UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE POLITICHE,
ECONOMICHE E SOCIALI

Political communication and populist rhetoric An analysis of Italian politicians in the digital arena.

By

RICCARDO RUTA

DRAFT
DRAFT
DRAFT

07/22

Abstract

(the spacing is set to 1.5)

no more than 250 words for the abstract

- a description of the research question – what we know and what we don't know
- how the research has attempted to answer to this question
- a brief description of the methods
- brief results
- key conclusions that put the research into a larger context

Contents

1 Data cleaning **1**

1.1 Import the dataset and check variables 1

1.2 Adjust date.time format 1

1.2.1 Check the conversion 2

1.3 Create the week variable 2

1.3.1 Check the variable 2

1.4 Create the month variable 3

1.4.1 Check the number of month 3

1.5 Create the trimester variable 3

1.5.1 Check the number of trimesters 4

1.6 Create the year variables 4

1.6.1 Check the number of years 4

1.7 Count the number of missing values 5

1.7.1 Inspect where are the missings 5

1.7.2 Remove rows with missing tweets 6

1.8 Check that the variables make sense 7

1.8.1 Adjust the variable genere 7

1.8.2 Verify the substitution 8

1.9 Create a new dataset selecting only necessary informations 8

1.10 Create the corpus 9

1.11 Create the DFM 9

1.12 Remove the emoji 10

1.12.1 Now the data are ready for the next analysis 11

2 Preliminar analysis **12**

2.1 Who is inside this dataset? 12

2.2 Topfeatures frequency 13

2.2.1 Relative frequency of the topfeatures by Party ID 15

2.3 Most common hashtag 16

2.3.1 Most common hashtag by Gender 17

2.3.2 Co-occurrence Plot of hashtags 18

2.4 Most frequently mentioned usernames 19

2.4.1 Most frequently mentioned usernames by gender 20

2.4.2 Co-occurrence plot of usernames 22

2.5 How many times a politician cite his/her party 22

2.5.1 Create the variable with the name of the official Twitter account 24

2.5.2 Count for each party how many times a politician cite their respective party 24

2.6 How many times the party leader is cited by his/her party 25

2.6.1 Create the variable with the official leader's account for every party 25

2.6.2 Count for each party how many times a politician cite his/her party leader 26

2.7 How many times a politician cite itself in the tweet 27

3 Dictionary analysis **30**

3.1 Create the dictionary 30

3.1.1 Group and weight the dfm 32

3.2 Decadri_Boussalis_Grundl 34

3.2.1 Level of sparsity 34

3.2.2 General level of populism in time divided into 3 components . 35

3.2.3 General level of populism in time 37

3.2.4 Most populist parliamentary group for each component 38

3.2.5 Most populist parliamentary group 40

3.3 Rooduijn_Pauwels_Italian 42

3.3.1 Level of sparsity 42

3.3.2 General level of populism in time 43

3.3.3 Most populist parliamentary group 43

3.3.4 Distribution of parliamentary group populism 44

3.4 Grundl_Italian_adapted 47

3.4.1 Level of sparsity 47

3.4.2 General level of populism in time 48

3.4.3 Most populist parliamentary group 48

3.4.4 Distribution of parliamentary grouppopulism 49

3.5 Decadri_Boussalis 51

3.5.1 Level of sparsity 51

3.5.2 General level of populism in time 51

3.5.3 Most populist parliamentary group 52

3.5.4 Distribution of party populism 52
3.6 Compare the general level of populism in time for the dictionaries . . 54
3.7 Compare how the dictionaries score for the most populist parliamentary group 55

4 Sentiment analysis **57**

4.1 Inspect the dictionary 57
4.1.1 Clean text from dataframe 57
4.2 Create the filtered dataframes 58
4.3 Create nrc objects 59
4.4 Giorgia Meloni 61
4.4.1 Proportion of the emotion 61
4.4.2 Wordcloud of emotions 62
4.5 Giuseppe Conte 64
4.5.1 Proportion of the emotion 64
4.5.2 Wordcloud of emotions 65
4.6 Matteo Renzi 66
4.6.1 Proportion of the emotion 66
4.6.2 Wordcloud of emotions 67
4.7 Matteo Salvini 68
4.7.1 Proportion of the emotion 68
4.7.2 Wordcloud of emotions 69
4.8 Enrico Letta 70
4.8.1 Proportion of the emotion 70

4.8.2 Wordcloud of emotions 71
4.9 Silvio Berlusconi 72
4.9.1 Proportion of the emotion 72
4.9.2 Wordcloud of emotions 73
4.10 Roberto Speranza 74
4.10.1 Proportion of the emotion 74
4.10.2 Wordcloud of emotions 75

5 LDA Topic model analysis **76**
5.1 CREATE THE DTM 76
5.1.1 Remove all the account's mentions 76
5.1.2 Convert the Document Feature Matrix (Dfm) in a Topic Model (Dtm) 76
5.2 FIND THE BEST NUMBER OF TOPICS K 76
5.2.1 Search the best number of Topics comparing coherence and exclusivity values 76
5.2.2 Plot the values of coherence and exclusivity in order to find the best K 77
5.3 ANALISYS OF THE TOPICS 78
5.3.1 Run the analysis selecting k = 11 78
5.3.2 The most important terms from the model, for each topic . . 78
5.3.3 Interpret the terms 79

6 FER: Facial Emotion Recognition Analysis **80**
6.1 Report on the analysis made with FER Python package 80

1 Data cleaning

1.1 Import the dataset and check variables

```
# import the data
tw <-  read_csv("data/large_files/politicians_final_corrected.csv",
                show_col_types = FALSE )


kable(colnames(tw), col.names = "variables")
```

variables
tw_screen_name
nome
tweet_testo
creato_il
creato_il_code
url
party_id
genere
chamber
status

1.2 Adjust date.time format

```
# RUN IN THIS ORDER !!
Sys.setlocale("LC_TIME", "C")
tw$date <- as.Date(strptime(tw$creato_il,"%a %b %d %H:%M:%S %z %Y",
                            tz = "CET"))
tw$date <- na.replace(tw$date, as.Date(tw$creato_il))
```

1.2.1 Check the conversion

```
check_dates <- tw %>% select(creato_il,date)
kable(head(check_dates), col.names = c("Old date", "New date"))
```

Old date	New date
2021-02-13	2021-02-13
2021-02-09	2021-02-09
2021-02-07	2021-02-07
2021-01-21	2021-01-21
2021-01-21	2021-01-21
2021-01-20	2021-01-20

```
kable(tail(check_dates), col.names = c("Old date", "New date"))
```

Old date	New date
Mon Dec 28 09:51:35 +0000 2020	2020-12-28
Tue Jul 20 11:15:44 +0000 2021	2021-07-20
Thu Nov 26 13:46:51 +0000 2020	2020-11-26
Fri Oct 15 17:28:57 +0000 2021	2021-10-15
Wed Jun 03 12:22:31 +0000 2020	2020-06-03
Fri Dec 03 21:01:20 +0000 2021	2021-12-03

1.3 Create the week variable

```
tw <- tw %>% mutate(week = cut.Date(date, breaks = "1 week", labels = FALSE))
```

1.3.1 Check the variable

Inspect the first and the last dates and check if the number of weeks is correct

```r
max(tw$date)
```

```
## [1] "2022-04-18"
```

```r
min(tw$date)
```

```
## [1] "2020-01-01"
```

```r
difftime(max(tw$date), min(tw$date), units = "weeks")
```

```
## Time difference of 119.7143 weeks
```

1.4 Create the month variable

```r
tw <- tw %>% mutate(month = cut.Date(date, breaks = "1 month", labels = FALSE))
```

1.4.1 Check the number of month

```r
max(tw$month)
```

```
## [1] 28
```

```r
length(seq(from = min(tw$date), to = max(tw$date), by = 'month'))
```

```
## [1] 28
```

1.5 Create the trimester variable

```
tw <- tw %>% mutate(quarter = cut.Date(date, breaks = "1 quarter", labels = FALSE))
```

1.5.1 Check the number of trimesters

```
max(tw$quarter)
```

```
## [1] 10
```

```
length(seq.Date(from = min(tw$date), to = max(tw$date), by = 'quarter'))
```

```
## [1] 10
```

1.6 Create the year variables

```
tw <- tw %>% mutate(year = cut.Date(date, breaks = "year", labels = FALSE))
```

1.6.1 Check the number of years

```
max(tw$year)
```

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

```
length(seq.Date(from = min(tw$date), to = max(tw$date), by = 'year'))
```

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

1.7 Count the number of missing values

```
sum(is.na(tw))
```

```
## [1] 154672
```

1.7.1 Inspect where are the missings

```
missings <- c(
sum(is.na(tw$tw_screen_name)),
sum(is.na(tw$nome)),
sum(is.na(tw$tweet_testo)),
sum(is.na(tw$creato_il)),
sum(is.na(tw$creato_il_code)),
sum(is.na(tw$url)),
sum(is.na(tw$party_id)),
sum(is.na(tw$genere)),
sum(is.na(tw$chamber)),
sum(is.na(tw$status)),
sum(is.na(tw$date)),
sum(is.na(tw$week)),
sum(is.na(tw$month)),
sum(is.na(tw$quarter)),
sum(is.na(tw$year)))

missing_df <- data.frame(colnames(tw), missings)
kable(missing_df)
```

colnames.tw.	missings
tw_screen_name	0
nome	0
tweet_testo	6494
creato_il	0
creato_il_code	0
url	148178
party_id	0
genere	0
chamber	0
status	0
date	0
week	0
month	0
quarter	0
year	0

From this check I'll obtain 148178 urls missing, this variable is not collected properly and we will not use in the analysis, and also results 6494 tweets missings, those are the cases when someone post only images or video without text, so the extraction is correct.

1.7.2 Remove rows with missing tweets

```
sum(is.na(tw$tweet_testo))
```

```
## [1] 6494
```

```
tw <- tw %>% drop_na(tweet_testo)
```

1.8 Check that the variables make sense

```
unique(tw$party_id)
```

```
##  [1] "PD"           "FDI"          "M5S"          "FI"           "REG_LEAGUES"
##  [6] "MISTO"        "LEGA"         "IV"           "INDIPENDENTE" "CI"
## [11] "LEU"
```

```
unique(tw$genere)
```

```
## [1] "male"   "female" "male "
```

```
unique(tw$chamber)
```

```
## [1] "NotParl" "Senate"  "Camera"
```

```
unique(tw$status)
```

```
## [1] "sottosegretario" "presregione"     "viceministro"    "ministro"
## [5] "segretario"      "Parl"
```

1.8.1 Adjust the variable genere

```
# Remove space from genere variable [RUN ONLY ONCE!]
a <- unique(tw$genere)
a[3]
```

```
## [1] "male "
```

```
which(tw$genere == a[3])
```

```
## [1] 33300 33301 33302 33303 33304
```

```
tw$genere <- gsub(a[3],"male",tw$genere)
```

1.8.2 Verify the substitution

```
which(tw$genere == a[3])
```

```
## integer(0)
```

```
unique(tw$genere)
```

```
## [1] "male"   "female"
```

Now all the variables are ready for next steps

1.9 Create a new dataset selecting only necessary informations

```
# Select variables for the analysis
dataset <- tw %>% select(nome, tweet_testo, genere, party_id,
                         chamber,status, date, week, month, quarter, year )
colnames(dataset)
```

```
##  [1] "nome"        "tweet_testo" "genere"      "party_id"    "chamber"
##  [6] "status"      "date"        "week"        "month"       "quarter"
## [11] "year"
```

1.10 Create the corpus

```
corpus <- corpus(dataset, text = "tweet_testo")
ndoc(corpus)
```

```
## [1] 391197
```

1.11 Create the DFM

```
# Split the corpus into single tokens (remain positional)
doc.tokens <- tokens(corpus,
                                  remove_punct = TRUE,
                                  remove_numbers = TRUE,
                                  remove_symbols = TRUE,
                                  remove_url = TRUE)

# Import my stopwords
my_word <- as.list(read_csv("data/it_stopwords_new_list.csv",
                            show_col_types = FALSE))

# Attach unrecognized symbols
my_list <- c(" ","c'è","+"," ", my_word$stopwords,
             stopwords('italian'), stopwords("english"))

# Save my_list
#save(my_list,file="data/my_list.Rda")

doc.tokens <- tokens_select(doc.tokens, my_list, selection='remove')
```

```r
DFM <- dfm(doc.tokens, tolower = TRUE)
```

1.12 Remove the emoji

```r
# Create a copy of the dfm
test <- DFM
# Remove from the copy all the non ASCII carachters
test@Dimnames$features <- gsub("[^\x01-\x7F]", "", test@Dimnames$features)

# Check the difference from the list of features before and after apply gsub
a <- unique(test@Dimnames$features)
b <- unique(DFM@Dimnames$features)
setdiff(b,a) #I have selected also words that must not be removed

# Create an object with the features after remove non ASCII characters
c <- test@Dimnames$features
# Create an object with the original features
d <- DFM@Dimnames$features

# Create the list of the removed features
diff <- setdiff(d,c)
emoji <- diff[diff %>% nchar() < 4]
emoji <- list(emoji)
# Now i can remove this list from the dfm
DFM <- dfm_remove(DFM, emoji)

#save(DFM,file="data/dfm.Rda")
```

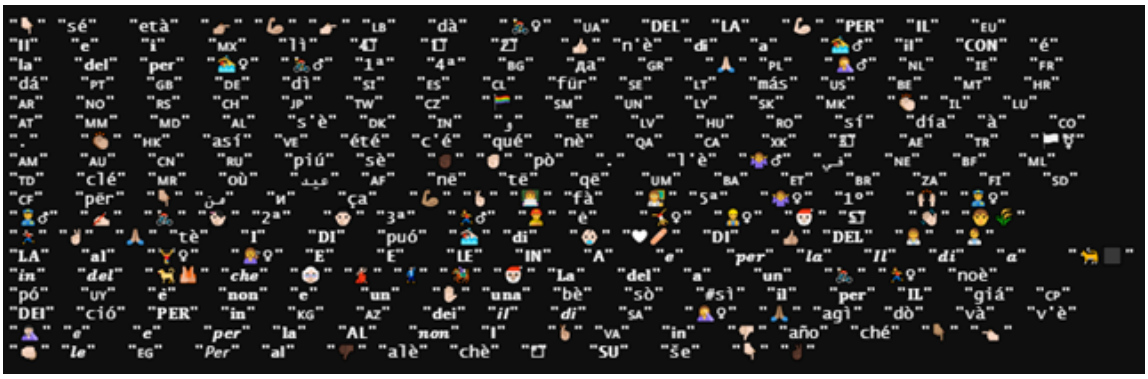


Figure 1: Emoji removed

1.12.1 Now the data are ready for the next analysis

2 Preliminar analysis

2.1 Who is inside this dataset?

```
# Number of parliamentarians
n_parl <- length(unique(dataset$nome))
n_parl
```

```
## [1] 730
```

```
# How many parliamentarians for each party?
n_parl_party <- dataset %>% group_by(party_id) %>% count()
kable(n_parl_party)
```

party_id	n
CI	6954
FDI	36177
FI	65264
INDIPENDENTE	2186
IV	3129
LEGA	87162
LEU	7868
M5S	54418
MISTO	34644
PD	91997
REG_LEAGUES	1398

```
# Gender composition
n_gender <- dataset %>% group_by(genere) %>% count()
kable(n_gender)
```

genere	n
female	111216
male	279981

```
# Wich is the period of analysis?
max(tw$date)
```

