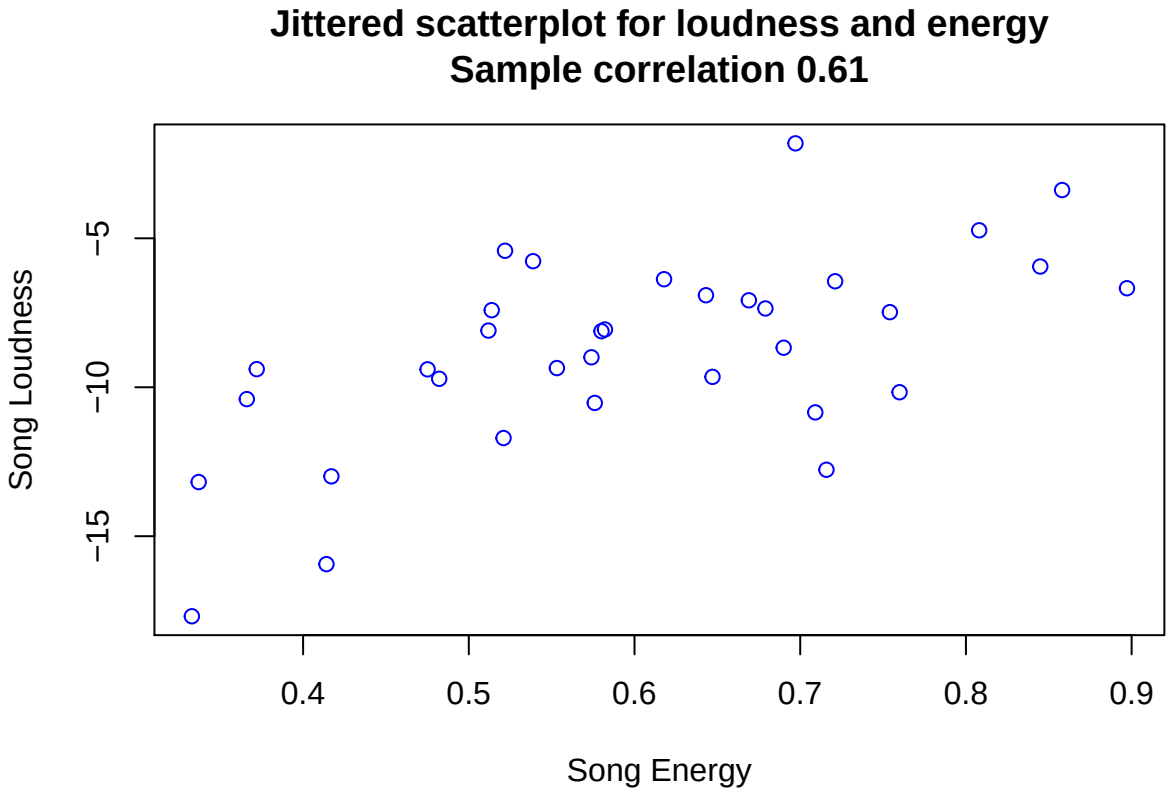
```r
#Create an object called sigcorr that has the results of cor.mtest for columns 10-23 of the crime data.
sigcorr <- cor.mtest(df, conf.level = .95)

#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] 34 12
```

```
str(music2)
```

```
## 'data.frame':    34 obs. of  12 variables:
```

```
##  $ song_popularity : int  49 57 53 26 44 47 48 22 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         TRUE  FALSE FALSE    FALSE FALSE       FALSE
## 2         TRUE         TRUE  FALSE FALSE     TRUE FALSE       FALSE
## 3         TRUE         TRUE   TRUE FALSE     TRUE FALSE       FALSE
## 4         TRUE         TRUE  FALSE FALSE     TRUE  TRUE       FALSE
## 5         TRUE         TRUE  FALSE  TRUE     TRUE  TRUE       FALSE
## 6         TRUE         TRUE   TRUE  TRUE     TRUE  TRUE       FALSE
## 7         TRUE         TRUE   TRUE  TRUE     TRUE  TRUE        TRUE
## 8         TRUE         TRUE   TRUE  TRUE     TRUE  TRUE        TRUE
## 9         TRUE         TRUE   TRUE  TRUE     TRUE  TRUE        TRUE
## 10        TRUE         TRUE   TRUE  TRUE     TRUE  TRUE        TRUE
## 11        TRUE         TRUE   TRUE  TRUE     TRUE  TRUE        TRUE
##    acousticness instrumentalness liveness valence tempo
## 1         FALSE            FALSE    FALSE   FALSE FALSE
## 2         FALSE            FALSE    FALSE   FALSE FALSE
## 3         FALSE            FALSE    FALSE   FALSE FALSE
## 4          TRUE            FALSE    FALSE   FALSE FALSE
## 5          TRUE            FALSE    FALSE   FALSE FALSE
## 6          TRUE            FALSE    FALSE   FALSE FALSE
## 7          TRUE            FALSE    FALSE   FALSE FALSE
## 8          TRUE            FALSE     TRUE   FALSE FALSE
## 9          TRUE             TRUE     TRUE   FALSE FALSE
## 10         TRUE             TRUE     TRUE   FALSE  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
```

