Social Media 2023

Faggin Alberto

2023-09-15

Introduction to the problem

Our analysis aims to explore in detail the main characteristics of Italian music through the study of the Sanremo Festival, an event of considerable importance in the national musical panorama. Since its first edition in 1951, the Sanremo Festival has represented a platform for many Italian artists, offering them the opportunity to present their songs to the general public.

Over the years, the Sanremo Festival has welcomed a vast range of musical genres, embracing different artistic currents and sound styles. The wide variety of participants and songs offered made the Festival a true mirror of the evolution of Italian music over time.

Through our research, we intend to trace a path that allows us to understand the evolution of Italian music starting from the Sanremo Festival. We will analyze the songs that managed to rank among the top three in the different editions of the Festival, as these songs have proven to have a significant impact on the public and represent important musical trends of the moment.

We will explore the stylistic and sonic characteristics of these songs, highlighting the distinctive elements that made them memorable and successful. We will analyze the lyrics of the songs and their sentiment, hoping to obtain an exhaustive overview of the musical dynamics of our country and the influence that the Festival has had in promoting and supporting Italian music over the years, also outlining any differences between first, second and third place.

Sentiment analysis

In this section, our aim is to analyze and understand the differences between the top three winners of the Sanremo Festival, through a sentiment analysis. As a first step, we loaded the previously used libraries, which are also essential in this phase to clean the data and create the document term matrix, as done in the previous section. Furthermore, in order to conduct a sentiment analysis, we have integrated the following libraries: TextWiller, which allows us to use the Madda vocabulary to classify words based on their positive or negative valence, and syuzhet, which allows us to obtain a more detailed sentiment classification.

```
library(readtext)
library(knitr)
library(dplyr)
library(lubridate)
library(tidyverse)
library(tidytext)
library(reshape2)
library(tm)
library(topicmodels)
library(stringr)
```

```r
library(SnowballC)
library(LDAvis)
library(tsne)
library(wordcloud)
library(ldatuning)
library(quanteda)
library(syuzhet)
library(TextWiller)
```

Subsequently, we proceeded with the data reading phase. It is important to underline that, in order to facilitate and facilitate the analysis, the song lyrics have been divided into three different folders, one for each position in the chart. This subdivision allowed us to create three distinct corpora relating to the three classified. Initially, we used the "Sys.glob" function to perform an expansion of the file paths using wildcards, thus creating a "character" type object in which each line is associated with the path relative to the individual song within the specific folder (first, second and third). Then, using a "for" loop, we read all the songs. The "text" object was used to hold song lyrics in a "character" format. Finally, using the "corpus" function, we created a corpus object composed of N documents, one for each group. To be specific, we obtained 62 documents for first place, 57 documents for second place, and 47 documents for third place. As previously noted, the analysis was conducted using the lyrics of the 166 songs available on the Genius platform.

```r
setwd("/Users/albertofaggin/Desktop/Master/Second year/Secondo semestre/Social Media/Progetto")
lyrics_1 = Sys.glob(file.path("~/Desktop/Master/Second year/Secondo semestre/Social Media/Progetto/Test



text_1= NULL
for (i in 1:length(lyrics_1)) {
  t=readtext(lyrics_1[i])
  X <- as.character(t[1:nrow(t),])
  text_1 <- c(text_1, X)
}

sep.corpus_1 <- corpus(text_1)

lyrics_2 = Sys.glob(file.path("~/Desktop/Master/Second year/Secondo semestre/Social Media/Progetto/Test


text_2= NULL
for (i in 1:length(lyrics_2)) {
  t=readtext(lyrics_2[i])
  X <- as.character(t[1:nrow(t),])
  text_2 <- c(text_2, X)
}

sep.corpus_2 <- corpus(text_2)

lyrics_3 = Sys.glob(file.path("~/Desktop/Master/Second year/Secondo semestre/Social Media/Progetto/Test


text_3= NULL
for (i in 1:length(lyrics_3)) {
  t=readtext(lyrics_3[i])
```

```r
  X <- as.character(t[1:nrow(t),])
  text_3 <- c(text_3, X)
}

sep.corpus_3 <- corpus(text_3)
```

Next, we used the "tibble" function to create a dataset for each of the three ranking positions, using text_[i] objects, with i = 1,2,3. This process generated three objects in "list" format, in which each line represents the lyrics of each individual song, ordered chronologically starting from the first year of the Festival until today.

```r
text_df_1 <- tibble(line = 1:length(text_1), text = text_1)
text_df_1
```

```
## # A tibble: 62 x 2
##     line text
##    <int> <chr>
##  1     1 "\"Tanti\"\n\"fiori\"\n\"In\"\n\"questo\"\n\"giorno\"\n\"lieto\"\n\"ho~
##  2     2 "\"Dio\"\n\"del\"\n\"Ciel\"\n\"se\"\n\"fossi\"\n\"una\"\n\"colomba\"\n~
##  3     3 "\"Lungo\"\n\"un\"\n\"viale\"\n\"ingiallito\"\n\"d'autunno\"\n\"Triste~
##  4     4 "\"Buongiorno\"\n\"tristezza\"\n\"Io\"\n\"non\"\n\"sapevo\"\n\"lusingh~
##  5     5 "\"La\"\n\"prima\"\n\"rosa\"\n\"rossa\"\n\"è\"\n\"già\"\n\"sbocciata\"~
##  6     6 "\"È\"\n\"tornata\"\n\"L'hanno\"\n\"accolta\"\n\"le\"\n\"stesse\"\n\"c~
##  7     7 "\"Penso\"\n\"che\"\n\"un\"\n\"sogno\"\n\"così\"\n\"non\"\n\"ritorni\"~
##  8     8 "\"Mille\"\n\"violini\"\n\"suonati\"\n\"dal\"\n\"vento\"\n\"Tutti\"\n\~
##  9     9 "\"I\"\n\"miei\"\n\"sorrisi\"\n\"e\"\n\"i\"\n\"tuoi\"\n\"Si\"\n\"sono\~
## 10    10 "\"Sei\"\n\"quasi\"\n\"fatta\"\n\"per\"\n\"me,\"\n\"dipinta\"\n\"per\"~
## # i 52 more rows
```

```r
text_df_2 <- tibble(line = 1:length(text_2), text = text_2)
text_df_2
```

```
## # A tibble: 57 x 2
##     line text
##    <int> <chr>
##  1     1 "\"La\"\n\"luna\"\n\"si\"\n\"veste\"\n\"d'argento\"\n\"Il\"\n\"sole\"\~
##  2     2 "\"Su\"\n\"un\"\n\"campo\"\n\"di\"\n\"grano\"\n\"che\"\n\"dirvi\"\n\"n~
##  3     3 "\"Sul\"\n\"vecchio\"\n\"ponte\"\n\"nella\"\n\"valle\"\n\"aspetto\"\n\~
##  4     4 "\"Mi\"\n\"piace\"\n\"tanto\"\n\"accarezzarti\"\n\"Sugli\"\n\"occhi\"\~
##  5     5 "\"Amami\"\n\"Ti\"\n\"voglio\"\n\"bene\"\n\"!\"\n\"Con\"\n\"24000\"\n\~
##  6     6 "\"Vivevo\"\n\"sola\"\n\"sola\"\n\"In\"\n\"un\"\n\"paese\"\n\"lontano\~
##  7     7 "\"Amor,\"\n\"mon\"\n\"amour,\"\n\"my\"\n\"love\"\n\"(Mon\"\n\"amour)\~
##  8     8 "\"La\"\n\"verità\"\n\"mi\"\n\"fa\"\n\"male,\"\n\"lo\"\n\"so\"\n\"La\"~
##  9     9 "\"C'è\"\n\"una\"\n\"casa\"\n\"bianca\"\n\"che\"\n\"Io\"\n\"non\"\n\"s~
## 10    10 "\"Che\"\n\"cos'è?\"\n\"C'è\"\n\"nell'aria\"\n\"qualcosa\"\n\"di\"\n\"~
## # i 47 more rows
```

```r
text_df_3 <- tibble(line = 1:length(text_3), text = text_3)
text_df_3
```

```
## # A tibble: 47 x 2
```

```
##     line text
##    <int> <chr>
##  1     1 "\"Va\"\n\"serenata\"\n\"mia\"\n\"Stasera\"\n\"ad\"\n\"ascoltare\"\n\"~
##  2     2 "\"Suona\"\n\"La\"\n\"campana\"\n\"sopra\"\n\"la\"\n\"collina\"\n\"Là,~
##  3     3 "\"Lassù\"\n\"in\"\n\"un\"\n\"ripostiglio\"\n\"polveroso\"\n\"Tra\"\n\~
##  4     4 "\"Vieni\"\n\"a\"\n\"vedere\"\n\"il\"\n\"mio\"\n\"mare\"\n\"Io\"\n\"lo~
##  5     5 "\"Bella\"\n\"come\"\n\"nulla\"\n\"al\"\n\"mondo\"\n\"eri\"\n\"per\"\n~
##  6     6 "\"Un\"\n\"volo\"\n\"di\"\n\"gabbiani\"\n\"telecomandati\"\n\"E\"\n\"u~
##  7     7 "\"Dice\"\n\"che\"\n\"era\"\n\"un\"\n\"bell'uomo\"\n\"e\"\n\"veniva,\"~
##  8     8 "\"Non\"\n\"cerco\"\n\"un\"\n\"re\"\n\"di\"\n\"denari\"\n\"Io\"\n\"cer~
##  9     9 "\"Da\"\n\"troppo\"\n\"tempo\"\n\"mi\"\n\"trascuro,\"\n\"questo\"\n\"s~
## 10    10 "\"Piove\"\n\"da\"\n\"qualche\"\n\"minuto\"\n\"Ti\"\n\"guardo\"\n\"e\"~
## # i 37 more rows
```