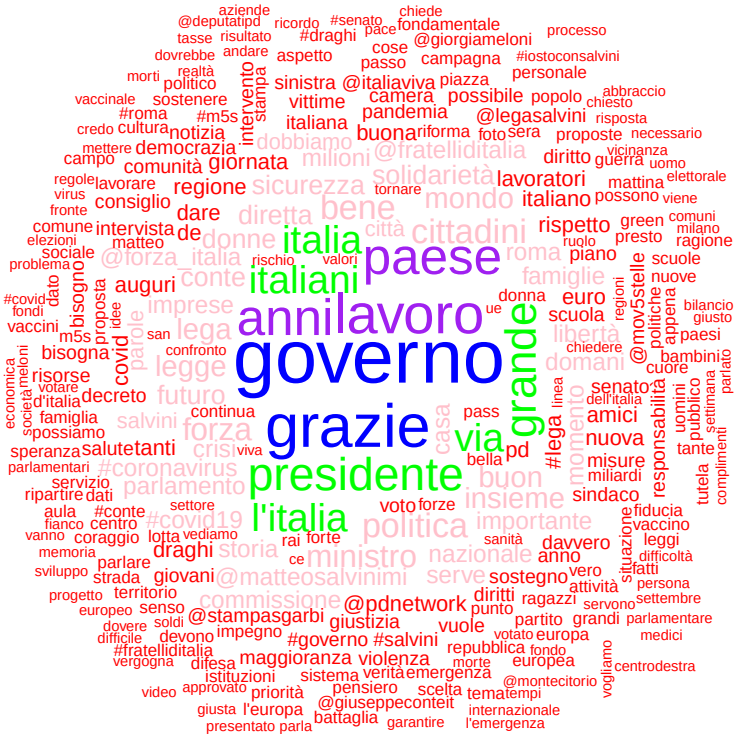
```
## [1] "2022-04-18"
```

```
min(tw$date)
```

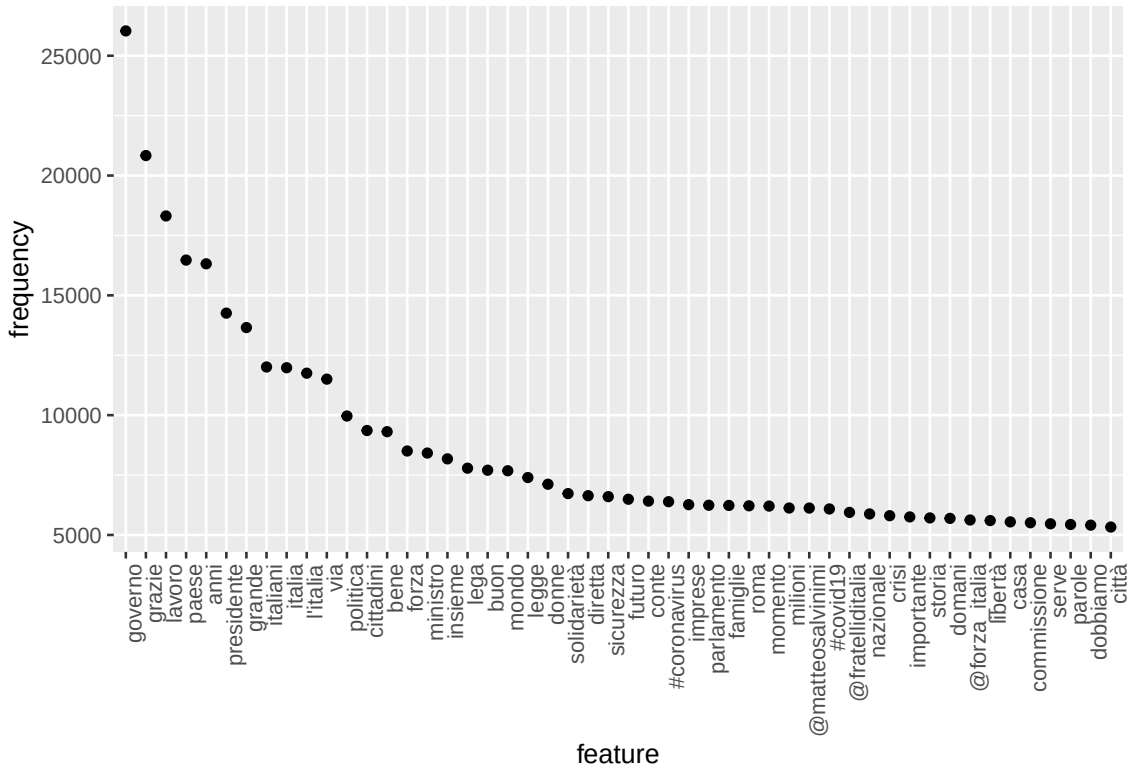
```
## [1] "2020-01-01"
```

2.2 Topfeatures frequency

```
# Textplotwordcloud
set.seed(123)
textplot_wordcloud(DFM, min_count = 20,max_words = 300,
     color = c('red', 'pink', 'green', 'purple', 'blue'))
```



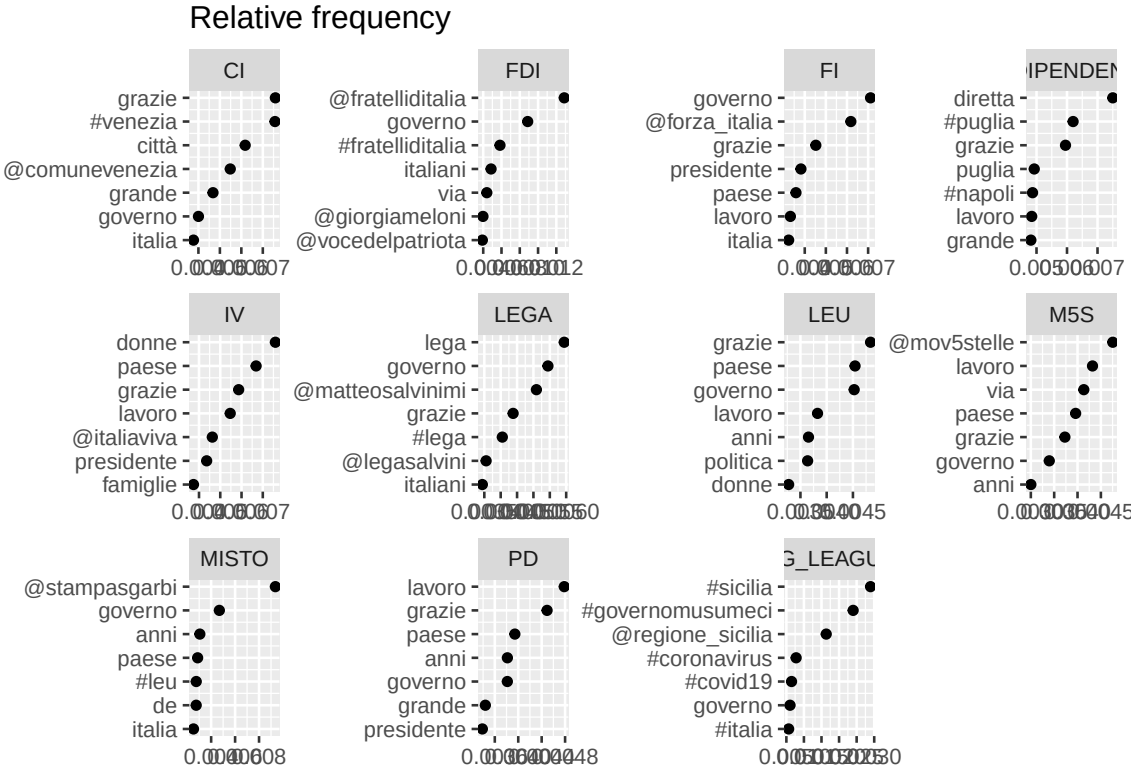

```r
# Plot frequency of the topfeatures in the DFM

features_dfm <- textstat_frequency(DFM, n = 50)

# Sort by reverse frequency order

features_dfm$feature <- with(features_dfm, reorder(feature, -frequency))

ggplot(features_dfm, aes(x = feature, y = frequency)) +

  geom_point() +

  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



2.2.1 Relative frequency of the topfeatures by Party ID

```
# group and weight the DFM
dfm_party_weight <- dfm_group(DFM, groups = party_id) %>%
  dfm_weight(scheme = "prop")

# Plot relative frequency by party_id
freq_weight <- textstat_frequency(dfm_party_weight, n = 7, groups = party_id)



ggplot(data = freq_weight, aes(x = nrow(freq_weight):1, y = frequency)) +
     geom_point() +
     facet_wrap(~ group, scales = "free") +
```

```
    coord_flip() +
    scale_x_continuous(breaks = nrow(freq_weight):1,
                       labels = freq_weight$feature) +
  ggtitle("Relative frequency") +
  labs(x = NULL, y = NULL)
```

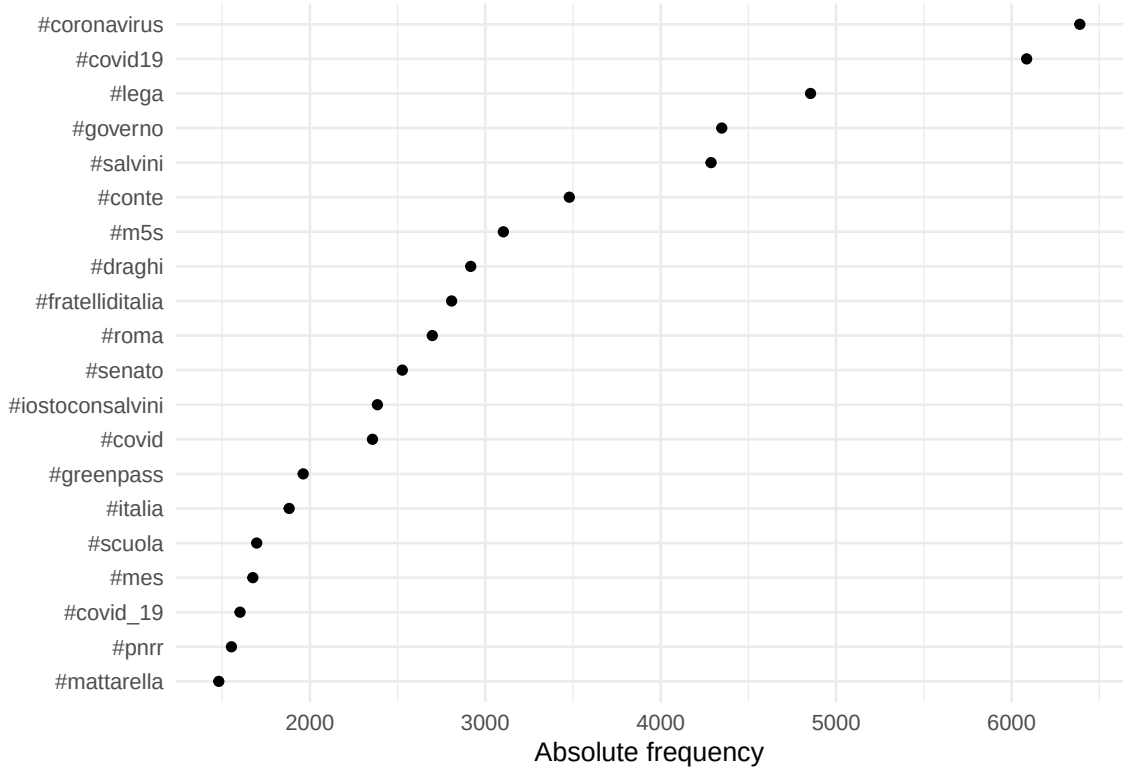


2.3 Most common hashtag

```
tag_dfm <- dfm_select(DFM, pattern = "#*")
toptag <- names(topfeatures(tag_dfm, 20))
toptag
```

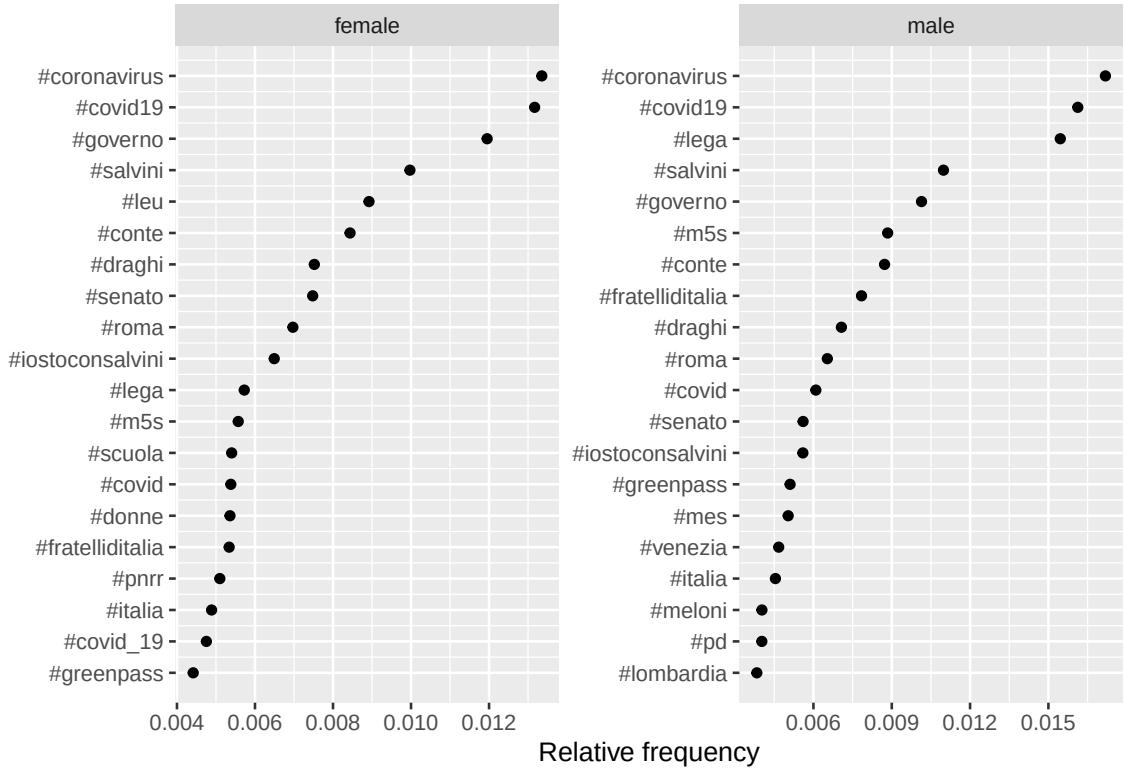
```
##  [1] "#coronavirus"     "#covid19"         "#lega"            "#governo"
```