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

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

```r
#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
## -23.443  -7.718   1.347   5.868  18.059
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       56.19889   31.60792   1.778   0.0892 .
## danceability     -36.84766   24.02590  -1.534   0.1394
## energy            10.12586   28.67741   0.353   0.7274
## key               -0.35083    0.83263  -0.421   0.6776
## loudness          -0.92685    1.03385  -0.897   0.3797
## mode              -5.24705    5.24838  -1.000   0.3283
## speechiness      -28.02709   71.55473  -0.392   0.6991
## acousticness      -5.19518   10.36378  -0.501   0.6212
## instrumentalness -22.58730   98.21114  -0.230   0.8202
## liveness          -3.72106   12.62196  -0.295   0.7709
## valence            0.83002   14.04925   0.059   0.9534
## tempo             -0.01644    0.10325  -0.159   0.8749
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.28 on 22 degrees of freedom
## Multiple R-squared:  0.2543, Adjusted R-squared:  -0.1186
## F-statistic: 0.682 on 11 and 22 DF,  p-value: 0.7409
```

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

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

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

```r
#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
## -26.276  -4.882   1.617   5.284  19.823
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   33.8213    16.7349   2.021   0.0523 .
## danceability -30.8820    14.1616  -2.181   0.0372 *
## energy        16.6427    15.7850   1.054   0.3001
## loudness      -1.2614     0.7255  -1.739   0.0923 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.85 on 30 degrees of freedom
## Multiple R-squared:  0.2064, Adjusted R-squared:  0.1271
## F-statistic: 2.601 on 3 and 30 DF,  p-value: 0.07036
```

BIC

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

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

```
## [1] "danceability"
```

```
#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
## -27.2087  -4.5914  -0.6727   7.4862  19.6713
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    54.854      8.241   6.656 1.65e-07 ***
## danceability  -30.854     14.334  -2.152    0.039 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.02 on 32 degrees of freedom
## Multiple R-squared:  0.1265, Adjusted R-squared:  0.09917
## F-statistic: 4.633 on 1 and 32 DF,  p-value: 0.03901
```

CP

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

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

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

```
## [1] "danceability"
```

```
#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
## -27.2087  -4.5914  -0.6727   7.4862  19.6713
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    54.854      8.241   6.656 1.65e-07 ***
## danceability  -30.854     14.334  -2.152    0.039 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.02 on 32 degrees of freedom
## Multiple R-squared:  0.1265, Adjusted R-squared:  0.09917
## F-statistic: 4.633 on 1 and 32 DF,  p-value: 0.03901
```

```
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")
```