After creating the datasets, a cleaning process had to be performed in order to remove elements that were not necessary for the analysis. Initially, we eliminated apostrophes and punctuation marks such as "?", ",", "!", "“’ and " " using the "str_replace_all" function, which replaces all matches defined in the function. Since the "text_df_" objects are structured with as many rows as there are testi in the first three positions and with two columns, one for the document count and the other for the song lyrics, we subsequently divided the second column into smaller units called "tokens". We used the "unnest_tokens" function to carry out this subdivision, in this case considering the tokens as single words. It is important to note that a token could also represent a phrase, a character, or a sequence of characters, but in our case we considered words as tokens. To further improve the quality of the analysis, we applied the "get_stopwords" function to remove so-called stop words. Stop words are common and frequent words such as articles (for example, "il", "la", "un"), prepositions (for example, "di", "a", "da"), conjunctions (for example, "and", "but", "or") and pronouns (for example, "I", "you", "it") that usually do not contribute relevant information in the context of a text. Furthermore, we also eliminated words such as "chorus", "verse" and "lyrics" as they were present in the testi downloaded from the Genius platform, but had no specific meaning for the analysis conducted. Furthermore, we have removed words with a length of less than 3 characters, as they are usually considered to be of little significance due to their brevity. These additional cleaning steps were performed to ensure greater accuracy and consistency in data processing.

```r
# Correction of apostrophes
text_df_1 <- text_df_1 %>%
  mutate(text = str_replace_all(text_1, "[\'’](?!\\s)", "' "))

testi_all_1 <- text_df_1 %>%
  unnest_tokens(word, text)

# I eliminate stopwords
testi_all_1 <- testi_all_1 %>%
  anti_join(get_stopwords(language = "it"))

# I delete the words chorus and verse
testi_all_1 <- testi_all_1 %>%
   filter(word!="chorus"&word!="verse"&word!="lyrics"&word!="You" &word!="might"&word!="also"
          &word!="likeEmbed"&word!="likeembed"&word!="like")

# We eliminate "words" 1, 2 char long
testi_all_1 <- testi_all_1 %>%
   filter(nchar(word)>3)

# Runners-up
text_df_2 <- text_df_2 %>%
```

```r
   mutate(text = str_replace_all(text_2, "[\'’](?!\\s)", "' "))
testi_all_2 <- text_df_2 %>%
   unnest_tokens(word, text)
testi_all_2 <- testi_all_2 %>%
   anti_join(get_stopwords(language = "it"))
testi_all_2 <- testi_all_2 %>%
   filter(word!="chorus"&word!="verse"&word!="lyrics"&word!="You" &word!="might"&word!="also"
          &word!="likeEmbed"&word!="likeembed"&word!="like")
testi_all_2 <- testi_all_2 %>%
   filter(nchar(word)>3)

# Third place
text_df_3 <- text_df_3 %>%
   mutate(text = str_replace_all(text_3, "[\'’](?!\\s)", "' "))
testi_all_3 <- text_df_3 %>%
   unnest_tokens(word, text)
testi_all_3 <- testi_all_3 %>%
   anti_join(get_stopwords(language = "it"))
testi_all_3 <- testi_all_3 %>%
   filter(word!="chorus"&word!="verse"&word!="lyrics"&word!="You" &word!="might"&word!="also"
          &word!="likeEmbed"&word!="likeembed"&word!="like")
testi_all_3 <- testi_all_3 %>%
   filter(nchar(word)>3)
```

Once the data had been created and cleaned, we proceeded with a preliminary analysis in order to identify the words most used by singers in the 73 editions of the Sanremo Festival. We took two approaches: using a wordcloud and calculating the ten most frequent words. The wordcloud was generated to provide a visual representation of the most frequently occurring words. Using this technique, the size of words in the wordcloud is proportional to their frequency in the corpus of song lyrics. This way, you can get a visual snapshot of the words that appear most frequently in the dataset. Furthermore, we performed the calculation of the ten most frequent words. This allowed us to identify more specifically the words that were used most frequently by singers during the 73 editions of the Festival. This approach provided us with a list of the most common words, allowing us to gain an overview of the predominant themes or terms in the participating songs.

```r
# We calculate the frequencies of each word per document
testi_all_1 = testi_all_1 %>% group_by(line,word) %>% dplyr::mutate(freq = n()) %>% unique()

# Calculate overall frequencies to clean rare words
testi_all_1 = testi_all_1 %>% ungroup() %>% group_by(word) %>% dplyr::mutate(tot=n())

# top 25 words by frequency
testi_all_1 %>% group_by(word) %>% dplyr::summarise(tot=n()) %>% arrange(desc(tot)) %>% print(n=10)
```

```
## # A tibble: 2,004 x 2
##    word     tot
##    <chr>  <int>
##  1 amore     36
##  2 sempre    26
##  3 vita      24
##  4 occhi     23
##  5 solo      23
##  6 senza     22
```

```
##   7 bene       21
##   8 quando     19
##   9 così       18
##  10 dentro     18
## # i 1,994 more rows
```

```r
# I prepare a structure for tagcloud
tag_words_1 = testi_all_1 %>% select(word,tot) %>% unique()

# Runners-up
testi_all_2 = testi_all_2 %>% group_by(line,word) %>% dplyr::mutate(freq = n()) %>% unique()
testi_all_2 = testi_all_2 %>% ungroup() %>% group_by(word) %>% dplyr::mutate(tot=n())
testi_all_2 %>% group_by(word) %>% dplyr::summarise(tot=n()) %>% arrange(desc(tot)) %>% print(n=10)
```

```
## # A tibble: 1,989 x 2
##     word      tot
##     <chr>   <int>
##   1 amore      30
##   2 cuore      26
##   3 quando     24
##   4 vita       22
##   5 solo       21
##   6 sempre     20
##   7 cosa       19
##   8 ancora     16
##   9 notte      16
##  10 senza      16
## # i 1,979 more rows
```

```r
tag_words_2 = testi_all_2 %>% select(word,tot) %>% unique()

# Third place
testi_all_3 = testi_all_3 %>% group_by(line,word) %>% dplyr::mutate(freq = n()) %>% unique()
testi_all_3 = testi_all_3 %>% ungroup() %>% group_by(word) %>% dplyr::mutate(tot=n())
testi_all_3 %>% group_by(word) %>% dplyr::summarise(tot=n()) %>% arrange(desc(tot)) %>% print(n=10)
```

```
## # A tibble: 1,577 x 2
##     word      tot
##     <chr>   <int>
##   1 amore      27
##   2 cuore      20
##   3 senza      20
##   4 solo       19
##   5 dentro     18
##   6 ancora     16
##   7 mondo      15
##   8 sempre     15
##   9 ogni       14
##  10 tempo      14
## # i 1,567 more rows
```

```r
tag_words_3 = testi_all_3 %>% select(word,tot) %>% unique()
```

```r
# Word cloud plot
set.seed(1234)
ccol = RColorBrewer::brewer.pal(8,"Dark2")
wordcloud(words = tag_words_1$word,freq = tag_words_1$tot,min.freq = 5,colors=ccol)
```


```r
set.seed(124)
wordcloud(words = tag_words_2$word,freq = tag_words_2$tot,min.freq = 5,colors=ccol)
```