```
##  [5] "#salvini"          "#conte"           "#m5s"             "#draghi"
##  [9] "#fratellliditalia" "#roma"            "#senato"          "#iostoconsalvini"
## [13] "#covid"            "#greenpass"       "#italia"          "#scuola"
## [17] "#mes"              "#covid_19"        "#pnrr"            "#mattarella"
```



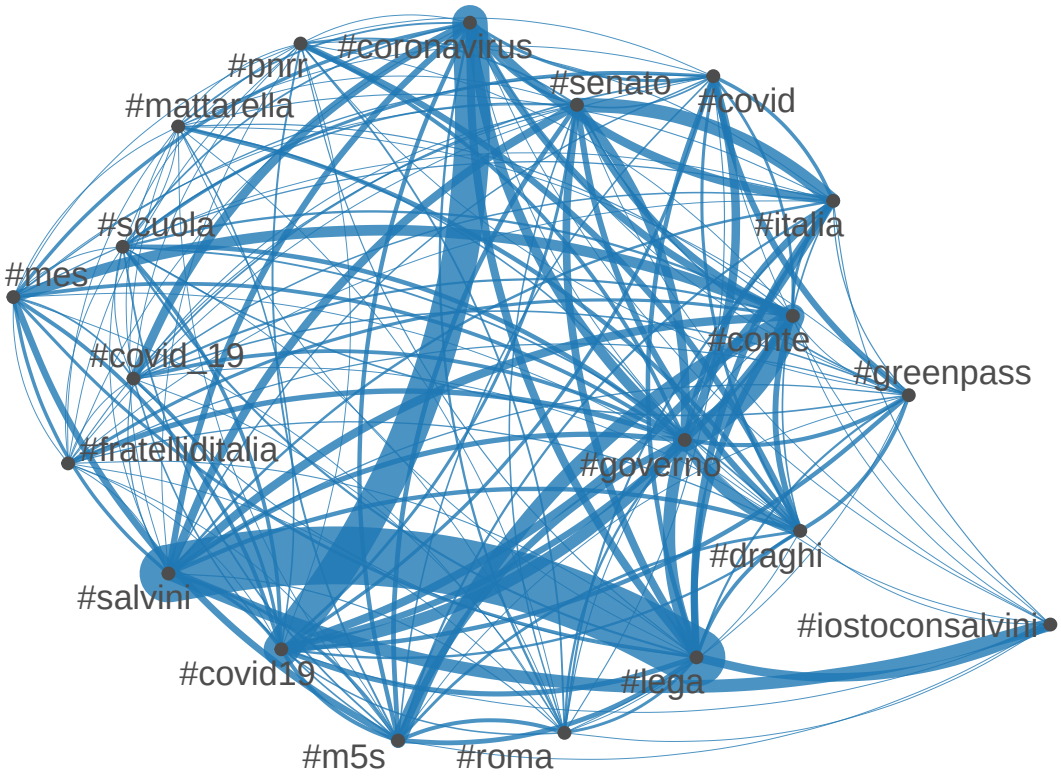
2.3.1 Most common hashtag by Gender

```
# group and weight the DFM
dfm_gender_weight <- dfm_group(tag_dfm, groups = genere) %>%
  dfm_weight(scheme = "prop")


tstat_freq <- textstat_frequency(dfm_gender_weight, n = 20,
                                 groups = dfm_gender_weight$genere)
```



2.3.2 Co-occurrence Plot of hashtags

```
# NOT WEIGHTED
tag_dfm_NOT_W <- dfm_select(DFM, pattern = "#*")
toptag_NOT <- names(topfeatures(tag_dfm_NOT_W, 20))


tag_fcm_NOT <- fcm(tag_dfm_NOT_W)
set.seed(666)
topgat_fcm_NOT <- fcm_select(tag_fcm_NOT, pattern = toptag_NOT)
textplot_network(topgat_fcm_NOT, min_freq = 0.1, edge_alpha = 0.8, edge_size = 5)
```



2.4 Most frequently mentioned usernames

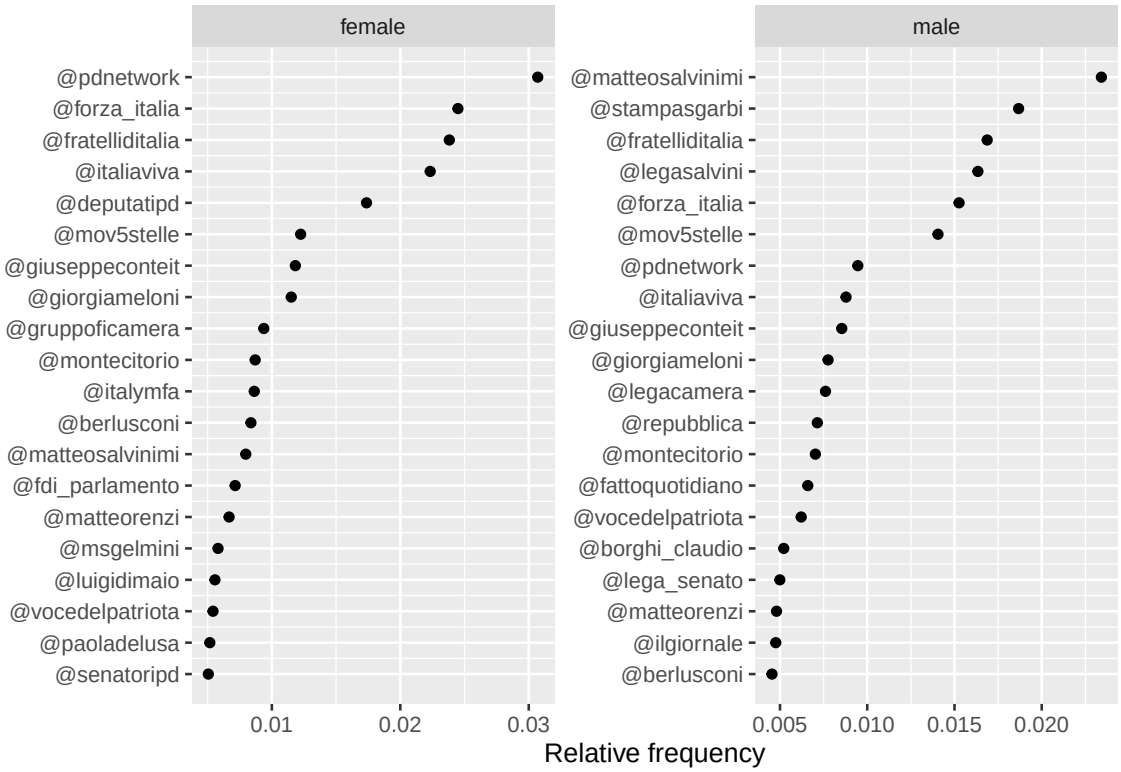
```
user_dfm <- dfm_select(DFM, pattern = "@*")
topuser <- names(topfeatures(user_dfm, 20, scheme = "docfreq"))
kable(topuser, col.names = "Most mentioned username")
```

Most mentioned username
@matteosalvinimi
@fratelliditalia
@forza_italia
@pdnetwork
@stampasgarbi
@mov5stelle
@legasalvini
@italiaviva
@giuseppeconteit
@giorgiameloni
@montecitorio
@deputatipd
@repubblica
@vocedelpatriota
@legacamera
@berlusconi
@matteorenzi
@fattoquotidiano
@enricoletta
@borghi_claudio

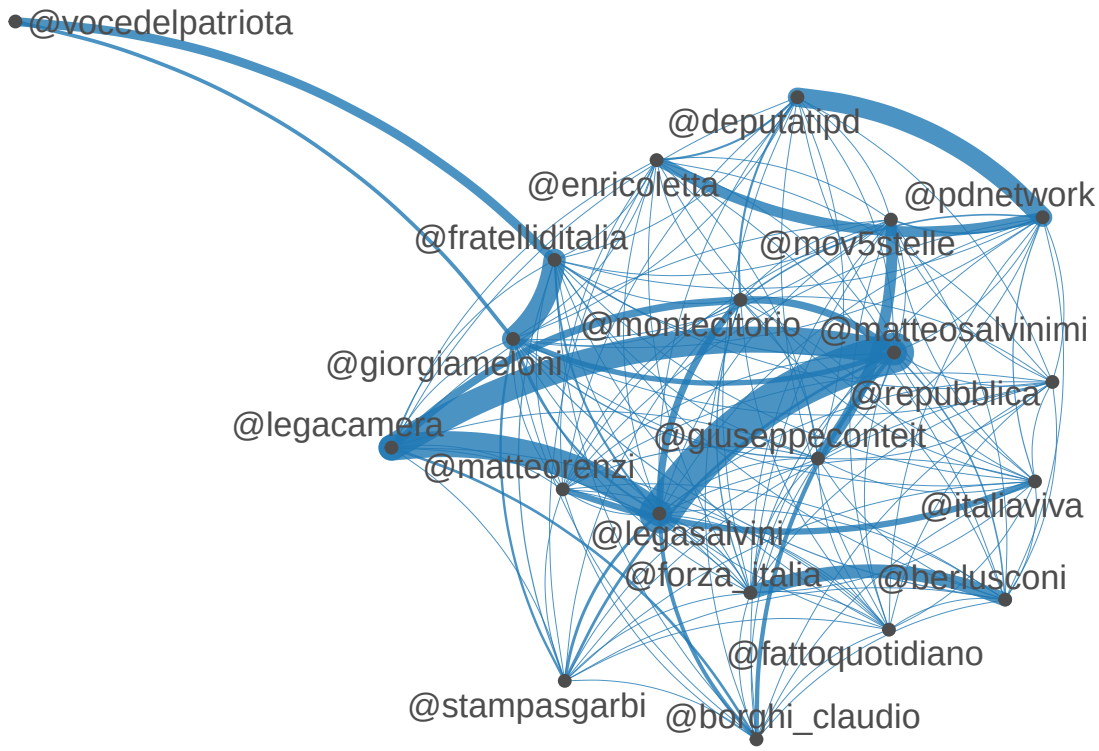
2.4.1 Most frequently mentioned usernames by gender

```
# group and weight the DFM
user_dfm_gender_weight <- dfm_group(user_dfm, groups = genere) %>%
  dfm_weight(scheme = "prop")
```

```
user_tstat_freq <- textstat_frequency(user_dfm_gender_weight, n = 20,
                                      groups = user_dfm_gender_weight$genere)
```



2.4.2 Co-occurrence plot of usernames



2.5 How many times a politician cite his/her party

```
party_citations <- data.frame(first = vector(), second = vector() )
system.time(
for (i in unique(tw$party_id))
{
  a <- paste("#", i ,sep = "")
  b <- tw %>% filter(grepl(a,tweet_testo)&party_id== i) %>% count()
  c <- tw %>% filter(party_id == i) %>% count()
  d <- (b/c) * 100
  party_citations <- rbind(party_citations, cbind(i,b,c,d))
```

```
}
)


#save(party_citations, file = "data/party_citations.Rda")
```

```
colnames(party_citations) <- c("party","n_citations", "Number of tweets", "perc")


kable(party_citations %>% arrange(desc(perc)),
      col.names = c("Party","Number of citations",
                    "number of tweets", "% of citations"))
```

Party	Number of citations	number of tweets	% of citations
M5S	1581	54418	2.9052887
LEGA	511	87162	0.5862647
FDI	131	36177	0.3621085
PD	179	91997	0.1945716
IV	5	3129	0.1597955
FI	62	65264	0.0949988
CI	1	6954	0.0143802
REG_LEAGUES	0	1398	0.0000000
MISTO	0	34644	0.0000000
INDIPENDENTE	0	2186	0.0000000
LEU	0	7868	0.0000000

In the above script i search the # for the parliamentary group, but is very unlikely, for example, that someone use the #IV for talking about the "Italia Viva" party, so i decided to enrich the dataframe creating a new variable with the name of the official twitter page for every party, and repeat the search using it.

I created the variable party_Page for only those parliamentary group that has a direct connection with a party (i excluded Reg_leagues, misto and indipendente)

2.5.1 Create the variable with the name of the official Twitter account

```
tw <- tw %>% mutate(party_page = if_else(party_id == "PD", "@pdnetwork",
if_else( party_id == "FDI", "@FratellidItalia",
    if_else(party_id == "M5S", "@Mov5Stelle",
      if_else(party_id == "FI", "@forza_italia",
        if_else(party_id == "LEGA", "@LegaSalvini",
            if_else(party_id == "CI", "@coraggio_italia",
              if_else(party_id == "LEU", "@liberi_uguali",
                                                    "NA")))))))))
```

2.5.2 Count for each party how many times a politician cite their respective party

```
party_citations_page <- data.frame(first = vector(), second = vector(),
                                   third = vector(), fourth = vector())
system.time(
  for (i in unique(tw$party_page))
  {
    b <- tw %>% filter(grepl(i,tweet_testo)&party_page== i) %>% count()
    c <- tw %>% filter(party_page == i) %>% count()
    d <- (b/c) * 100
    party_citations_page <- rbind(party_citations_page, cbind(i,b,c,d))

  }
)
# save(party_citations_page, file = "data/party_citations_page.Rda")
```

```
colnames(party_citations_page) <- c("party","n_citations",
                                    "Number of tweets", "perc")


kable(party_citations_page %>% filter(party != "NA") %>%
        arrange(desc(perc)),
      col.names = c("Party","Number of citations",
                    "number of tweets", "% of citations"))
```

Party	Number of citations	number of tweets	% of citations
@FratellidItalia	5842	36177	16.1483816
@forza_italia	5203	65264	7.9722358
@Mov5Stelle	3873	54418	7.1171304
@ItaliaViva	201	3129	6.4237776
@pdnetwork	4194	91997	4.5588443
@LegaSalvini	3364	87162	3.8594800
@coraggio_italia	131	6954	1.8838079
@liberi_uguali	16	7868	0.2033554

2.6 How many times the party leader is cited by his/her party

2.6.1 Create the variable with the official leader's account for every party

```
tw <- tw %>% mutate(party_leader =
if_else(party_id == "PD" & date < "2021-03-14", "@nzingaretti",
if_else(party_id == "PD" & date > "2021-03-14", "@EnricoLetta",
if_else( party_id == "FDI", "@GiorgiaMeloni",
if_else(party_id == "M5S" &date < "2020-01-22" , "@luigidimaio",
```

```
if_else(party_id == "M5S" &date > "2020-01-22" &date < "2021-08-06", "@vitocrimi",
if_else(party_id == "M5S" & date > "2021-08-061", "@GiuseppeConteIT",
if_else(party_id == "FI", "@berlusconi",
if_else(party_id == "LEGA", "@matteosalvinimi",                                    i
if_else(party_id == "CI", "@LuigiBrugnaro",
if_else(party_id == "LEU", "@robersperanza",
"NA"))))))))))))
```

2.6.2 Count for each party how many times a politician cite his/ her party leader

```
leader_citations <- data.frame(first = vector(), second = vector(),
                               third = vector(), fourth = vector())
system.time(
  for (i in unique(tw$party_leader))
  {
    b <- tw %>% filter(grepl(i,tweet_testo)&party_leader== i) %>% count()
    c <- tw %>% filter(party_leader == i) %>% count()
    d <- (b/c) * 100
    leader_citations <- rbind(leader_citations, cbind(i,b,c,d))