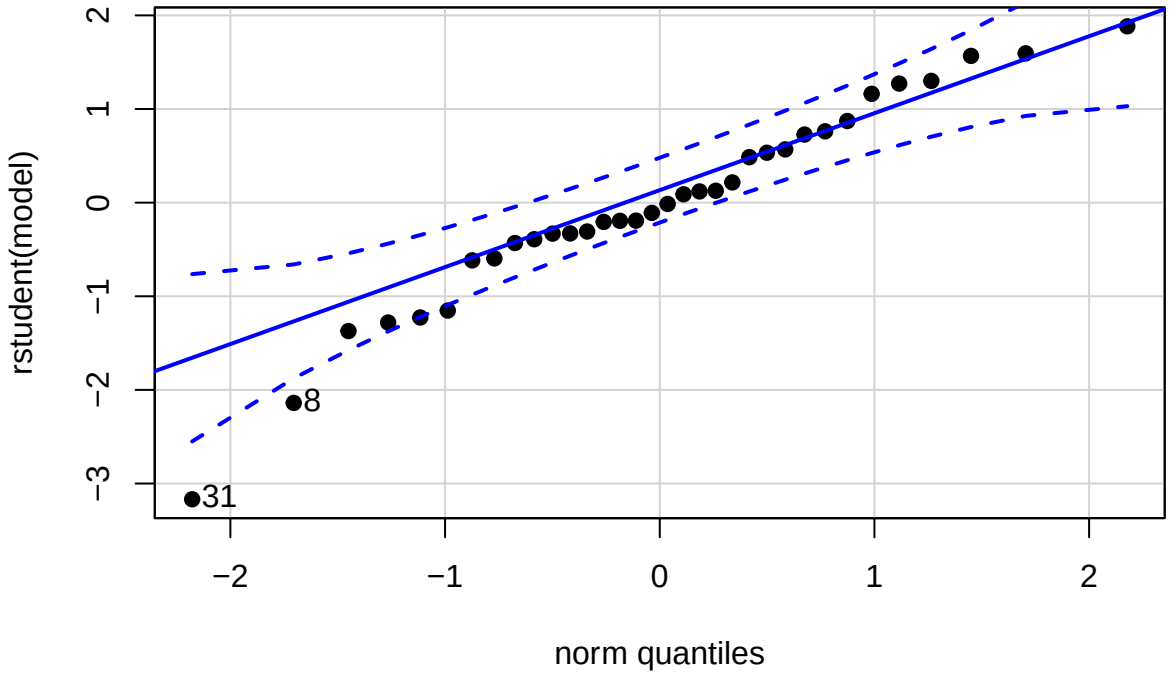
```
##
## Attaching package: 'olsrr'
```

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

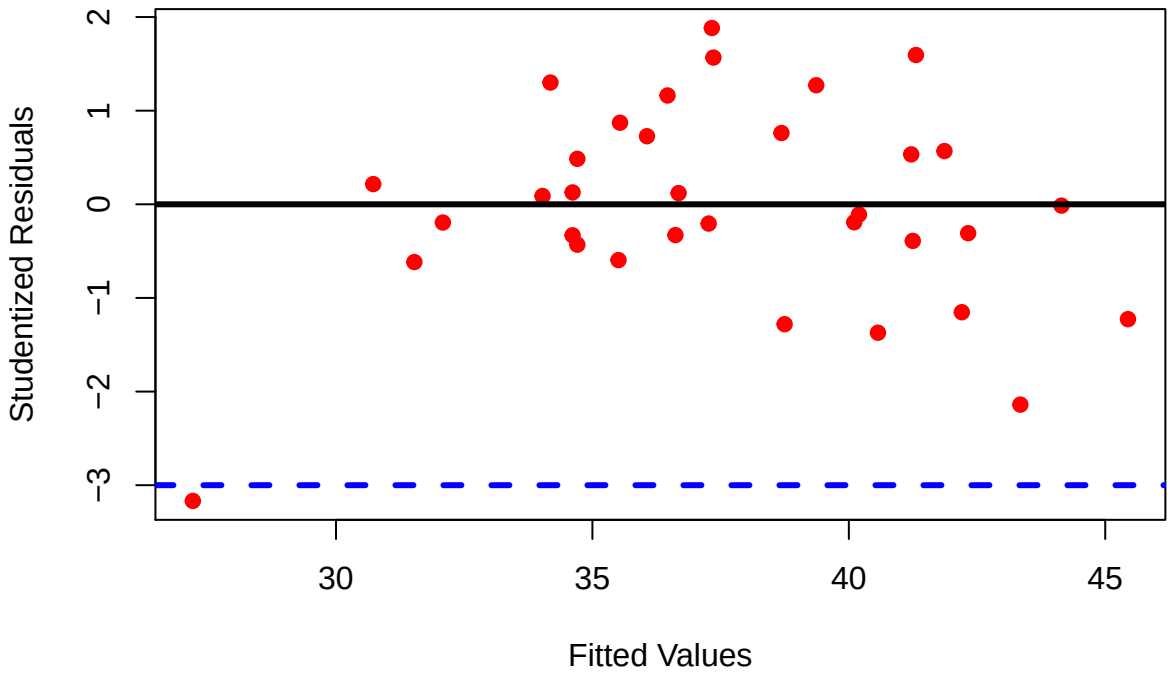
```
## Loading required package: carData
```