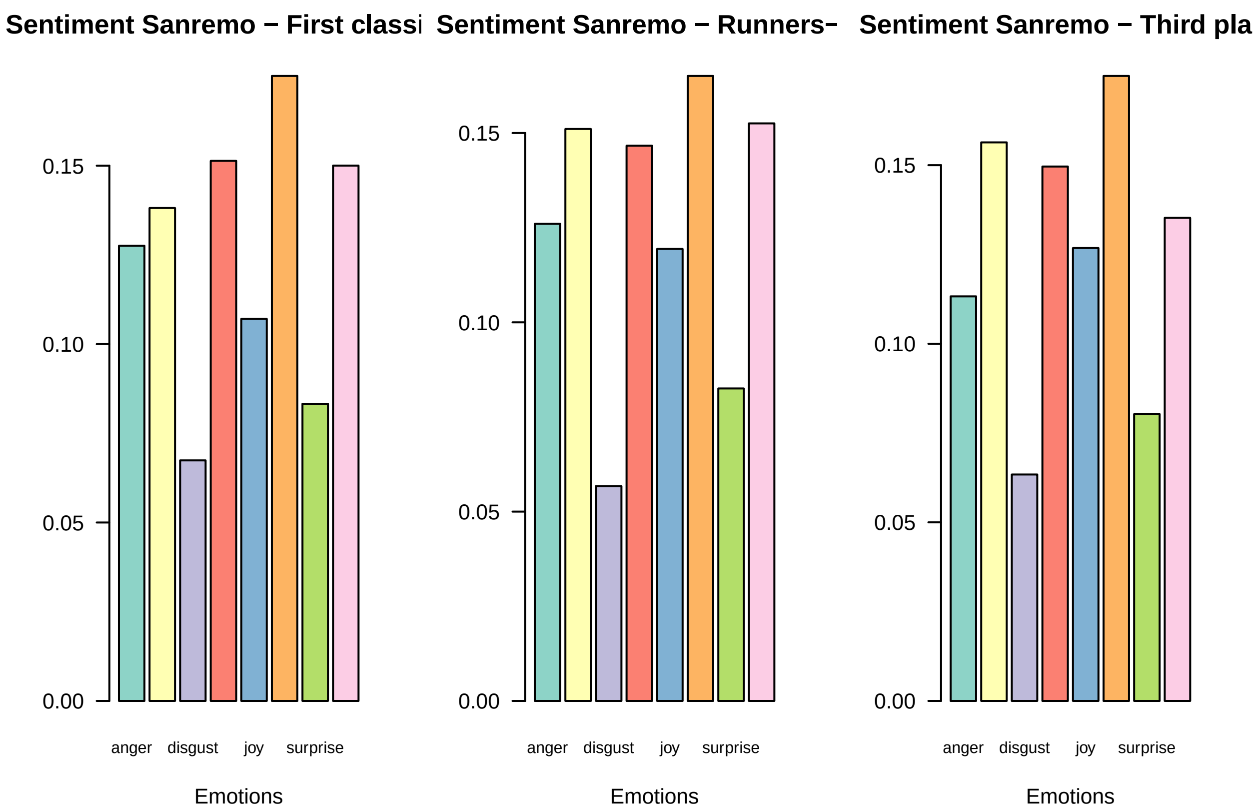

```r
set.seed(123)
wordcloud(words = tag_words_3$word,freq = tag_words_3$tot,min.freq = 5,colors=ccol)
```

It is evident, both from the previous section and from this analysis, how the predominant theme of love clearly emerges, a key word in the songs of the Sanremo Festival. Before proceeding with an in-depth analysis of the feeling associated with the word "love", it is interesting to observe the general feeling that characterizes the first three positions in the Sanremo Festival rankings. First of all, the "syuzhet" library was used in order to obtain a definition of the sentiment associated with each word via the "get_nrc_sentiment" command. In particular, three datasets called "dati_[i]" were created, corresponding respectively to the first three positions in the ranking. These datasets were generated by selecting exclusively the column of words from the "testi_all_[i]" dataframe.

```
dati_1 = testi_all_1[,2]
dati_1 = data.frame(dati_1)
sentiment_scores_1 <- get_nrc_sentiment(dati_1[,1], lang="italian")

dati_2 = testi_all_2[,2]
dati_2 = data.frame(dati_2)
sentiment_scores_2 <- get_nrc_sentiment(dati_2[,1], lang="italian")

dati_3 = testi_all_3[,2]
dati_3 = data.frame(dati_3)
sentiment_scores_3 <- get_nrc_sentiment(dati_3[,1], lang="italian")
```

Subsequently, it was possible to graphically represent the sentiment present in the three different positions in the ranking. The peculiarity of the "get_nrc_sentiment" function of the "syuzhet" library consists in the more accurate classification of the feelings of words, dividing them into the following categories: anger, anticipation, disgust, fear, joy, sadness, surprise and trust.

```r
par(mfrow=c(1,3))
barplot(
   colSums(prop.table(sentiment_scores_1[, 1:8])),
   space = 0.2,
   horiz = FALSE,
   las = 1,
   cex.names = 0.7,
   col = brewer.pal(n = 8, name = "Set3"),
   main = "Sentiment Sanremo - First classified",
   xlab="Emotions", ylab = NULL)

barplot(
   colSums(prop.table(sentiment_scores_2[, 1:8])),
   space = 0.2,
   horiz = FALSE,
   las = 1,
   cex.names = 0.7,
   col = brewer.pal(n = 8, name = "Set3"),
   main = "Sentiment Sanremo - Runners-up",
   xlab="Emotions", ylab = NULL)

barplot(
   colSums(prop.table(sentiment_scores_3[, 1:8])),
   space = 0.2,
   horiz = FALSE,
   las = 1,
   cex.names = 0.7,
   col = brewer.pal(n = 8, name = "Set3"),
   main = "Sentiment Sanremo - Third place",
   xlab="Emotions", ylab = NULL)
```