  }
)
 #save(leader_citations, file = "data/leader_citations.Rda")
```

```
colnames(leader_citations) <- c("leader","n_citations",
                                "Number of tweets", "perc")
```

```
kable(leader_citations %>% filter(leader != "NA") %>%
        arrange(desc(perc)),
      col.names = c("Leader","Number of citations",
                    "Number of tweets", "% of citations"))
```

Leader	Number of citations	Number of tweets	% of citations
@matteosalvinimi	4826	87162	5.5368165
@GiorgiaMeloni	1745	36177	4.8235066
@GiuseppeConteIT	444	15517	2.8613778
@luigidimaio	30	1184	2.5337838
@berlusconi	1533	65264	2.3489213
@EnricoLetta	709	44520	1.5925427
@matteorenzi	46	3129	1.4701182
@nzingaretti	475	47305	1.0041222
@robersperanza	45	7868	0.5719370
@vitocrimi	107	37544	0.2849989
@LuigiBrugnaro	19	6954	0.2732240

2.7 How many times a politician cite itself in the tweet

```
self_citations <- data.frame(first = vector(), second = vector() )
system.time(
for (i in unique(tw$tw_screen_name))
{
  a <- paste("@", i ,sep = "")
  b <- tw %>% filter(grepl(a,tweet_testo) & tw_screen_name== i) %>% count()
  c <- tw %>% filter(tw_screen_name == i) %>% count()
  d <- (b/c) * 100
  self_citations <- rbind(self_citations, cbind(i,b,c,d))
```

```
}
)
#save(self_citations, file = "data/self_citations.Rda")
```

```
colnames(self_citations) <- c("Politician","n_citations",
                              "Number of tweets", "perc")


kable(self_citations %>% filter(n_citations > 2) %>%
        arrange(desc(perc)),
      col.names = c("Politician","Number of citations",
                    "Number of tweets", "% of citations"))
```

Politician	Number of citations	Number of tweets	% of citations
wandaferro1	32	55	58.1818182
FrassinettiP	32	163	19.6319018
albertlaniece	51	282	18.0851064
Luca_Sut	20	341	5.8651026
DalilaNesci	17	341	4.9853372
PatassiniTullio	13	714	1.8207283
matteodallosso	3	170	1.7647059
sbonaccini	33	2884	1.1442441
sfnlcd	9	1308	0.6880734
gianluc_ferrara	3	560	0.5357143
adolfo_urso	7	1966	0.3560529
gualtierieurope	4	1432	0.2793296
MassimoUngaro	3	1135	0.2643172
EugenioGiani	3	1235	0.2429150
pierofassino	3	1255	0.2390438
ecdelre	4	2113	0.1893043
guglielmopicchi	3	3234	0.0927644

3 Dictionary analysis

At the level of political parties, which ones make most use of populist rhetoric?

At the level of individual politicians, which ones make most use of populist rhetoric?

I use 3 dictionary to perform the analysis

- Rooduijn & Pauwels: Rooduijn, M., and T. Pauwels. 2011. "Measuring Populism: Comparing Two Methods of Content Analysis." West European Politics 34 (6): 1272–1283.
- Decadri & Boussalis: Decadri, S., & Boussalis, C. (2020). Populism, party membership, and language complexity in the Italian chamber of deputies. Journal of Elections, Public Opinion and Parties, 30(4), 484-503.
- Grundl: Gründl J. Populist ideas on social media: A dictionary-based measurement of populist communication. New Media & Society. December 2020.
- Decadri & Boussalis + Grundl: this is simply a more extended version of the D&B dictionary, which also contains some terms taken from Grundl.

3.1 Create the dictionary

I imported the excel file with the words for the dictionaries, excluding NA's.

```
# import dictionaries file
dict <-  read_excel("data/populism_dictionaries.xlsx")
variable.names(dict)
```

```
## [1] "Rooduijn_Pauwels_Italian"
```

```
## [2] "Grundl_Italian_adapted"
## [3] "Decadri_Boussalis"
## [4] "Decadri_Boussalis_Grundl_People"
## [5] "Decadri_Boussalis_Grundl_Common Will"
## [6] "Decadri_Boussalis_Grundl_Elite"
```

```
# create the dictionary
Rooduijn_Pauwels_Italian <-
  dictionary(list(populism =
                    (dict$Rooduijn_Pauwels_Italian
                     [!is.na(dict$Rooduijn_Pauwels_Italian)])))

Grundl_Italian_adapted <-
  dictionary(list(populism =
                    dict$Grundl_Italian_adapted
                  [!is.na(dict$Grundl_Italian_adapted)]))

Decadri_Boussalis <-
  dictionary(list(populism =
                    dict$Decadri_Boussalis
                  [!is.na(dict$Decadri_Boussalis)]))

Decadri_Boussalis_Grundl <-
  dictionary(list(people =
                    dict$Decadri_Boussalis_Grundl_People
                  [!is.na(dict$Decadri_Boussalis_Grundl_People)],
                  common_will =
                    dict$`Decadri_Boussalis_Grundl_Common Will`
                  [!is.na(dict$`Decadri_Boussalis_Grundl_Common Will`)],
```

```
                  elite =
                    dict$Decadri_Boussalis_Grundl_Elite
                  [!is.na(dict$Decadri_Boussalis_Grundl_Elite)]))
```

```
dictionaries <- c("Rooduijn_Pauwels_Italian", "Grundl_Italian_adapted",
                  "Decadri_Boussalis","Decadri_Boussalis_Grundl")
n.words <- c(length(Rooduijn_Pauwels_Italian$populism),
             length(Grundl_Italian_adapted$populism),
             length(Decadri_Boussalis$populism),
             (length(Decadri_Boussalis_Grundl$people)+
                length(Decadri_Boussalis_Grundl$common_will)+
                length(Decadri_Boussalis_Grundl$elite))
              )
number_of_words <- data.frame(dictionaries,n.words)
kable(number_of_words)
```

dictionaries	n.words
Rooduijn_Pauwels_Italian	18
Grundl_Italian_adapted	135
Decadri_Boussalis	25
Decadri_Boussalis_Grundl	77

3.1.1 Group and weight the dfm

```
# By party & month
dfm_weigh_p_month <- dfm_group(DFM, groups = interaction(party_id, month))%>%
  dfm_weight(scheme = "prop")
# By party & quarter
dfm_weigh_p_quart <- dfm_group(DFM, groups = interaction(party_id, quarter))%>%
```

```
  dfm_weight(scheme = "prop")
# By party & year
dfm_weigh_p_year <- dfm_group(DFM, groups = interaction(party_id, year))%>%
  dfm_weight(scheme = "prop")
# By party & day
dfm_weigh_p_day <- dfm_group(DFM, groups = interaction(party_id, date))%>%
  dfm_weight(scheme = "prop")
```

Apply dictionary

3.2 Decadri_Boussalis_Grundl

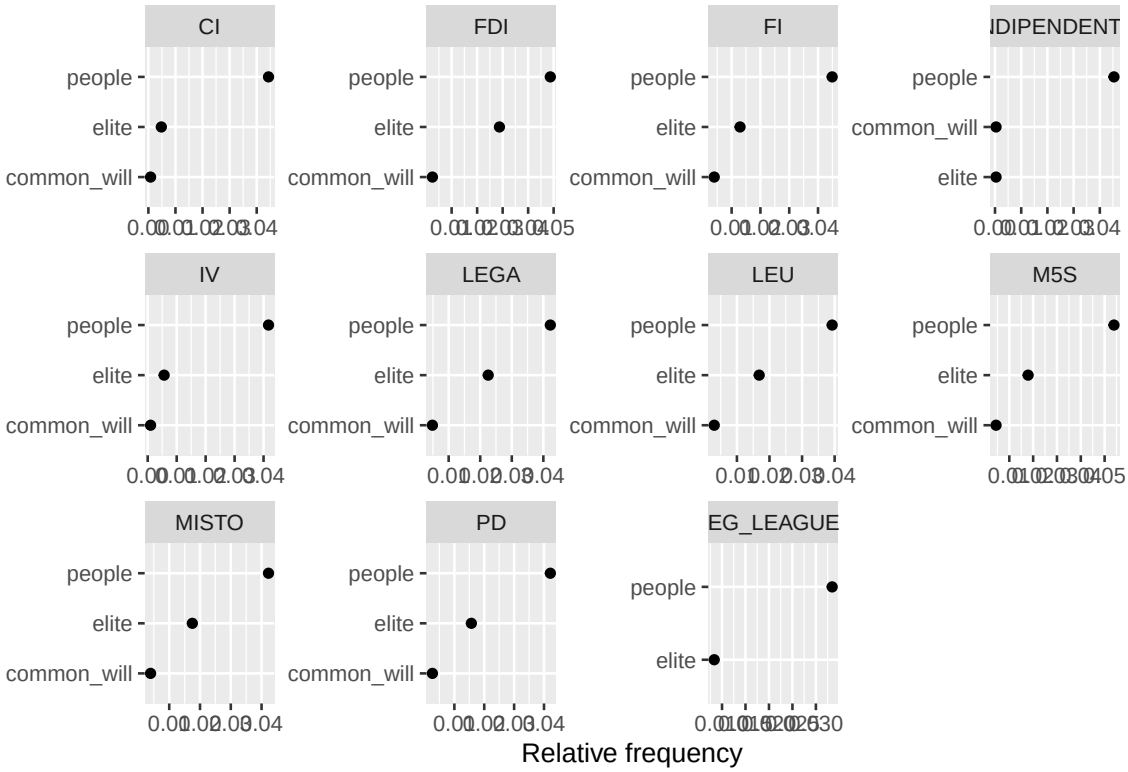
3.2.1 Level of sparsity

daily: 12.08%

weekly: 0.55%

monthly: 0%

```
# Dictionary analysis with Decadri_Boussalis_Grundl
# By quarter
dfm_dict1  <- dfm_lookup(dfm_weigh_p_quart, dictionary = Decadri_Boussalis_Grundl)
# By date
dfm_by_date1 <- dfm_lookup(dfm_weigh_p_day, dictionary = Decadri_Boussalis_Grundl)
```

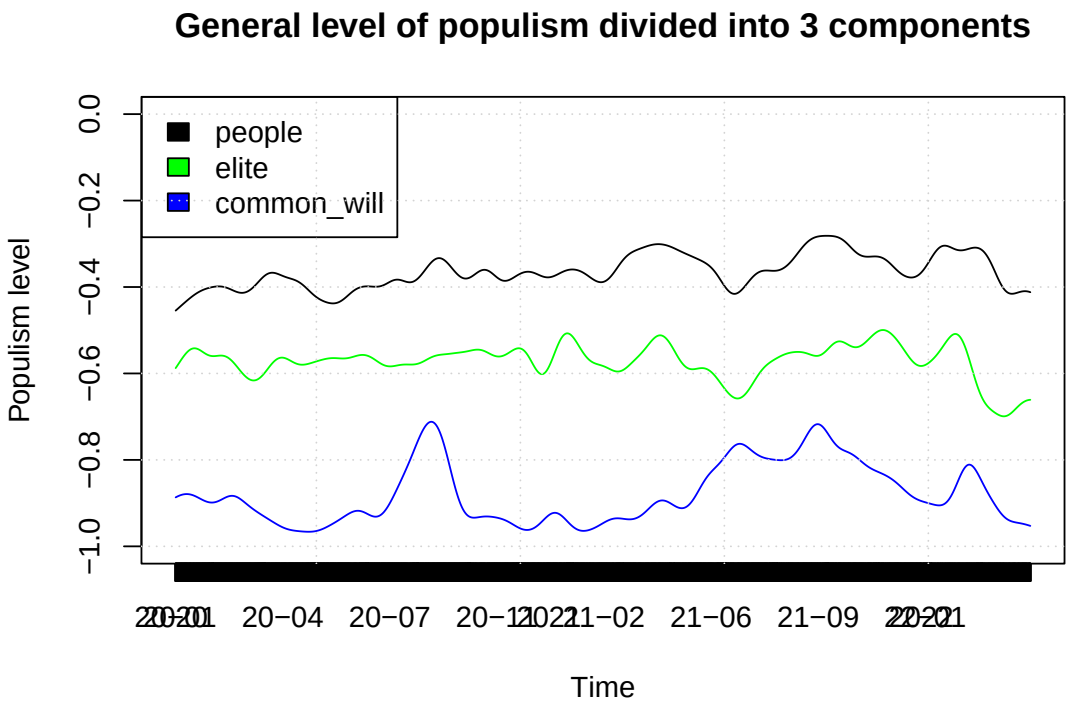


Looking at the populist rhetoric for each party divided into the 3 components people-centrism, anti-elitism and common-will, we note that the most frequent components is People-centrism.

3.2.2 General level of populism in time divided into 3 components

```
dat_smooth1.1 <- ksmooth(x = dfm_by_date1$date,
                         y = dfm_by_date1[,"people"] -
                           !dfm_by_date1[,"people"],
                         kernel = "normal", bandwidth = 30)

dat_smooth1.2 <- ksmooth(x = dfm_by_date1$date,
                         y = dfm_by_date1[,"common_will"] -
                           !dfm_by_date1[,"common_will"],
                         kernel = "normal", bandwidth = 30)

dat_smooth1.3 <- ksmooth(x = dfm_by_date1$date,
                         y = dfm_by_date1[,"elite"] -
                           !dfm_by_date1[,"elite"],
                         kernel = "normal", bandwidth = 30)

plot_time_1 <- plot(dat_smooth1.1$x, dat_smooth1.1$y,
                    type = "l", ylab = "Populism level", xlab = "Time",
                    ylim = c(-1,0), xlim = c(min(dfm_by_date1$date),
                                             max(dfm_by_date1$date)))

  lines(dat_smooth1.2$x, dat_smooth1.2$y,
        type = "l", ylab = "Populism level", xlab = "Time",col = "blue")
```

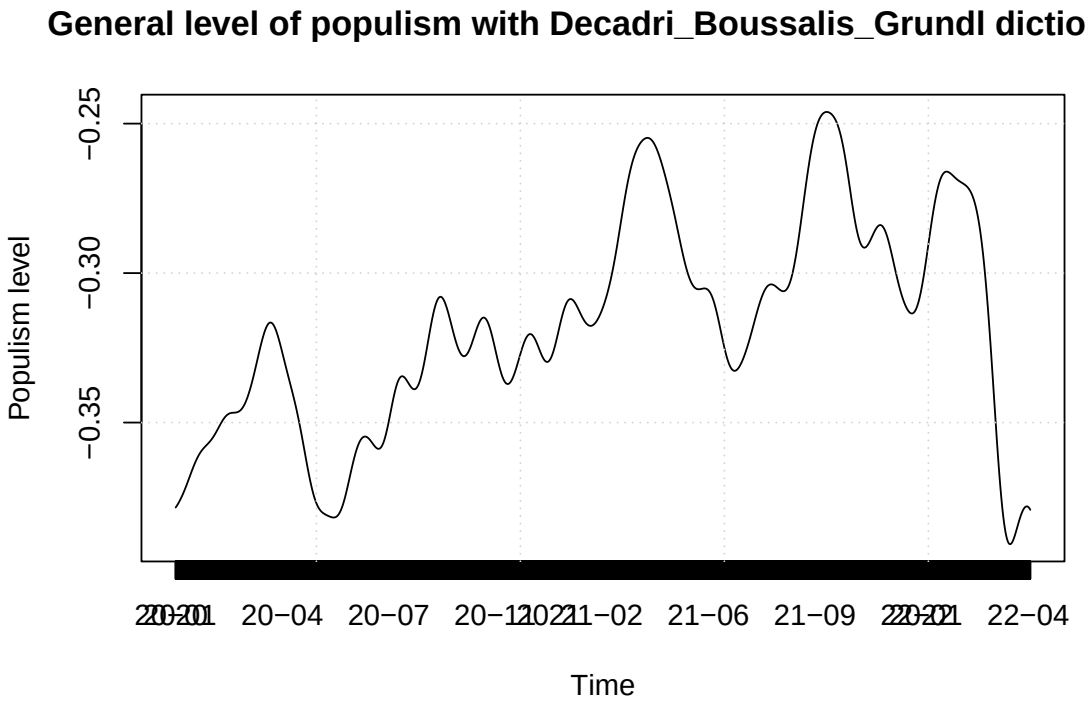
```
lines(dat_smooth1.3$x, dat_smooth1.3$y,
      type = "l", ylab = "Populism level", xlab = "Time", col = "green")


legend("topleft", legend = c("people","elite","common_will"),
       fill = c("black", "green", "blue"))
axis(1, dfm_by_date1$date,
   format(dfm_by_date1$date, "%y-%m"))
grid()
title(main = "General level of populism divided into 3 components")
```



This plot is coherent with the previous one and show us that people is the Component that score better.

3.2.3 General level of populism in time