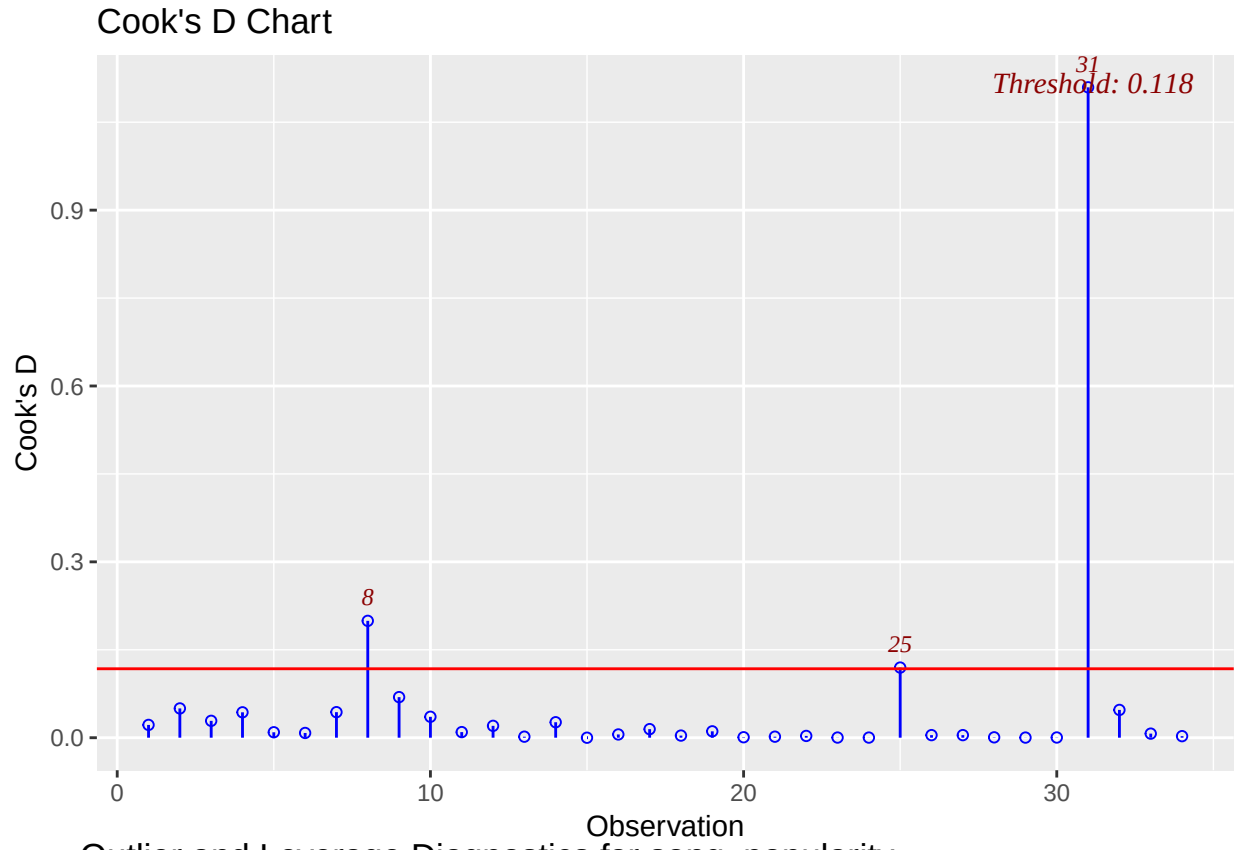
```
myResPlots(modfin,"Model for Song Popularity")
```