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

From the graphic analysis it is possible to notice that sadness, anger and fear are the elements sought by Italians when they wish to place a song at the top of the charts. Given the aforementioned peculiarities, we undertook an in-depth analysis of the feelings expressed by the singers, focusing on the detailed research of the words that distinguish and characterize these feelings. For this purpose, we used wordclouds for more detailed observation. The first aspect we focused our attention on was the feeling of sadness. To achieve this objective, we initially saved in an object called "sad_word_[i]" the words which, within the three different "dati_[i]" dataframes, present a sad sentiment, easily obtainable through the "sentiment_scores_[i] command]$sadness". Next, we examined the sad feeling words for the three positions using the command: "head(sad_word_order_1, n = 12)".

```r
sad_words_1 <- dati_1[sentiment_scores_1$sadness> 0,]
sad_word_order_1 <- sort(table(unlist(sad_words_1)), decreasing = TRUE)

head(sad_word_order_1, n = 12)
```

```
##
##        male       fondo       colpa      dolore       bello      morire      musica
##          13           9           7           7           6           6           5
##   piangere        buio       corsa     deserto malinconia
##           5           3           3           3           3
```

```r
sad_words_2 <- dati_2[sentiment_scores_2$sadness> 0,]
sad_word_order_2 <- sort(table(unlist(sad_words_2)), decreasing = TRUE)

head(sad_word_order_2, n = 12)
```

```
##
##       male       buio     dolore     stanco    cantare     morire      perso
##         11          6          6          5          4          4          4
##      bello      fondo      madre malinconia     musica
##          3          3          3          3          3
```

```r
sad_words_3 <- dati_3[sentiment_scores_3$sadness> 0,]
sad_word_order_3 <- sort(table(unlist(sad_words_3)), decreasing = TRUE)

head(sad_word_order_3, n = 12)
```

```
##
##       male     dolore      fondo    cantare     morire     musica      bello
##         10          5          5          4          4          4          3
##      bugia cancellare      colpa     guerra      madre
##          3          3          3          3          3
```

In this case it is possible to observe how the words associated with a sad feeling are: “bad”, “pain”, “bottom” and “die”. Furthermore, for a simpler visual representation, we built 3 different dataframes called sad_dati_[i], which contained the frequency and words to which the feeling relating to sadness was associated, in order to also show the wordclouds for the three groups .

```r
sad_dati_1 = sad_word_order_1
sad_dati_1 = data.frame(sad_dati_1)

sad_dati_2 = sad_word_order_2
sad_dati_2 = data.frame(sad_dati_2)

sad_dati_3 = sad_word_order_3
sad_dati_3 = data.frame(sad_dati_3)
par(mfrow=c(1,3))
wordcloud(words = sad_dati_1$Var1, freq = sad_dati_1$Freq,min.freq = 3,
          color = brewer.pal(4,"Dark2"))

wordcloud(words = sad_dati_2$Var1, freq = sad_dati_2$Freq,min.freq = 3,
          color = brewer.pal(4,"Dark2"))

wordcloud(words = sad_dati_3$Var1, freq = sad_dati_3$Freq,min.freq = 3,
          color = brewer.pal(4,"Dark2"))
```

Subsequently we focused our attention on the feeling of anger. The procedure was similar to the one carried out previously.

```r
anger_words_1 <- dati_1[sentiment_scores_1$anger > 0,]
anger_word_order_1 <- sort(table(unlist(anger_words_1)), decreasing = TRUE)

head(anger_word_order_1, n = 12)
```

```
##
##     male   parole    paura    colpa    forza     buio  deserto    bugia
##       13       13        8        7        6        3        3        2
##    colpo    folla   guerra infinito
##        2        2        2        2
```

```r
anger_words_2 <- dati_2[sentiment_scores_2$anger > 0,]
anger_word_order_2 <- sort(table(unlist(anger_words_2)), decreasing = TRUE)

head(anger_word_order_2, n = 12)
```

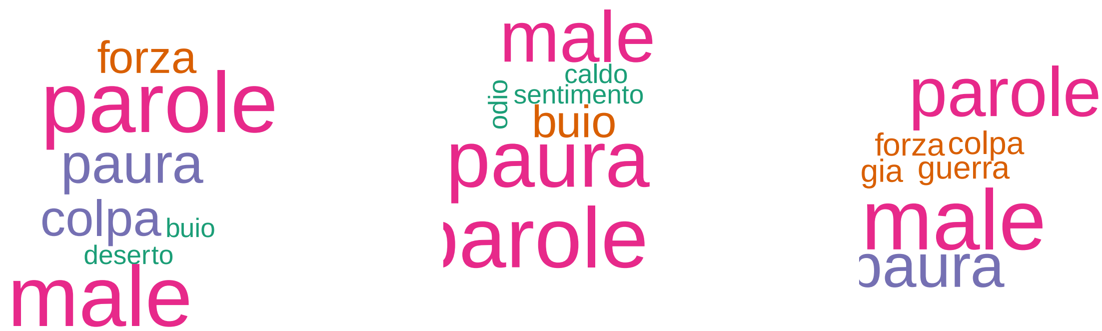
```
##
##     parole      paura       male       buio      caldo       odio sentimento
##         13         12         11          6          3          3          3
##  abbandono  argomento    cattivo      colpo      danno
##          2          2          2          2          2
```

```r
anger_words_3 <- dati_3[sentiment_scores_3$anger > 0,]
anger_word_order_3 <- sort(table(unlist(anger_words_3)), decreasing = TRUE)

head(anger_word_order_3, n = 12)
```

```
##
##     male   parole    paura    bugia    colpa    forza   guerra    canto
##       10        8        7        3        3        3        3        2
##  deserto incontro infinito  perdere
##        2        2        2        2
```

In this case it is possible to observe how the words associated with a sad feeling are: "evil", "fear", "dark" and "guilt". In order to obtain a more intuitive visual representation, also in this case, we proceeded with the creation of three distinct dataframes called "anger_dati_[i]". These dataframes contain the frequencies of words associated with the feeling of anger, thus allowing the generation of wordclouds.

```r
anger_dati_1 = anger_word_order_1
anger_dati_1 = data.frame(anger_dati_1)

anger_dati_2 = anger_word_order_2
anger_dati_2 = data.frame(anger_dati_2)

anger_dati_3 = anger_word_order_3
anger_dati_3 = data.frame(anger_dati_3)

par(mfrow=c(1,3))
wordcloud(words = anger_dati_1$Var1, freq = anger_dati_1$Freq,min.freq = 3,
          color = brewer.pal(4,"Dark2"))

wordcloud(words = anger_dati_2$Var1, freq = anger_dati_2$Freq,min.freq = 3,
          color = brewer.pal(4,"Dark2"))

wordcloud(words = anger_dati_3$Var1, freq = anger_dati_3$Freq,min.freq = 3,
          color = brewer.pal(4,"Dark2"))
```