```
dat_smooth1 <- ksmooth(x = dfm_by_date1$date,
                       y = ((dfm_by_date1[,"people"]+
                             dfm_by_date1[,"common_will"]+
                             dfm_by_date1[,"elite"])/3) -
                         !((dfm_by_date1[,"people"]+
                             dfm_by_date1[,"common_will"]+
                             dfm_by_date1[,"elite"])/3),
                       kernel = "normal", bandwidth = 30)

plot_time_1 <- plot(dat_smooth1$x, dat_smooth1$y,
                    type = "l", ylab = "Populism level", xlab = "Time")
                    axis(1, dat_smooth1$x,
                         format(dat_smooth1$x, "%y-%m"))
                    grid()
title(main = "General level of populism with Decadri_Boussalis_Grundl dictionary")
```



3.2.4 Most populist parliamentary group for each component

```
# Most populist parliamentary group
dfm_dict1_tstat_party <- textstat_frequency(dfm_dict1, groups = party_id)
kable(dfm_dict1_tstat_party %>% filter(feature == "people") %>% slice_max(frequency
```

	feature	frequency	rank	docfreq	group
22	people	0.0539309	1	10	M5S
4	people	0.0486981	1	10	FDI
10	people	0.0454089	1	10	INDIPENDENTE
7	people	0.0449102	1	10	FI
1	people	0.0443528	1	10	CI
25	people	0.0422541	1	10	MISTO
16	people	0.0422111	1	10	LEGA
28	people	0.0421103	1	10	PD
13	people	0.0417078	1	10	IV
19	people	0.0392343	1	10	LEU
31	people	0.0334640	1	9	REG_LEAGUES

```
kable(dfm_dict1_tstat_party %>% filter(feature == "elite") %>% slice_max(frequency,
```

	feature	frequency	rank	docfreq	group
5	elite	0.0287440	2	10	FDI
17	elite	0.0225301	2	10	LEGA
23	elite	0.0178936	2	10	M5S
26	elite	0.0175129	2	10	MISTO
20	elite	0.0168347	2	10	LEU
29	elite	0.0156834	2	10	PD
8	elite	0.0129287	2	10	FI
32	elite	0.0083365	2	10	REG_LEAGUES
14	elite	0.0056664	2	9	IV
2	elite	0.0047789	2	9	CI
12	elite	0.0004012	2	2	INDIPENDENTE

```
kable(dfm_dict1_tstat_party %>% filter(feature == "common_will") %>% slice_max(frequ
```

	feature	frequency	rank	docfreq	group
18	common_will	0.0048497	3	10	LEGA
24	common_will	0.0044336	3	10	M5S
9	common_will	0.0039613	3	10	FI
27	common_will	0.0039426	3	9	MISTO
21	common_will	0.0030362	3	9	LEU
30	common_will	0.0026507	3	10	PD
6	common_will	0.0024601	3	10	FDI
15	common_will	0.0009923	3	4	IV
3	common_will	0.0008098	3	4	CI
11	common_will	0.0004012	2	2	INDIPENDENTE

3.2.5 Most populist parliamentary group

```
# Add variable for the general level of populism
dfm_dict1_tstat_party <- dfm_dict1_tstat_party %>%
  group_by(group) %>% mutate(populism = mean(frequency))
dfm_dict1_tstat_party_filtered <- dfm_dict1_tstat_party %>%
  select(group,populism) %>% unique() %>% arrange(desc(populism))
kable(dfm_dict1_tstat_party_filtered)
```

group	populism
FDI	0.0266340
M5S	0.0254194
LEGA	0.0231970
MISTO	0.0212365
REG_LEAGUES	0.0209002
FI	0.0206001
PD	0.0201481
LEU	0.0197017
CI	0.0166472
IV	0.0161222
INDIPENDENTE	0.0154037

3.3 Rooduijn_Pauwels_Italian

3.3.1 Level of sparsity

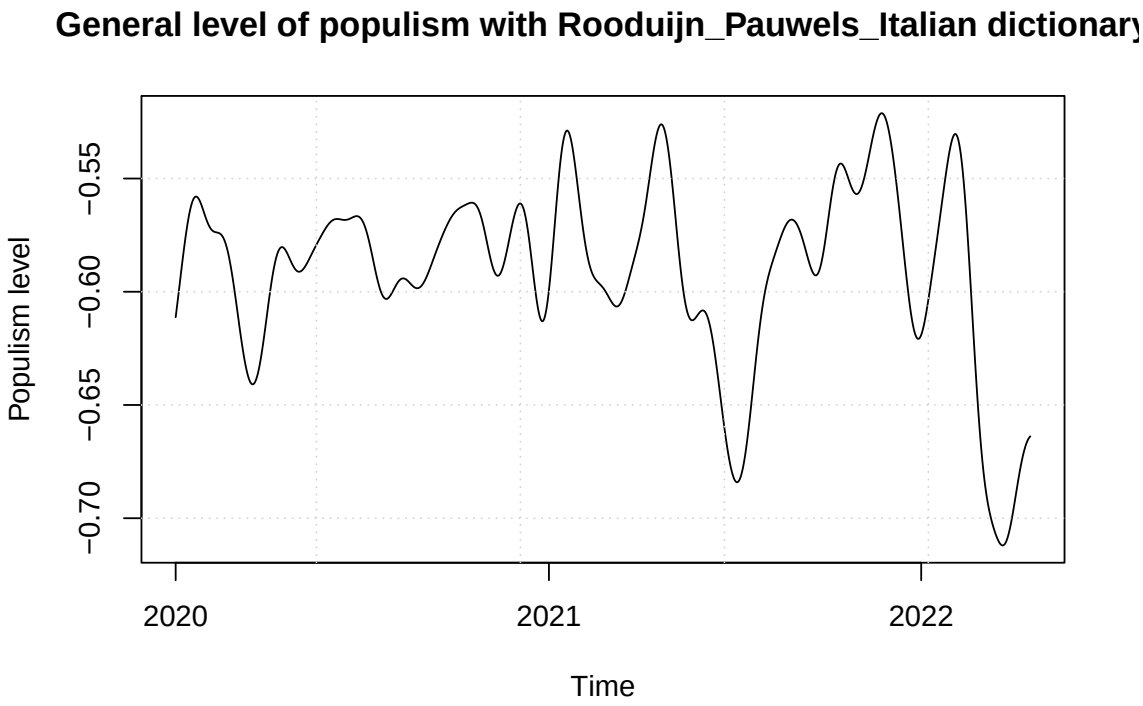
daily: 0.60%

weekly: 0.%

monthly: 0%

```
# Dictionary analysis with Rooduijn_Pauwels_Italian
# By quarter
dfm_dict2  <- dfm_lookup(dfm_weigh_p_quart, dictionary = Rooduijn_Pauwels_Italian)
# Group by date
dfm_by_date2 <- dfm_lookup(dfm_weigh_p_day, dictionary = Rooduijn_Pauwels_Italian)
#dfm_by_date2

# Group by week
#dfm_by_week2 <- dfm_group(dfm_dict2, groups= week)
#dfm_by_week2
# Group by month
#dfm_by_month2 <- dfm_group(dfm_dict2, groups= month)

#kable(dfm_by_month2)
```

3.3.2 General level of populism in time



3.3.3 Most populist parliamentary group

```
# Most populist parliamentary group
dfm_dict2_tstat_party <- textstat_frequency(dfm_dict2, groups = party_id)
kable(dfm_dict2_tstat_party %>% slice_max(frequency, n = 20))
```

	feature	frequency	rank	docfreq	group
2	populism	0.0274375	1	10	FDI
6	populism	0.0215939	1	10	LEGA
7	populism	0.0159693	1	10	LEU
9	populism	0.0156860	1	10	MISTO
10	populism	0.0149336	1	10	PD
8	populism	0.0144550	1	10	M5S
3	populism	0.0116489	1	10	FI
11	populism	0.0083365	1	10	REG_LEAGUES
5	populism	0.0056664	1	9	IV
1	populism	0.0040565	1	9	CI
4	populism	0.0002734	1	1	INDIPENDENTE

3.3.4 Distribution of parliamentary group populism

```
summary(dfm_dict2_tstat_party$frequency)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0002734 0.0070015 0.0144550 0.0127324 0.0158277 0.0274375
```

```
#TBD
hist(dfm_dict2_tstat_party$frequency, prob=T)
points(density(dfm_dict2_tstat_party$frequency),type="l",col="blue")
rug(dfm_dict2_tstat_party$frequency,col="red")

m <- mean(dfm_dict2_tstat_party$frequency)
std<-sqrt(var(dfm_dict2_tstat_party$frequency))

hist(dfm_dict2_tstat_party$frequency,prob=T,main="Frequency")
```

```
curve(dnorm(x, mean=m, sd=std), col="darkblue", lwd=2, add=TRUE)
```

```
# above the median
kable(dfm_dict2_tstat_party  %>% filter(frequency > median(frequency)))
```

	feature	frequency	rank	docfreq	group
2	populism	0.0274375	1	10	FDI
6	populism	0.0215939	1	10	LEGA
7	populism	0.0159693	1	10	LEU
9	populism	0.0156860	1	10	MISTO
10	populism	0.0149336	1	10	PD

```
# above the mean
kable(dfm_dict2_tstat_party %>% filter(frequency > mean(frequency)))
```

	feature	frequency	rank	docfreq	group
2	populism	0.0274375	1	10	FDI
6	populism	0.0215939	1	10	LEGA
7	populism	0.0159693	1	10	LEU
8	populism	0.0144550	1	10	M5S
9	populism	0.0156860	1	10	MISTO
10	populism	0.0149336	1	10	PD

```
# below the first quantiles
kable(dfm_dict2_tstat_party %>% filter(frequency < 0.0070015))
```

	feature	frequency	rank	docfreq	group
1	populism	0.0040565	1	9	CI
4	populism	0.0002734	1	1	INDIPENDENTE
5	populism	0.0056664	1	9	IV

```
# above the third quantiles

kable(dfm_dict2_tstat_party %>% filter(frequency > 0.0158277))
```

	feature	frequency	rank	docfreq	group
2	populism	0.0274375	1	10	FDI
6	populism	0.0215939	1	10	LEGA
7	populism	0.0159693	1	10	LEU

3.4 Grundl_Italian_adapted

3.4.1 Level of sparsity

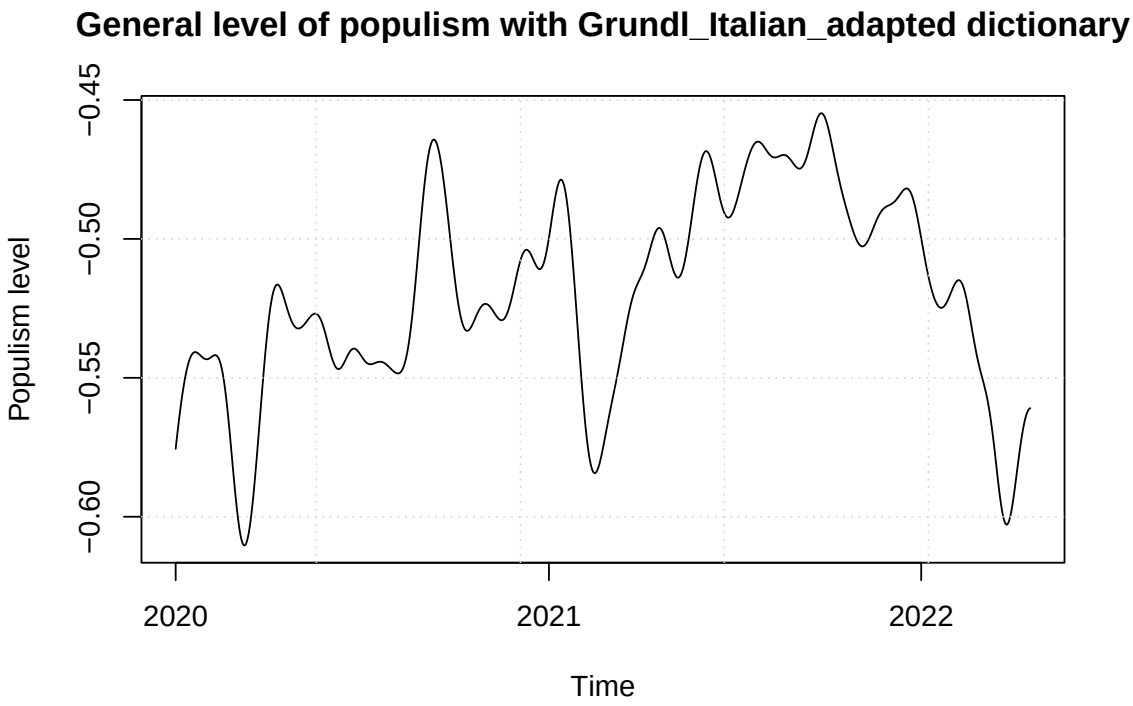
daily: 0.24%

weekly: 0.0%

monthly: 0%

```
# Dictionary analysis with Grundl_Italian_adapted
# By quarter
dfm_dict3  <- dfm_lookup(dfm_weigh_p_quart, dictionary = Grundl_Italian_adapted)
# Group by date
dfm_by_date3 <- dfm_lookup(dfm_weigh_p_day, dictionary = Grundl_Italian_adapted)
#dfm_by_date2
```

3.4.2 General level of populism in time



3.4.3 Most populist parliamentary group