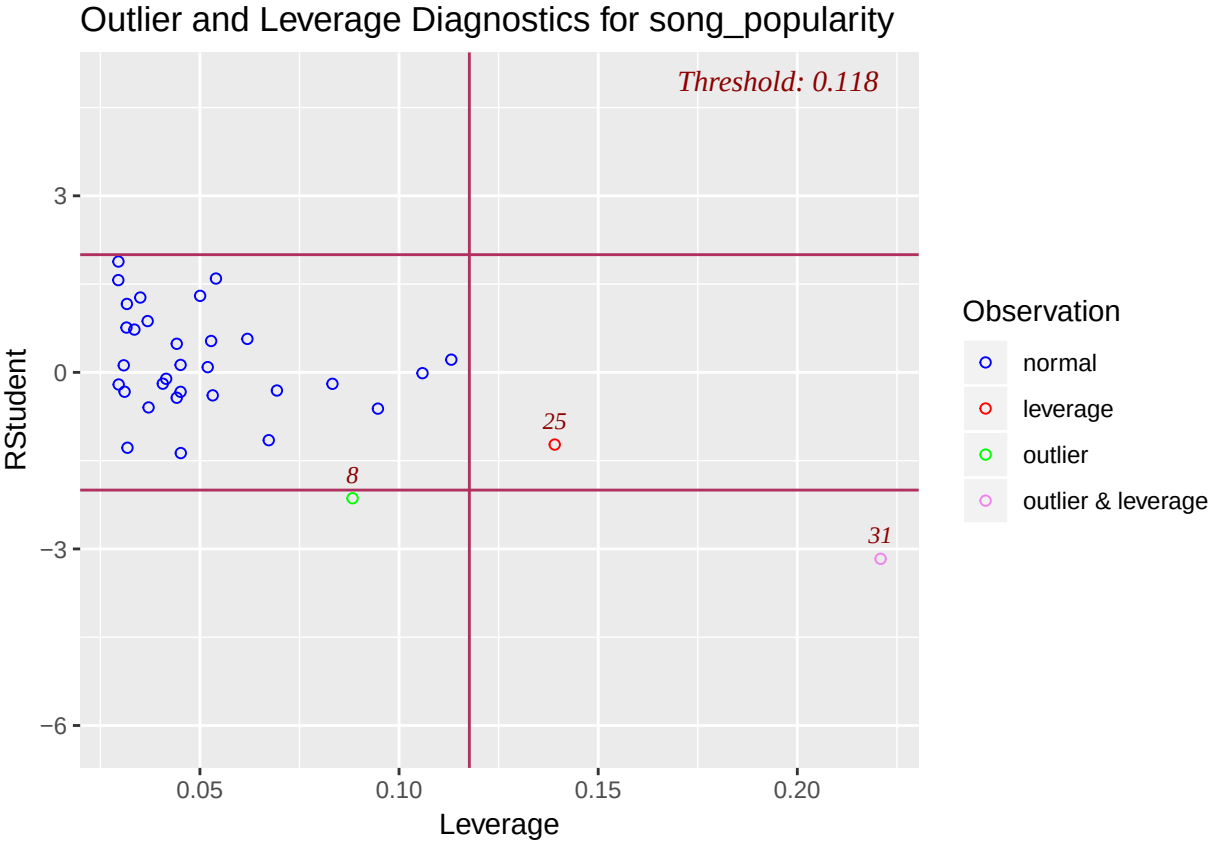
NQ Plot of Studentized Residuals, Model for Song Popularity