Finally, we focused our attention on the feeling of joy. Also in this case the procedure was similar to the one carried out previously.

```
joy_words_1 <- dati_1[sentiment_scores_1$joy > 0,]
joy_word_order_1 <- sort(table(unlist(joy_words_1)), decreasing = TRUE)

head(joy_word_order_1, n = 12)
```

```
##
##      sole     dolce      vero     bello     bacio    musica orgoglio   sorriso
##        18         7         7         6         5         5         5         5
## abbraccio     bella    felice      vivo
##         4         4         4         4
```

```
joy_words_2 <- dati_2[sentiment_scores_2$joy > 0,]
joy_word_order_2 <- sort(table(unlist(joy_words_2)), decreasing = TRUE)

head(joy_word_order_2, n = 12)
```

```
##
##       sole       bella       bacio       dolce     cantare     sorriso       bello
##         10           6           5           5           4           4           3
##      madre      musica        pace sentimento      sicuro
##          3           3           3           3           3
```

```r
joy_words_3 <- dati_3[sentiment_scores_3$joy > 0,]
joy_word_order_3 <- sort(table(unlist(joy_words_3)), decreasing = TRUE)

head(joy_word_order_3, n = 12)
```

```
##
##     sole    dolce  sorriso    bella  cantare   musica     vero     vivo
##        8        5        5        4        4        4        4        4
##  bambino    bello     caro crescere
##        3        3        3        3
```

In this case it is possible to observe how the words associated with a sad feeling are: “sun”, “sweet”, “smile” and “beautiful”. Finally, also in this case to obtain a more intuitive visual representation, we created three distinct dataframes called “joy__dati__[i]”. These dataframes contain the frequencies of words associated with the feeling of joy, thus allowing the generation of wordclouds.

```r
joy_dati_1 = joy_word_order_1
joy_dati_1 = data.frame(joy_dati_1)

joy_dati_2 = joy_word_order_2
joy_dati_2 = data.frame(joy_dati_2)

anger_dati_3 = anger_word_order_3
anger_dati_3 = data.frame(anger_dati_3)

par(mfrow=c(1,3))
wordcloud(words = joy_dati_1$Var1, freq = joy_dati_1$Freq,min.freq = 3,
          color = brewer.pal(4,"Dark2"))

wordcloud(words = joy_dati_2$Var1, freq = joy_dati_2$Freq,min.freq = 3,
          color = brewer.pal(4,"Dark2"))

wordcloud(words = anger_dati_3$Var1, freq = anger_dati_3$Freq,min.freq = 3,
          color = brewer.pal(4,"Dark2"))
```

In addition, in order to provide a concise representation of the sentiment of the top three ranked, we generated a wordcloud that summarizes the most frequent words associated with the feelings of joy, happiness, sadness and anger. This approach allows for a compact and comprehensive visualization of such feelings. To achieve this goal, we initially created an object called "cloud_emotions_data_[i]", which included all the words related to the four sentiments considered. Next, in order to create a document-term matrix, we used the "VectorSource" and "Corpus" functions to obtain the "cloud_corpus_[i]" object. This corpus is made up of four documents, each associated with a certain feeling. Finally, by using the "TermDocumentMatrix" function, we were able to construct the document-term matrix, where each row represents a word and each column corresponds to a sentiment. The value present in each cell indicates the frequency with which a given word appears within the four previously defined documents. It is important to note that a word can be associated with two sentiments at the same time. For example, the word "evil" can be associated with both the feeling of anger and fear.

```
cloud_emotions_data_1 <- c(
  paste(dati_1[sentiment_scores_1$sadness> 0,], collapse = " "),
  paste(dati_1[sentiment_scores_1$joy > 0,], collapse = " "),
  paste(dati_1[sentiment_scores_1$anger > 0,], collapse = " "),
  paste(dati_1[sentiment_scores_1$fear > 0,], collapse = " "))


cloud_corpus_1 <- Corpus(VectorSource(cloud_emotions_data_1))
cloud_tdm_1 <- TermDocumentMatrix(cloud_corpus_1)
cloud_tdm_1 <- as.matrix(cloud_tdm_1)
head(cloud_tdm_1, 10)
```

```
##              Docs
```

```
## Terms         1 2 3 4
##   abbandonare 1 0 0 0
##   bastardo    1 0 0 0
##   battito     1 0 1 1
##   bello       6 6 0 0
##   bomba       1 0 1 1
##   brutto      1 0 1 1
##   bugia       2 0 2 0
##   buio        3 0 3 3
##   cadere      1 0 0 0
##   cantare     2 2 0 0
```

```r
cloud_emotions_data_2 <- c(
  paste(dati_2[sentiment_scores_2$sadness> 0,], collapse = " "),
  paste(dati_2[sentiment_scores_2$joy > 0,], collapse = " "),
  paste(dati_2[sentiment_scores_2$anger > 0,], collapse = " "),
  paste(dati_2[sentiment_scores_2$fear > 0,], collapse = " "))
cloud_corpus_2 <- Corpus(VectorSource(cloud_emotions_data_2))
cloud_tdm_2 <- TermDocumentMatrix(cloud_corpus_2)
cloud_tdm_2 <- as.matrix(cloud_tdm_2)

head(cloud_tdm_2, 10)
```

```
##              Docs
## Terms         1 2 3 4
##   abbandonare 1 0 0 0
##   abbandono   2 0 2 2
##   amarezza    1 0 1 0
##   arte        1 1 0 0
##   bello       3 3 0 0
##   blues       1 0 0 1
##   bugia       1 0 1 0
##   buio        6 0 6 6
##   cadere      1 0 0 0
##   cancellare  1 0 0 1
```

```r
cloud_emotions_data_3 <- c(
  paste(dati_3[sentiment_scores_3$sadness> 0,], collapse = " "),
  paste(dati_3[sentiment_scores_3$joy > 0,], collapse = " "),
  paste(dati_3[sentiment_scores_3$anger > 0,], collapse = " "),
  paste(dati_3[sentiment_scores_3$fear > 0,], collapse = " "))
cloud_corpus_3 <- Corpus(VectorSource(cloud_emotions_data_3))
cloud_tdm_3 <- TermDocumentMatrix(cloud_corpus_3)
cloud_tdm_3 <- as.matrix(cloud_tdm_3)

head(cloud_tdm_3, 10)
```

```
##               Docs
## Terms          1 2 3 4
##   abbandono    1 0 1 1
##   agonia       1 0 1 1
##   assente      1 0 0 0
##   attraversare 1 0 1 1
```

```
##    bello        3 3 0 0
##    bugia        3 0 3 0
##    buia         1 0 0 1
##    buio         1 0 1 1
##    cadere       1 0 0 0
##    cancellare   3 0 0 3
```

At this point, thanks to the “comparison.cloud” function, it was possible to represent the wordcloud of the first 3 positions for the four feelings.

```r
set.seed(757) # this can be set to any integer
par(mfrow = c(1,3))
comparison.cloud(cloud_tdm_1, random.order = FALSE,
                 colors = c("green", "red", "orange", "blue"),
                 title.size = 1, max.words = 50, scale = c(2.5, 1), rot.per = 0.4)
comparison.cloud(cloud_tdm_2, random.order = FALSE,
                 colors = c("green", "red", "orange", "blue"),
                 title.size = 1, max.words = 50, scale = c(2.5, 1), rot.per = 0.4)
comparison.cloud(cloud_tdm_3, random.order = FALSE,
                 colors = c("green", "red", "orange", "blue"),
                 title.size = 1, max.words = 50, scale = c(2.5, 1), rot.per = 0.4)
```