```
# Most populist parliamentary group
dict_3_tstat_party <- textstat_frequency(dfm_dict3, groups = party_id)


kable(dict_3_tstat_party %>% slice_max(frequency, n = 20))
```

	feature	frequency	rank	docfreq	group
2	populism	0.0254825	1	10	FDI
8	populism	0.0231987	1	10	M5S
9	populism	0.0227410	1	10	MISTO
6	populism	0.0224236	1	10	LEGA
3	populism	0.0193174	1	10	FI
7	populism	0.0188197	1	10	LEU
10	populism	0.0174278	1	10	PD
1	populism	0.0160156	1	10	CI
11	populism	0.0108807	1	10	REG_LEAGUES
4	populism	0.0105241	1	8	INDIPENDENTE
5	populism	0.0080644	1	9	IV

3.4.4 Distribution of parliamentary grouppopulism

```
#TBD
summary(dict_3_tstat_party$frequency)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.008064 0.013448 0.018820 0.017718 0.022582 0.025483
```

```
# above the median
kable(dict_3_tstat_party  %>% filter(frequency > median(frequency)))
```

	feature	frequency	rank	docfreq	group
2	populism	0.0254825	1	10	FDI
3	populism	0.0193174	1	10	FI
6	populism	0.0224236	1	10	LEGA
8	populism	0.0231987	1	10	M5S
9	populism	0.0227410	1	10	MISTO

```
# above the mean

kable(dict_3_tstat_party %>% filter(frequency > mean(frequency)))
```

	feature	frequency	rank	docfreq	group
2	populism	0.0254825	1	10	FDI
3	populism	0.0193174	1	10	FI
6	populism	0.0224236	1	10	LEGA
7	populism	0.0188197	1	10	LEU
8	populism	0.0231987	1	10	M5S
9	populism	0.0227410	1	10	MISTO

```
# below the first quantiles

kable(dict_3_tstat_party %>% filter(frequency < 0.013448))
```

	feature	frequency	rank	docfreq	group
4	populism	0.0105241	1	8	INDIPENDENTE
5	populism	0.0080644	1	9	IV
11	populism	0.0108807	1	10	REG_LEAGUES

```
# above the third quantiles

kable(dict_3_tstat_party %>% filter(frequency > 0.022582 ))
```

	feature	frequency	rank	docfreq	group
2	populism	0.0254825	1	10	FDI
8	populism	0.0231987	1	10	M5S
9	populism	0.0227410	1	10	MISTO

3.5 Decadri_Boussalis

3.5.1 Level of sparsity

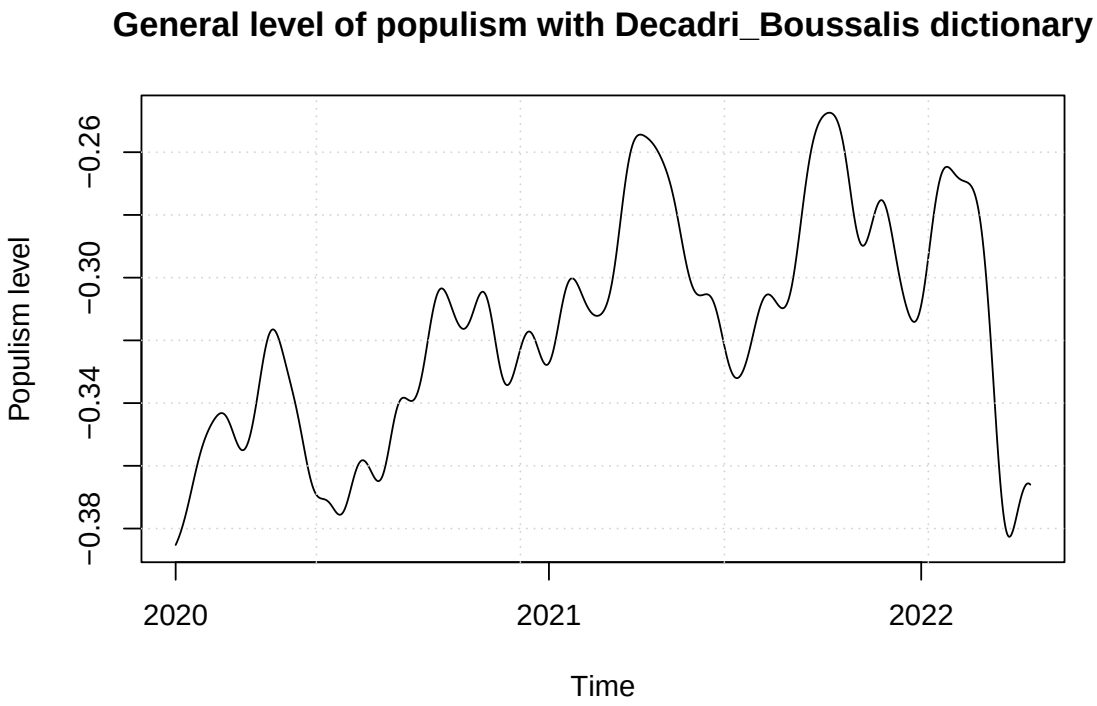
daily: 0%

weekly: 0.0%

monthly: 0%

```
# Dictionary analysis with Decadri_Boussalis
# By quarter
dfm_dict4  <- dfm_lookup(dfm_weigh_p_quart, dictionary = Decadri_Boussalis)
# By date
dfm_by_date4 <- dfm_lookup(dfm_weigh_p_day, dictionary = Decadri_Boussalis)
```

3.5.2 General level of populism in time



3.5.3 Most populist parliamentary group

```
# Most populist parliamentary group
dict_4_tstat_party <- textstat_frequency(dfm_dict4, groups = party_id)


kable(dict_4_tstat_party %>% slice_max(frequency, n = 20))
```

	feature	frequency	rank	docfreq	group
2	populism	0.0781517	1	10	FDI
8	populism	0.0696170	1	10	M5S
6	populism	0.0656655	1	10	LEGA
3	populism	0.0600724	1	10	FI
9	populism	0.0595844	1	10	MISTO
7	populism	0.0588181	1	10	LEU
10	populism	0.0586910	1	10	PD
1	populism	0.0499145	1	10	CI
5	populism	0.0479199	1	10	IV
4	populism	0.0464945	1	10	INDIPENDENTE
11	populism	0.0435615	1	10	REG_LEAGUES

3.5.4 Distribution of party populism

```
#TBD
summary(dict_4_tstat_party$frequency)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.04356 0.04892 0.05882 0.05804 0.06287 0.07815
```

```
# above the median
kable(dict_4_tstat_party  %>% filter(frequency > median(frequency)))
```

	feature	frequency	rank	docfreq	group
2	populism	0.0781517	1	10	FDI
3	populism	0.0600724	1	10	FI
6	populism	0.0656655	1	10	LEGA
8	populism	0.0696170	1	10	M5S
9	populism	0.0595844	1	10	MISTO

```
# above the mean
kable(dict_4_tstat_party %>% filter(frequency > mean(frequency)))
```

	feature	frequency	rank	docfreq	group
2	populism	0.0781517	1	10	FDI
3	populism	0.0600724	1	10	FI
6	populism	0.0656655	1	10	LEGA
7	populism	0.0588181	1	10	LEU
8	populism	0.0696170	1	10	M5S
9	populism	0.0595844	1	10	MISTO
10	populism	0.0586910	1	10	PD

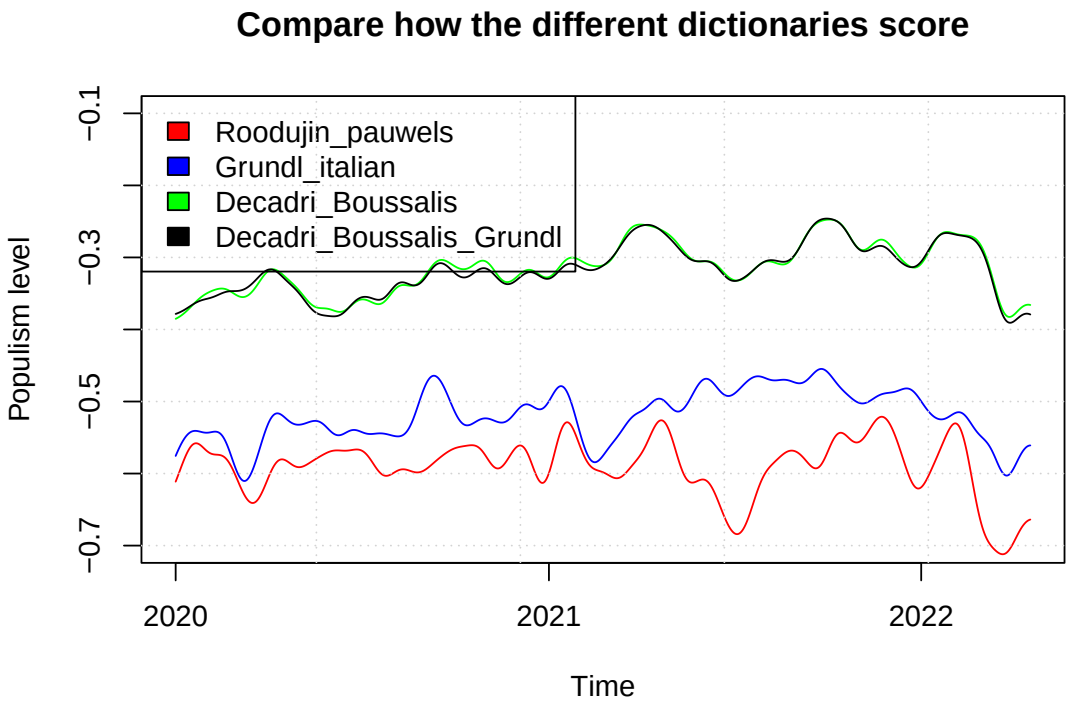
```
# below the first quantiles
kable(dict_4_tstat_party %>% filter(frequency < 0.04892 ))
```

	feature	frequency	rank	docfreq	group
4	populism	0.0464945	1	10	INDIPENDENTE
5	populism	0.0479199	1	10	IV
11	populism	0.0435615	1	10	REG_LEAGUES

```
# above the third quantiles
kable(dict_4_tstat_party %>% filter(frequency > 0.06287  ))
```

	feature	frequency	rank	docfreq	group
2	populism	0.0781517	1	10	FDI
6	populism	0.0656655	1	10	LEGA
8	populism	0.0696170	1	10	M5S

3.6 Compare the general level of populism in time for the dictionaries



3.7 Compare how the dictionaries score for the most populist parliamentary group

```
# Create the columns with the "populist score"
# 11 for the "most populist" and 1 for the least
dfm_dict1_tstat_party_filtered$my_rank <- rank(dfm_dict1_tstat_party_filtered$populi
dfm_dict2_tstat_party$my_rank <- rank(dfm_dict2_tstat_party$frequency)
dict_3_tstat_party$my_rank <- rank(dict_3_tstat_party$frequency)
dict_4_tstat_party$my_rank <- rank(dict_4_tstat_party$frequency)

# define the parliamentary group list
party <- c("LEGA","PD","M5S","FI","FDI","MISTO",
           "LEU","CI","IV","INDIPENDENTE","REG_LEAGUES")
# create an empty df
party_rank <- data.frame(first = vector(), second = vector(),
                         third = vector(), fourth = vector(), fifth = vector() )

# loop the rank for each parliamentary group
for (i in party)
{
  rank_dict_1 <- (dfm_dict1_tstat_party_filtered %>% filter(group == i ) %>% .$my_ra
  rank_dict_2 <- (dfm_dict2_tstat_party %>% filter(group == i ) %>% .$my_rank)
  rank_dict_3 <- (dict_3_tstat_party %>% filter(group == i ) %>% .$my_rank)
  rank_dict_4 <- (dict_4_tstat_party %>% filter(group == i ) %>% .$my_rank)

  party <- (i)
  party_rank <- rbind(party_rank, cbind(party, rank_dict_1, rank_dict_2,
                                        rank_dict_3, rank_dict_4))
```

```
}


# change the format of the columns in numeric
party_rank$rank_dict_1 <- as.numeric(party_rank$rank_dict_1)
party_rank$rank_dict_2 <- as.numeric(party_rank$rank_dict_2)
party_rank$rank_dict_3 <- as.numeric(party_rank$rank_dict_3)
party_rank$rank_dict_4 <- as.numeric(party_rank$rank_dict_4)

# Create the column with the sum of the single score
party_rank$total_score <- rowSums(party_rank[,-1])
kable(party_rank %>% arrange(desc(total_score)))
```

party	rank_dict_1	rank_dict_2	rank_dict_3	rank_dict_4	total_score
FDI	11	11	11	11	44
LEGA	9	10	8	9	36
M5S	10	6	10	10	36
MISTO	8	8	9	7	32
FI	6	5	7	8	26
LEU	4	9	6	6	25
PD	5	7	5	5	22
REG_LEAGUES	7	4	3	1	15
CI	3	2	4	4	13
IV	2	3	1	3	9
INDIPENDENTE	1	1	2	2	6

4 Sentiment analysis

http://saifmohammad.com/WebPages/lexicons.html

4.1 Inspect the dictionary

```
head(get_sentiment_dictionary(dictionary = "nrc", language = "italian"),15)
```

```
##       lang          word sentiment value
## 1  italian          abba  positive     1
## 2  italian     capacità  positive     1
## 3  italian sopra citato  positive     1
## 4  italian     assoluto  positive     1
## 5  italian  assoluzione  positive     1
## 6  italian    assorbito  positive     1
## 7  italian   abbondanza  positive     1
## 8  italian   abbondante  positive     1
## 9  italian   accademico  positive     1
## 10 italian    accademia  positive     1
## 11 italian  accettabile  positive     1
## 12 italian accettazione  positive     1
## 13 italian  accessibile  positive     1
## 14 italian      encomio  positive     1
## 15 italian     alloggio  positive     1
```

4.1.1 Clean text from dataframe

Define function to make the text extracted from dataframe suitable for analysis

```
# Define function to make the text suitable for analysis
clean.text = function(x)
{
  # tolower
  x = tolower(x)
  # remove rt
  x = gsub("rt", "", x)
  # remove at
  x = gsub("@\\w+", "", x)
  # remove punctuation
  x = gsub("[[:punct:]]", "", x)
  # remove numbers
  x = gsub("[[:digit:]]", "", x)
  # remove links http
  x = gsub("http\\w+", "", x)
  # remove tabs
  x = gsub("[ |\t]{2,}", "", x)
  # remove blank spaces at the beginning
  x = gsub("^ ", "", x)
  # remove blank spaces at the end
  x = gsub(" $", "", x)
  return(x)
}
```

4.2 Create the filtered dataframes