Fits vs. Studentized Residuals, Model for Song Popularity




Cook's D Chart
Threshold: 0.118
31
8
25
0.9
0.6
0.3
0.0
Cook's D
0
10
20
30
Observation


Outlier and Leverage Diagnostics for song_popularity
Threshold: 0.118
3
0
−3
−6
RStudent
25
8
31
0.05
0.10
0.15
0.20
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
## amor         amor   15
## coração   coração   15
## papo         papo   13
## vou           vou   13
## existo     existo   12
## terrível terrível   11
## bom           bom   10
## devolva   devolva   10
## vem           vem   10
## bem           bem    8
```

```
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
## mosca     mosca   33
## pai         pai   22
## tudo       tudo   22
## afasta   afasta   21
## cálice   cálice   21
## viva       viva   19
## pessoas pessoas   18
## gosta     gosta   17
## sala       sala   17
## jantar   jantar   16
```

Generate the worcloud for protest songs

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

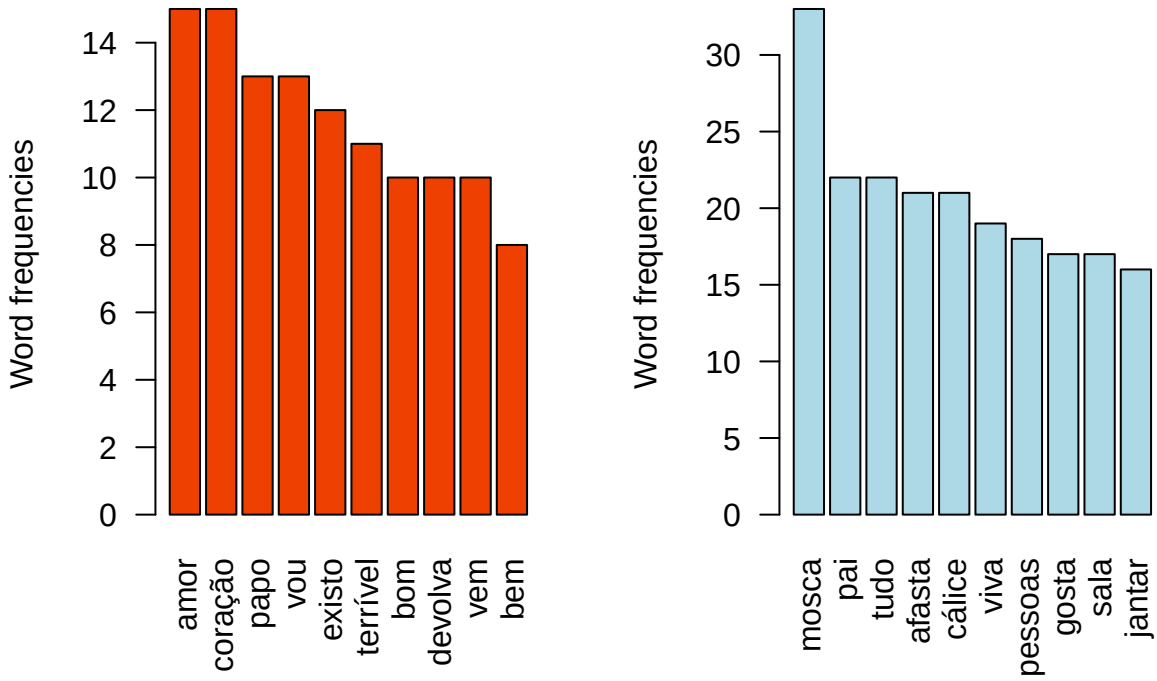
Let's analyse now as barplots:

```
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")
```

st frequent words for Jovem Guard Most frequent words for Protest mu



Classification