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

Finally, considering the significant importance of the word “love”, we decided to conduct an in-depth analysis of the sentiment associated with the words most closely linked to this broad concept, in order to identify the main differences on this theme in the various winners of the Festival. To achieve this objective, we initially built the document term matrix starting from the testi_all_[i] dataframe, subsequently we exploited the "findAssocs" function, which allows us to identify the associations in the document-term matrix relating to a specific word, in our case "Love". Furthermore, we specified to only consider words with an association greater than 25%. Subsequently, we built a dataframe containing the words most associated with the term "love" and their respective frequencies.

```
dtm_1 = testi_all_1 %>% cast_dtm(document = line,term = word,value = freq)
inspect(dtm_1)
```

```
## <<DocumentTermMatrix (documents: 62, terms: 2004)>>
## Non-/sparse entries: 3621/120627
## Sparsity           : 97%
## Maximal term length: 15
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs amore ancora cuore occhi quando sempre senza solo vita vorrei
##   45     4      0     0     0      4      0     0    1    0      2
##   46     0      4     0     1      3      1     1    2    0      0
##   49     5      0     1     0      0      1     0    0    0      0
##   50    22     13     1     0      0     10     0    2    1      0
##   56     0      0     0     0      1      0     0    0    1      0
##   57     0      0     0     0      0      1     5    0    3      0
##   58     1      0     0     0      3      0     0    5    0      0
##   60     0      0     0     0      0      1     0    0    0      0
##   61     0      0     0     2      0      6     0    0    0      9
##   62     0      3     1     0      4      1     1    1    2      0
```

```
love_as_1 = findAssocs(dtm_1, "amore", 0.25)
love_ass_1 = data.frame(love_as_1)
love_ass_1 = cbind(row.names(love_ass_1), love_ass_1)
colnames(love_ass_1) = c("word", "freq")
head(love_as_1)
```

```
## $amore
##      volata       bimbi      intero   vedermeli     operaio     porcile
##        0.74        0.74        0.74        0.74        0.74        0.74
##     ragazzi     ragazze   difendono      piazze uccidendoci    bastardo
##        0.74        0.74        0.74        0.74        0.74        0.74
##   vigliacco     memoria     gettata     chiamami       dovrà      finire
##        0.74        0.74        0.74        0.74        0.74        0.74
##  riempiremo   disperato       tuono     difendi    restasse    farfalle
##        0.74        0.74        0.74        0.74        0.74        0.74
##  togliergli    spengono   temporali        voci       madre   credevamo
##        0.74        0.74        0.74        0.74        0.74        0.74
##       sputo   maledetta        idee     umanità       perso      ancora
##        0.74        0.73        0.72        0.69        0.65        0.56
##      sempre    regalato       poeta    scrivere        vera    continua
##        0.53        0.51        0.51        0.51        0.51        0.48
##       libro    pensiero     signori     giocare    portando     scritte
##        0.48        0.48        0.48        0.42        0.41        0.41
##      timore    lascerai  sceglierai  passeranno   primavere      freddi
```

```
##        0.41        0.41        0.41        0.41        0.41        0.41
##     stupidi   maledette       perse       altre  sbaglierei    nasconde
##        0.41        0.41        0.41        0.41        0.41        0.40
##       notte    silenzio       dimmi       avere      lavoro       belli
##        0.40        0.40        0.38        0.38        0.37        0.37
##      grande     gridare        vent       barca      urlare       notti
##        0.37        0.37        0.37        0.35        0.35        0.35
##       unico     deserto     musica        mare       penso    arrivare
##        0.33        0.31        0.30        0.28        0.27        0.27
##     dormire     respiro
##        0.27        0.26
```

```r
dtm_2 = testi_all_2 %>% cast_dtm(document = line,term = word,value = freq)
inspect(dtm_1)
```

```
## <<DocumentTermMatrix (documents: 62, terms: 2004)>>
## Non-/sparse entries: 3621/120627
## Sparsity           : 97%
## Maximal term length: 15
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs amore ancora cuore occhi quando sempre senza solo vita vorrei
##   45     4      0     0     0      4      0     0    1    0      2
##   46     0      4     0     1      3      1     1    2    0      0
##   49     5      0     1     0      0      1     0    0    0      0
##   50    22     13     1     0      0     10     0    2    1      0
##   56     0      0     0     0      1      0     0    0    1      0
##   57     0      0     0     0      0      1     5    0    3      0
##   58     1      0     0     0      3      0     0    5    0      0
##   60     0      0     0     0      0      1     0    0    0      0
##   61     0      0     0     2      0      6     0    0    0      9
##   62     0      3     1     0      4      1     1    1    2      0
```

```r
love_as_2 = findAssocs(dtm_2, "amore", 0.25)
love_ass_2 = data.frame(love_as_2)
love_ass_2 = cbind(row.names(love_ass_2), love_ass_2)
colnames(love_ass_2) = c("word", "freq")

head(love_as_2)
```

```
## $amore
##          amare          gambe       trovarci          persi   avvicinarci         bocche
##           0.50           0.48           0.48           0.48          0.48           0.48
##    assaggiarci        nascere         pronto     nonostante    afferrarci         girare
##           0.48           0.48           0.48           0.48          0.48           0.48
##     ingranaggi        vediamo riconosciamo       perfetti      macchine       miracolo
##           0.48           0.48           0.48           0.48          0.48           0.48
##          nervi          anime       chiederò        stringo   dividiamolo          fatti
##           0.48           0.48           0.48           0.48          0.48           0.47
##         avanti        profumi        giovane         alberi     schiarita    innamorato
##           0.46           0.42           0.42           0.42          0.42           0.41
##          senso          prati          senti          cuore       capirai        sorriso
```

```
##         0.40         0.39         0.38         0.37         0.37         0.36
##         dire       stella     chitarra     arrivare   sentimento      braccia
##         0.35         0.32         0.31         0.31         0.30         0.28
##       chiudi       voluto       meglio        suono       pianto       ideale
##         0.28         0.27         0.27         0.27         0.27         0.27
##        andrò       chiuso         voce       stanco
##         0.27         0.27         0.26         0.25
```

```
dtm_3 = testi_all_3 %>% cast_dtm(document = line,term = word,value = freq)
inspect(dtm_3)
```

```
## <<DocumentTermMatrix (documents: 47, terms: 1577)>>
## Non-/sparse entries: 2766/71353
## Sparsity           : 96%
## Maximal term length: 17
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs amore ancora cuore forse mare mondo senza sole solo vita
##   18     6      1     0     0    2     2     0    1    1    0
##   20     5      9     0     0    0     1     0    0    0    0
##   29     0      0     0     0    8     1     4    8    4    3
##   31     2      0     0     0    0     0     0    0    2    0
##   38     3      0     1     0    1     1     4    4    1    0
##   41     8      0     0     0    0     0     0    0    1    3
##   44     0      0     0     4    0    11     0    0    4    5
##   45     0      0     4     0    3     0     5    0    1    3
##   46     3      0     0     1    0     0     0   12    0    0
##   47     0      0     2     0    0     4     0    0    7    0
```

```
love_as_3 = findAssocs(dtm_3, "amore", 0.25)
love_ass_3 = data.frame(love_as_3)
love_ass_3 = cbind(row.names(love_ass_3), love_ass_3)
colnames(love_ass_3) = c("word", "freq")
head(love_as_3)
```

```
## $amore
##   svegliarsi       centro   situazioni   riscoprire       tratto   semplicità
##         0.81         0.81         0.81         0.81         0.81         0.81
##      chiamalo   risponderà        mezze     tradisce      finisce     parlarsi
##         0.81         0.81         0.81         0.81         0.81         0.81
##      sapendo        costa    sincerità      carezze     spaccano     arrivare
##         0.81         0.81         0.81         0.81         0.81         0.58
##       uscita ricominciare     trovarsi       dietro         pace        verità
##         0.52         0.49         0.48         0.46         0.43         0.41
##       grande      ricordi     qualcosa        mezzo    camminare      mattina
##         0.36         0.34         0.32         0.29         0.29         0.29
##       realtà        pieni       ferito        pugni         lupi       scuola
##         0.27         0.26         0.26         0.26         0.26         0.26
##    ventinove      maestra   chiedevano         coro       occhio      collana
##         0.26         0.26         0.26         0.26         0.26         0.26
##       pietra       magica     stringevo     portarti   frantumava         ossa
##         0.26         0.26         0.26         0.26         0.26         0.26
```