```r
# Create filtered dataframes
MELONI <- dataset %>% filter(nome  == "MELONI Giorgia")
CONTE <- dataset %>% filter(nome  == "CONTE Giuseppe")
RENZI <- dataset %>% filter(nome  == "RENZI Matteo")
SALVINI <- dataset %>% filter(nome  == "SALVINI Matteo")
LETTA <- dataset %>% filter(nome  == "LETTA Enrico")
BERLUSCONI <- dataset %>% filter(nome  == "BERLUSCONI Silvio")
SPERANZA <- dataset %>% filter(nome  == "SPERANZA Roberto")
```

4.3 Create nrc objects

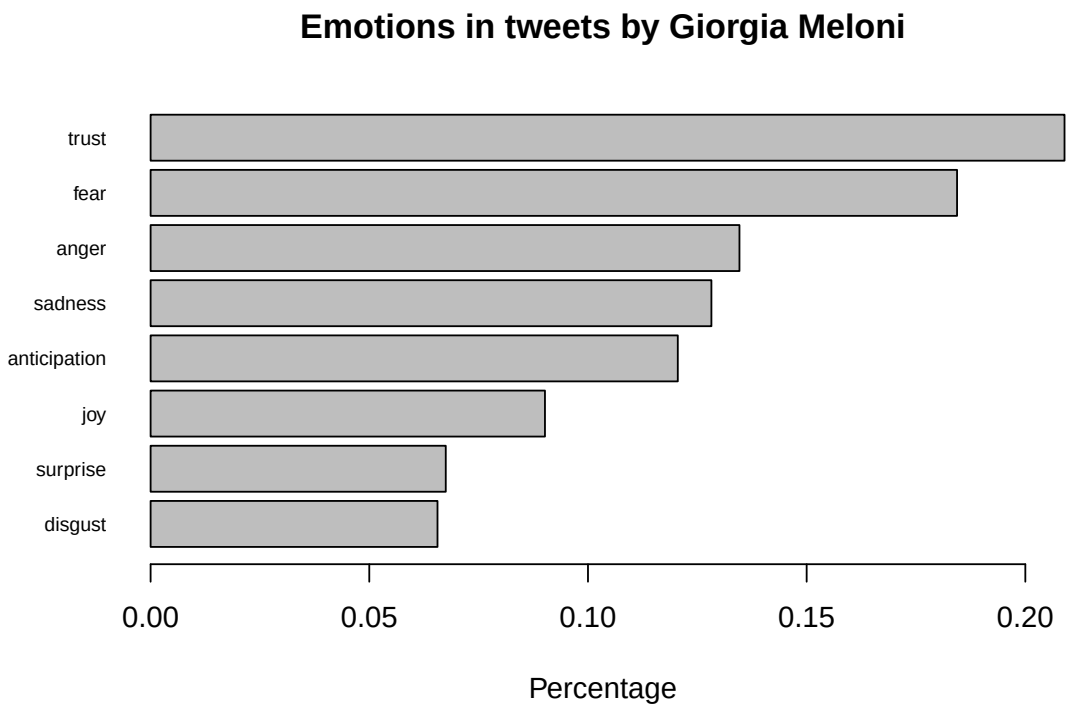
```r
# Create the nrc object
nrc_meloni <- get_nrc_sentiment(MELONI$tweet_testo, language="italian")
save(nrc_meloni,file="data/nrc_meloni.Rda")

nrc_conte <- get_nrc_sentiment(CONTE$tweet_testo, language="italian")
save(nrc_conte,file="data/nrc_conte.Rda")

nrc_renzi <- get_nrc_sentiment(RENZI$tweet_testo, language="italian")
save(nrc_renzi,file="data/nrc_renzi.Rda")

nrc_salvini <- get_nrc_sentiment(SALVINI$tweet_testo, language="italian")
save(nrc_salvini,file="data/nrc_salvini.Rda")

nrc_letta <- get_nrc_sentiment(LETTA$tweet_testo, language="italian")
save(nrc_letta,file="data/nrc_letta.Rda")

nrc_berlusconi <- get_nrc_sentiment(BERLUSCONI$tweet_testo, language="italian")
```

```
save(nrc_berlusconi, file="data/nrc_berlusconi.Rda")


nrc_speranza <- get_nrc_sentiment(SPERANZA$tweet_testo, language="italian")
save(nrc_speranza,file="data/nrc_speranza.Rda")
```

4.4 Giorgia Meloni

4.4.1 Proportion of the emotion

```
barplot(
  sort(colSums(prop.table(nrc_meloni[, 1:8]))),
  horiz = TRUE,
  cex.names = 0.7,
  las = 1,
  main = "Emotions in tweets by Giorgia Meloni", xlab="Percentage"
)
```



4.4.2 Wordcloud of emotions

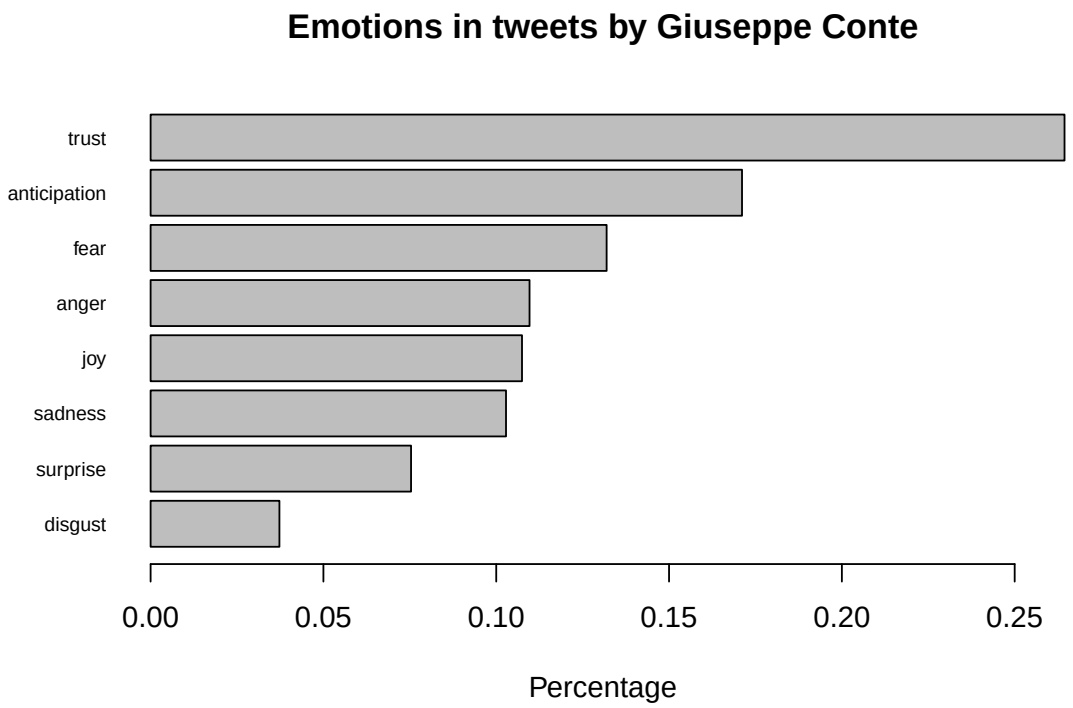
```
all <- c(
  paste(MELONI$tweet_testo[nrc_meloni$anger > 0], collapse=" "),
  paste(MELONI$tweet_testo[nrc_meloni$anticipation > 0], collapse=" "),
  paste(MELONI$tweet_testo[nrc_meloni$disgust > 0], collapse=" "),
  paste(MELONI$tweet_testo[nrc_meloni$fear > 0], collapse=" "),
  paste(MELONI$tweet_testo[nrc_meloni$joy > 0], collapse=" "),
  paste(MELONI$tweet_testo[nrc_meloni$sadness > 0], collapse=" "),
  paste(MELONI$tweet_testo[nrc_meloni$surprise > 0], collapse=" "),
  paste(MELONI$tweet_testo[nrc_meloni$trust > 0], collapse=" ")
)
# Call the function previously defined
all <- clean.text(all)
# remove stop-words
all = removeWords(all,  c(stopwords("italian")))
# create corpus
corpus_viz <- Corpus(VectorSource(all))
# create term-document matrix
tdm <- TermDocumentMatrix(corpus_viz)
# convert as matrix
tdm <- as.matrix(tdm)
# add column names
colnames(tdm) <- c('anger', 'anticipation', 'disgust', 'fear',
                   'joy', 'sadness', 'surprise', 'trust')
# Plot comparison wordcloud
layout(matrix(c(1, 2), nrow=2), heights=c(1, 4))
par(mar=rep(0, 4))
```

```
plot.new()
text(x=0.5, y=0.5, 'Emotion Comparison Word Cloud for tweets by Giorgia Meloni')
comparison.cloud(tdm, random.order=FALSE,
                 colors = c("#00B2FF", "red", "#FF0099",
                            "#6600CC", "green", "orange", "blue", "brown"),
                 title.size=1.5, max.words=80)
```

Emotion Comparison Word Cloud for tweets by Giorgia Meloni

4.5 Giuseppe Conte

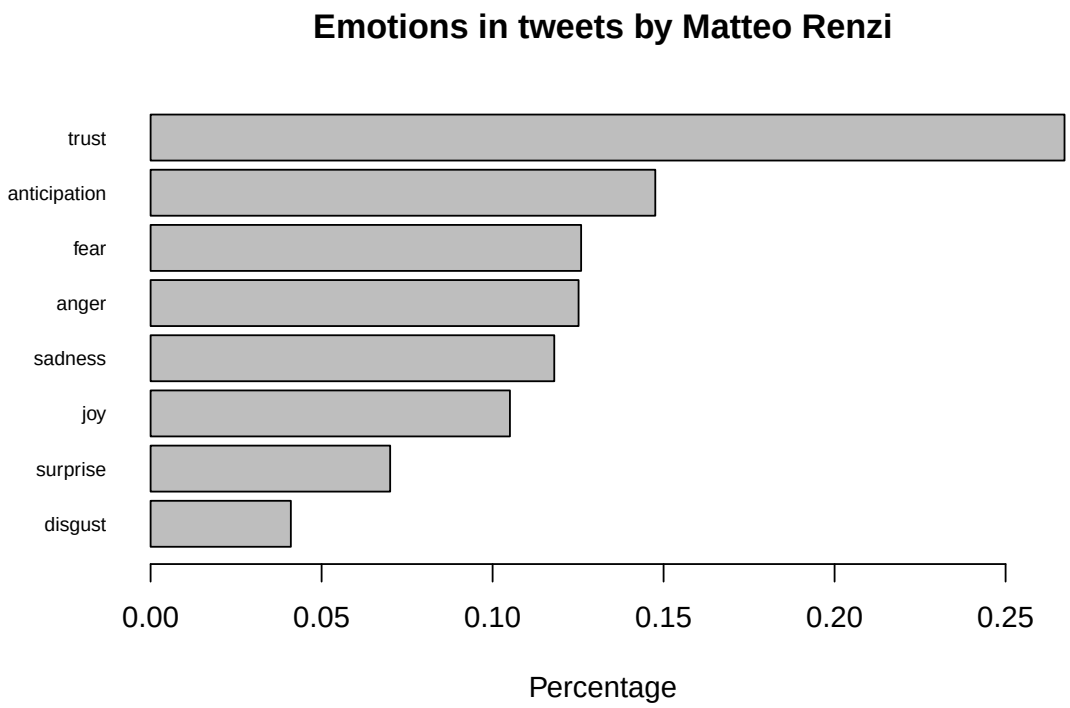
4.5.1 Proportion of the emotion



4.5.2 Wordcloud of emotions

4.6 Matteo Renzi

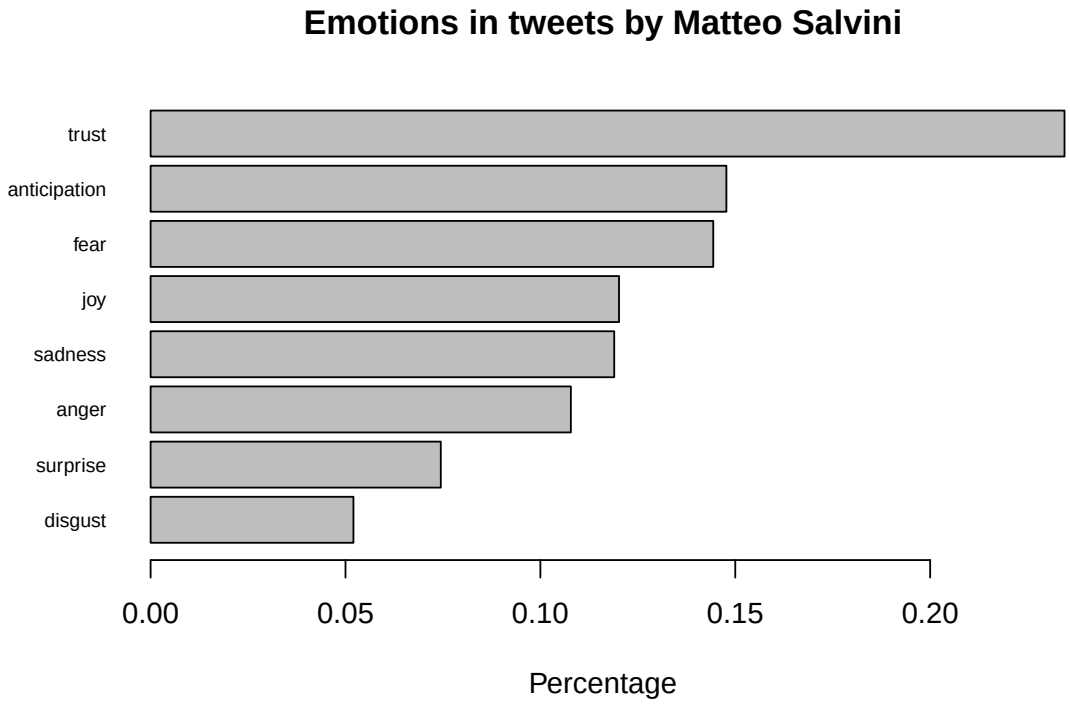
4.6.1 Proportion of the emotion



4.6.2 Wordcloud of emotions

4.7 Matteo Salvini

4.7.1 Proportion of the emotion

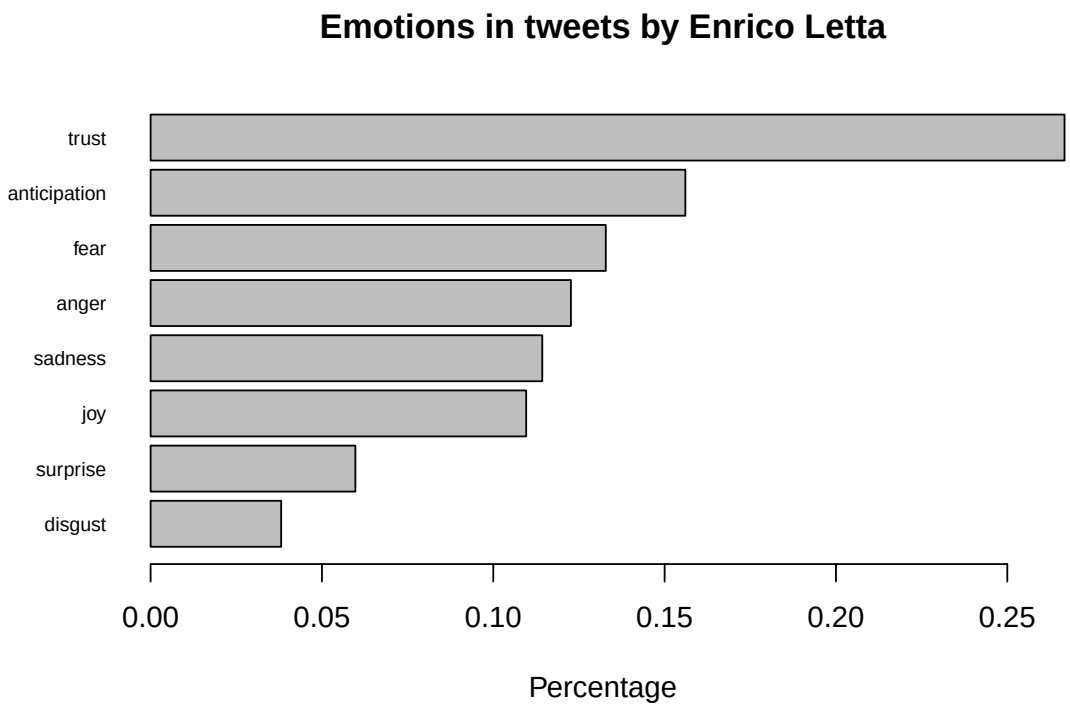


4.7.2 Wordcloud of emotions

Emotion Comparison Word Cloud for tweets by Matteo Salvini

4.8 Enrico Letta

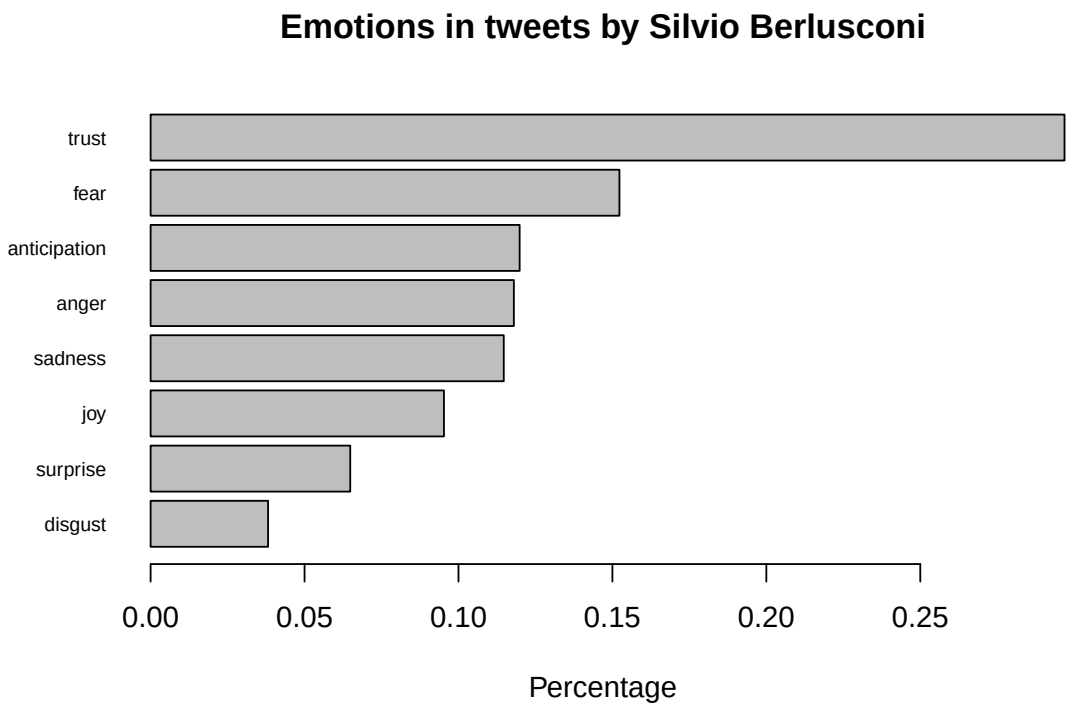
4.8.1 Proportion of the emotion



4.8.2 Wordcloud of emotions

4.9 Silvio Berlusconi

4.9.1 Proportion of the emotion

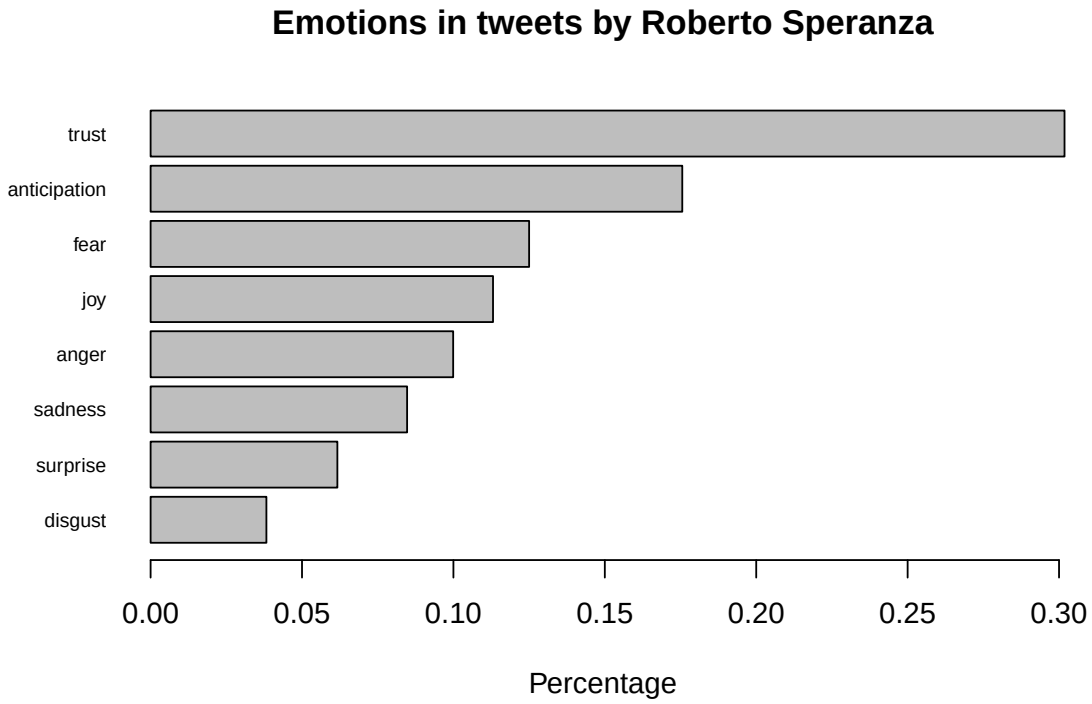


4.9.2 Wordcloud of emotions

Emotion Comparison Word Cloud for tweets by Silvio Berlusconi

4.10 Roberto Speranza

4.10.1 Proportion of the emotion



4.10.2 Wordcloud of emotions

5 LDA Topic model analysis

5.1 CREATE THE DTM

5.1.1 Remove all the account's mentions