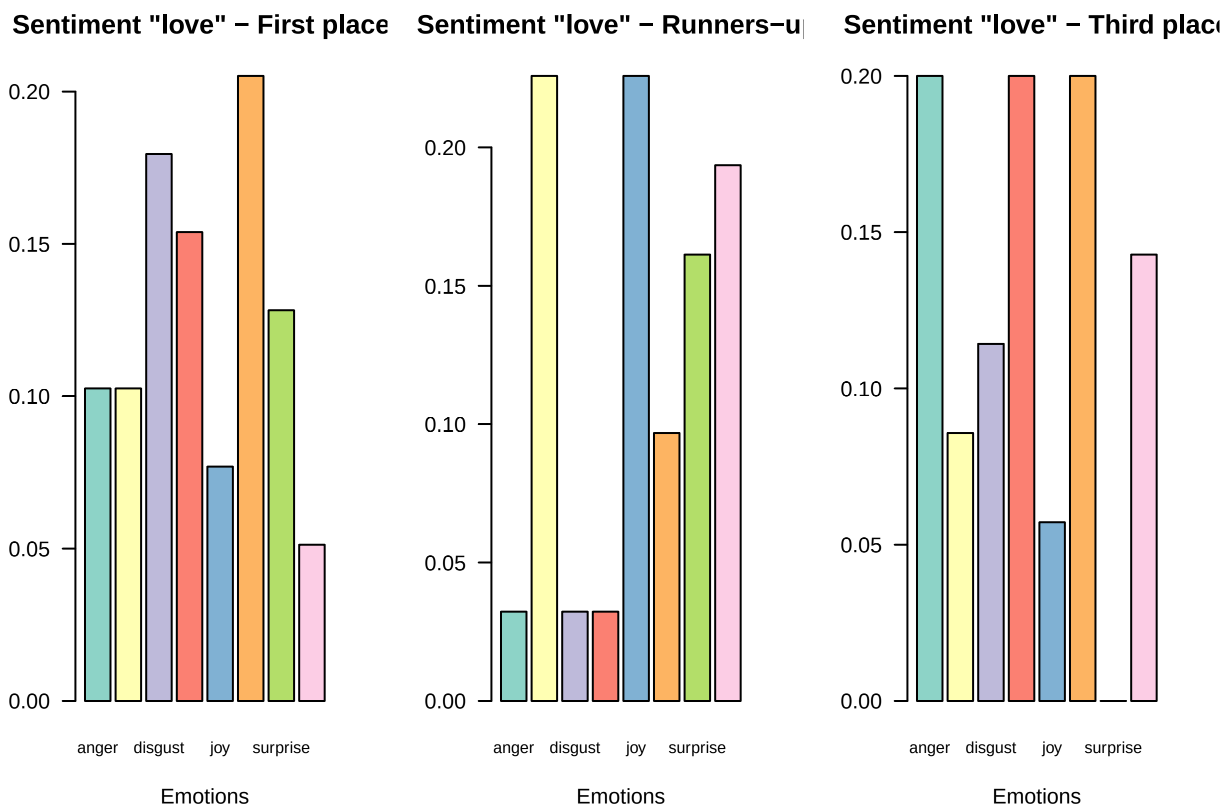
```
##    pensavano       dovuto        libro         odio   insegnarmi       smesso
##         0.26         0.26         0.26         0.26         0.26         0.26
##        farmi        provi    riuscirai      ricorda     colpisce       figlio
##         0.26         0.26         0.26         0.26         0.26         0.26
##   diventerai  dimenticato   difenderti      portavo pantaloncini       finita
##         0.26         0.26         0.26         0.26         0.26         0.26
##      restava        morsi       ferita       chiude    aspettavi       scegli
##         0.26         0.26         0.26         0.26         0.26         0.26
##      diversa     violenza  disobbedire      vietato        spara     distante
##         0.26         0.26         0.26         0.26         0.26         0.26
##        tanto
##         0.25
```

After creating these dataframes, we proceeded to apply the “get_nrc_sentiment” function to determine the sentiment associated with these words.

```
sent_love_1 <- get_nrc_sentiment(love_ass_1[,1], lang="italian")
sent_love_2 <- get_nrc_sentiment(love_ass_2[,1], lang="italian")
sent_love_3 <- get_nrc_sentiment(love_ass_3[,1], lang="italian")
```

This allowed us to represent, using barplots, the feelings associated with the term “love” in the three different positions.

```
par(mfrow= c(1,3))

barplot(
   colSums(prop.table(sent_love_1[, 1:8])),
   space = 0.2,
   horiz = FALSE,
   las = 1,
   cex.names = 0.7,
   col = brewer.pal(n = 8, name = "Set3"),
   main = "Sentiment \"love\" - First place ",
   xlab="Emotions", ylab = NULL)

barplot(
   colSums(prop.table(sent_love_2[, 1:8])),
   space = 0.2,
   horiz = FALSE,
   las = 1,
   cex.names = 0.7,
   col = brewer.pal(n = 8, name = "Set3"),
   main = "Sentiment \"love\" - Runners-up ",
   xlab="Emotions", ylab = NULL)

barplot(
   colSums(prop.table(sent_love_3[, 1:8])),
   space = 0.2,
   horiz = FALSE,
   las = 1,
   cex.names = 0.7,
   col = brewer.pal(n = 8, name = "Set3"),
   main = "Sentiment \"love\" - Third place",
   xlab="Emotions", ylab = NULL)
```




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

It can be observed how the first classified express a feeling of anger, sadness and disgust towards love. Their songs deal with themes of damaged relationships and romantic disappointments, evoking negative emotions in the audience; instead, the runners-up convey a strong feeling of joy associated with love. Their songs celebrate the beauty and happiness of love, bringing a positive and pleasant atmosphere; finally, the third classified, a mix of anger, sadness and fear linked to love emerges. Their songs reflect complex relationships and emotional difficulties, evoking negative and conflicting emotions. Finally, using the "Madda" lexicon, we examined the negative and positive feelings associated with the word "love." To accomplish this, we first created a new dataframe called "word_sent_[i]". Within this dataframe, we added a new column where we assigned the "neutral" sentiment to each word. Subsequently, by exploiting a for loop and making use of the "Madda" lexicon, we were able to classify words with "positive" and "negative" sentiments. Finally, we created the "tt_love_ss_[i]" object where we associated the sentiment obtained via the for loop from the "word_sent_[i]" dataframe to each word in the "love_ass_[i]" dataframe.

```
data(vocabolariMadda)
df_voc = reshape2::melt(vocabolariMadda,value.name = "word")

word_sent_1 = data.frame(word = unique(love_ass_1$word), s="neutrale")

for( i in 1:NROW(word_sent_1)) {
  idw = which(str_starts(word_sent_1$word[i],df_voc$word))
  if(length(idw) != 0) word_sent_1$s[i] = df_voc$L1[idw]

  if(i %% 100 == 0) cat(i,"\n")
```

```
}

tt_love_ss_1 = love_ass_1 %>% left_join(word_sent_1, by = "word")

word_sent_2 = data.frame(word = unique(love_ass_2$word), s="neutrale")
for( i in 1:NROW(word_sent_2)) {
  idw = which(str_starts(word_sent_2$word[i],df_voc$word))
  if(length(idw) != 0) word_sent_2$s[i] = df_voc$L1[idw]
  if(i %% 100 == 0) cat(i,"\n")
}
tt_love_ss_2 = love_ass_2 %>% left_join(word_sent_2, by = "word")

word_sent_3 = data.frame(word = unique(love_ass_3$word), s="neutrale")
for( i in 1:NROW(word_sent_3)) {
  idw = which(str_starts(word_sent_3$word[i],df_voc$word))
  if(length(idw) != 0) word_sent_3$s[i] = df_voc$L1[idw]
  if(i %% 100 == 0) cat(i,"\n")
}
tt_love_ss_3 = love_ass_3 %>% left_join(word_sent_3, by = "word")
```

At this point it was possible to create an object, "neg_word_love_[i]", containing the negative words associated with the term "love", and an object "pos_word_love_[i]", containing the positive words.

```
neg_word_love_1 = tt_love_ss_1 %>% filter(s == "negative")
head(neg_word_love_1, 10)
```

```
##           word freq        s
## 1      porcile 0.74 negative
## 2  uccidendoci 0.74 negative
## 3     bastardo 0.74 negative
## 4    vigliacco 0.74 negative
## 5    disperato 0.74 negative
## 6    maledetta 0.73 negative
## 7        perso 0.65 negative
## 8     lascerai 0.41 negative
## 9   passeranno 0.41 negative
## 10      freddi 0.41 negative
```

```
pos_word_love_1 = tt_love_ss_1 %>% filter(s == "positive")
head(pos_word_love_1, 10)
```

```
##         word freq        s
## 1    memoria 0.74 positive
## 2     finire 0.74 positive
## 3  credevamo 0.74 positive
## 4    umanità 0.69 positive
## 5   regalato 0.51 positive
## 6       vera 0.51 positive
## 7   continua 0.48 positive
## 8   pensiero 0.48 positive
## 9   portando 0.41 positive
## 10   stupidi 0.41 positive
```

```r
neg_word_love_2 = tt_love_ss_2 %>% filter(s == "negative")
head(neg_word_love_2, 10)
```

```
##        word freq        s
## 1     amare 0.50 negative
## 2     persi 0.48 negative
## 3   macchine 0.48 negative
## 4    profumi 0.42 negative
## 5 schiarita 0.42 negative
## 6     prati 0.39 negative
## 7    chiuso 0.27 negative
## 8    stanco 0.25 negative
```

```r
pos_word_love_2 = tt_love_ss_2 %>% filter(s == "positive")
head(pos_word_love_2, 10)
```

```
##            word freq        s
## 1        pronto 0.48 positive
## 2  riconosciamo 0.48 positive
## 3      perfetti 0.48 positive
## 4      miracolo 0.48 positive
## 5         anime 0.48 positive
## 6    innamorato 0.41 positive
## 7       sorriso 0.36 positive
## 8        stella 0.32 positive
## 9        meglio 0.27 positive
## 10       ideale 0.27 positive
```

```r
neg_word_love_3 = tt_love_ss_3 %>% filter(s == "negative")
head(neg_word_love_3, 10)
```

```
##           word freq        s
## 1   riscoprire 0.81 negative
## 2     tradisce 0.81 negative
## 3     parlarsi 0.81 negative
## 4       ferito 0.26 negative
## 5        pugni 0.26 negative
## 6   frantumava 0.26 negative
## 7    pensavano 0.26 negative
## 8         odio 0.26 negative
## 9     colpisce 0.26 negative
## 10 dimenticato 0.26 negative
```

```r
pos_word_love_3 = tt_love_ss_3 %>% filter(s == "positive")
head(pos_word_love_3, 10)
```

```
##          word freq        s
## 1  semplicità 0.81 positive
## 2  risponderà 0.81 positive
## 3     finisce 0.81 positive
## 4   sincerità 0.81 positive
```

```
## 5     carezze 0.81 positive
## 6        pace 0.43 positive
## 7      verità 0.41 positive
## 8      grande 0.36 positive
## 9     maestra 0.26 positive
## 10     magica 0.26 positive
```

At this point it was possible to observe the wordclouds for terms with a negative sentiment associated with the word love.

```r
par(mfrow=c(1,3))
wordcloud(words = neg_word_love_1$word, freq = neg_word_love_1$freq,
          min.freq = 0, color = brewer.pal(4,"Dark2") )
wordcloud(words = neg_word_love_2$word, freq = neg_word_love_2$freq,
          min.freq = 0, color = brewer.pal(4,"Dark2") )
wordcloud(words = neg_word_love_3$word, freq = neg_word_love_3$freq,
          min.freq = 0, color = brewer.pal(4,"Dark2") )
```

In this context, words such as “bastard”, “desperate”, “killing us” and “cowardly” emerged. These terms reflect a love experience characterized by disappointment, suffering and betrayal. The songs explored the dark side of love, highlighting the painful side of relationships and the emotional wounds that can result. On the other hand, words have emerged that evoke positive feelings related to love, such as “humanity”, “caresses” and “memory”. These words reveal a longing for human connection, affection, and the healing power of love-related memories. The songs sought to celebrate the positive aspects of love, recalling its ability to inspire joy, affection and a sense of belonging.

```r
par(mfrow=c(1,3))
wordcloud(words = pos_word_love_1$word, freq = pos_word_love_1$freq,
          min.freq = 0, color = brewer.pal(4,"Dark2") )
wordcloud(words = pos_word_love_2$word, freq = pos_word_love_2$freq,
          min.freq = 0, color = brewer.pal(4,"Dark2") )
wordcloud(words = pos_word_love_3$word, freq = pos_word_love_3$freq,
          min.freq = 0, color = brewer.pal(4,"Dark2") )
```

In conclusion, the Sanremo Festival offered a platform to reflect on the emotional state of the Italian nation, capturing the nuances of its collective feelings. The songs acted as a sincere reflection of the lived experiences of Italians, offering emotional songs and inviting reflection on the human condition. The top three at the Sanremo Festival present a variety of feelings associated with love in their songs. Overall, a common trend of sadness and anger emerges, but with some significant differences between the groups. The Sanremo Festival explored the complexity of love through the words used in songs. It offered an insight into the conflicting emotions that love can evoke, highlighting both its beauty and its challenges.

Future developments

In the context of our analysis, there are several future aspects that could further enrich our project. One of the main areas of development concerns advanced natural language processing. Thanks to machine learning and artificial intelligence, we could benefit from more sophisticated algorithms and more advanced language models that can understand and analyze text more accurately. This could include better understanding context, detecting nuances of meaning, and identifying linguistic subtleties. Another promising aspect concerns the analysis of emotions and feelings. Currently, sentiment analysis mainly focuses on the evaluation of positive, negative or neutral emotions. However, future developments could enable more sophisticated machine learning models capable of identifying a wider range of emotions and assessing their intensity. This would offer a deeper understanding of the feelings expressed in the text and the emotional nuances associated with the words used. Another development perspective concerns the analysis of semantic relationships between words. In the future, algorithms capable of recognizing and analyzing relationships between words could be developed. This would allow for a deeper understanding of the meanings of words and the conceptual connections between them. The extraction of structured information from text represents another interesting area of development. We may explore advanced methods to automatically identify and extract specific data, such as names of people, places, dates, entities, or key concepts from text. This would facilitate the aggregation and organization of information, making subsequent more efficient analysis possible. Finally, the integration of multimedia data could enrich our analysis. The combination of text, images, audio or video may offer a more complete understanding of a given topic or context. For example, we might explore combined text and image analysis to identify visual themes associated with certain concepts, or analyze transcribed text from videos or audio recordings. In conclusion, our project offers numerous opportunities for future development. Advanced natural language processing, emotion analysis, semantic relationship analysis, structured information extraction, and multimedia data integration are just some of the directions that could lead to greater understanding and analysis of our text.