```
DFM_trimmed@Dimnames$features <- gsub("^@", "", DFM_trimmed@Dimnames$features)
```

5.1.2 Convert the Document Feature Matrix (Dfm) in a Topic Model (Dtm)

```
dtm <- quanteda::convert(DFM_trimmed, to = "topicmodels")
```

5.2 FIND THE BEST NUMBER OF TOPICS K

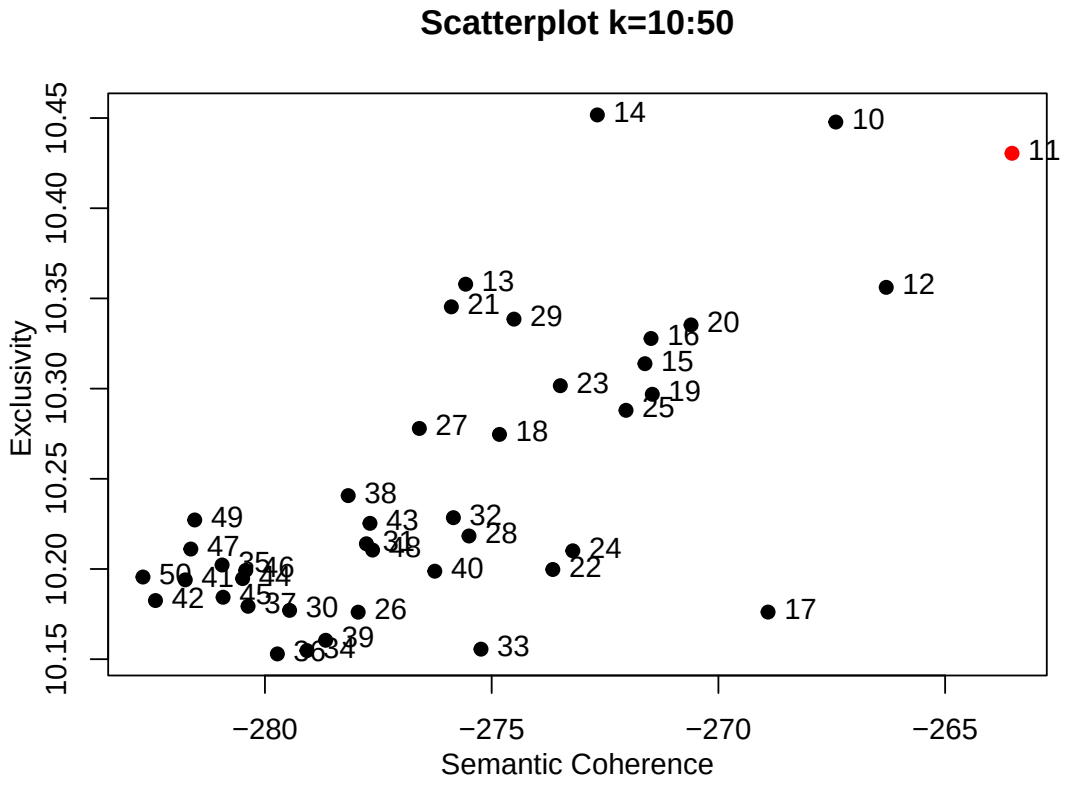
5.2.1 Search the best number of Topics comparing coherence and exclusivity values

K = 10:50

```
# 10 : 50 iter 1000
top1 <- c(10:50)
## Create an empty data frame
risultati <- data.frame(first=vector(), second=vector(), third=vector())
#Run the loop searching the best k value
system.time(
  for (i  in top1)
  {
    set.seed(123)
```

```
    lda_test <- LDA(dtm, method= "Gibbs", k = (i),
                    control=list(verbose=50L, iter=1000))
    topic <- (i)
    coherence_test <- mean(topic_coherence(lda_test, dtm))
    exclusivity_test <- mean(topic_exclusivity(lda_test))
    risultati <- rbind(risultati, cbind(topic, coherence_test, exclusivity_test))
  }
)
# save(risultati,file="data/results_K_10-50.Rda")
```

5.2.2 Plot the values of coherence and exclusivity in order to find the best K



K= 11 has the best values of coherence and exclusivity.

5.3 ANALISYS OF THE TOPICS

5.3.1 Run the analysis selecting k = 11

```
system.time(lda_11 <- LDA(dtm, method= "Gibbs", k = 11,
                          control = list(seed = 123)))
# save(lda_11, file = "data/lda_k_11.Rda")
```

5.3.2 The most important terms from the model, for each topic

Top terms 01	Top terms 02	Top terms 03	Top terms 04	Top terms 05	Top terms 06
governo	#coronavirus	diretta	forza_italia	solidarietà	grande
italiani	#covid19	roma	bene	libertà	grazie
lega	sicurezza	domani	cose	parole	mondo
conte	covid	città	vero	diritti	italia
matteosalvinimi	scuola	amici	davvero	rispetto	forza
fratelliditalia	salute	sindaco	ragione	democrazia	de
salvini	pandemia	insieme	problema	diritto	italiano
pd	pass	sera	ce	violenza	italiana
#lega	dati	territorio	parla	guerra	storia
casa	green	parlare	dovrebbe	popolo	l'italia

Top terms 7	Top terms 8	Top terms 9	Top terms 10	Top terms 11
lavoro	via	governo	anni	piano
paese	ministro	imprese	grazie	nazionale
presidente	legge	milioni	donne	futuro
politica	parlamento	euro	giornata	importante
buon	commissione	lavoratori	auguri	sociale
l'italia	senato	famiglie	vittime	giovani
momento	mov5stelle	sostegno	famiglia	europea
bene	voto	cittadini	comunità	fondamentale
pdnetwork	camera	misure	servizio	paesi
insieme	intervento	crisi	forze	intervista

5.3.3 Interpret the terms

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
titles_11	1?	PANDEMIA	3?	4?	DIRITTI	NAZIONE
	governo	#coronavirus	diretta	forza_italia	solidarietà	grande
	italiani	#covid19	roma	bene	libertà	grazie
	lega	sicurezza	domani	cose	parole	mondo
	conte	covid	città	vero	diritti	italia
	matteosalvinimi	scuola	amici	davvero	rispetto	forza
	fratelliditalia	salute	sindaco	ragione	democrazia	de
	salvini	pandemia	insieme	problema	diritto	italiano
	pd	pass	sera	ce	violenza	italiana
	#lega	dati	territorio	parla	guerra	storia
	casa	green	parlare	dovrebbe	popolo	l'italia

	Topic 7	Topic 8	Topic 9	Topic 10	Topic 11
titles_11	7?	PARLAMENTO	ECONOMIA	RICORRENZE	PNRR
	lavoro	via	governo	anni	piano
	paese	ministro	imprese	grazie	nazionale
	presidente	legge	milioni	donne	futuro
	politica	parlamento	euro	giornata	importante
	buon	commissione	lavoratori	auguri	sociale
	l'italia	senato	famiglie	vittime	giovani
	momento	mov5stelle	sostegno	famiglia	europea
	bene	voto	cittadini	comunità	fondamentale
	pdnetwork	camera	misure	servizio	paesi
	insieme	intervento	crisi	forze	intervista

6 FER: Facial Emotion Recognition Analysis

6.1 Report on the analysis made with FER Python package

The package use the FER-2013 dataset created by Pierre Luc Carrier and Aaron Courville.

The dataset was created using the Google image search API to search for images of faces that match a set of 184 emotion-related keywords like "blissful", "enraged," etc. These keywords were combined with words related to gender, age or ethnicity, to obtain nearly 600 strings which were used as facial image search queries. The first 1000 images returned for each query were kept for the next stage of processing. OpenCV face recognition was used to obtain bounding boxes around each face in the collected images. Human labelers than rejected incorrectly labeled images, corrected the cropping if necessary, and filtered out some duplicate images. Approved, cropped images were then resized to 48x48 pixels and converted to grayscale. Mehdi Mirza and Ian Goodfellow prepared a subset of the images for this contest,and mapped the fine-grained emotion keywords into the same seven broad categories used in the Toronto Face Database [Joshua Susskind, Adam Anderson, and Geoffrey E. Hinton. The Toronto face dataset. Technical Report UTML TR 2010-001, U. Toronto, 2010.]. The resulting dataset contains 35887 images, with 4953 "Anger" images, 547 "Disgust" images, 5121 "Fear" images, 8989 "Happiness" images, 6077 "Sadness" images, 4002 "Surprise" images, and 6198 "Neutral" images. FER-2013 could theoretical suffer from label errors due to the way it was collected, but Ian Goodfellow found that human accuracy on FER-2013 was 65±5%.

66% ACCURACY REPORTED BY OCTAVIO ARRIAGA, Matias Valdenegro-Toro, Paul Plöger (Real-time Convolutional Neural Networks for Emotion and Gender Classification)