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 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

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

1.5.1 Check the number of trimesters

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

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

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

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

1.6 Create the year variables

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

1.6.1 Check the number of years

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

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

```r
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
```

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

1.8 Check that the variables make sense

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

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

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

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

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

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

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

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

1.8.1 Adjust the variable genere

```r
# 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")
```

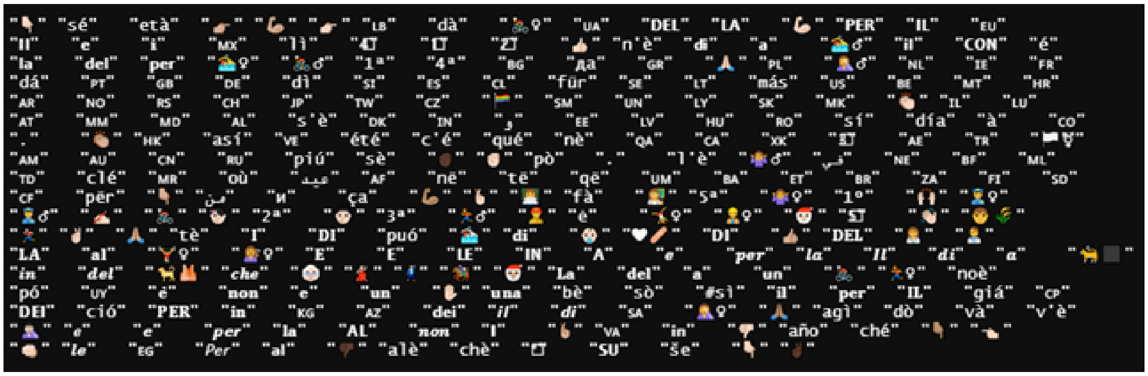


Figure 1: Emoji removed

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_id?
n_parl_party <- dataset %>% select(party_id, nome) %>% group_by(party_id) %>% unique
kable(n_parl_party)
```

party_id	n
CI	17
FDI	39
FI	96
INDIPENDENTE	6
IV	5
LEGA	134
LEU	15
M5S	197
MISTO	71
PD	144
REG_LEAGUES	7

```
# Gender composition
n_gender <- dataset %>% select(genere, nome) %>% group_by(genere) %>% unique() %>% c
kable(n_gender)
```

genere	n
female	258
male	472

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

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

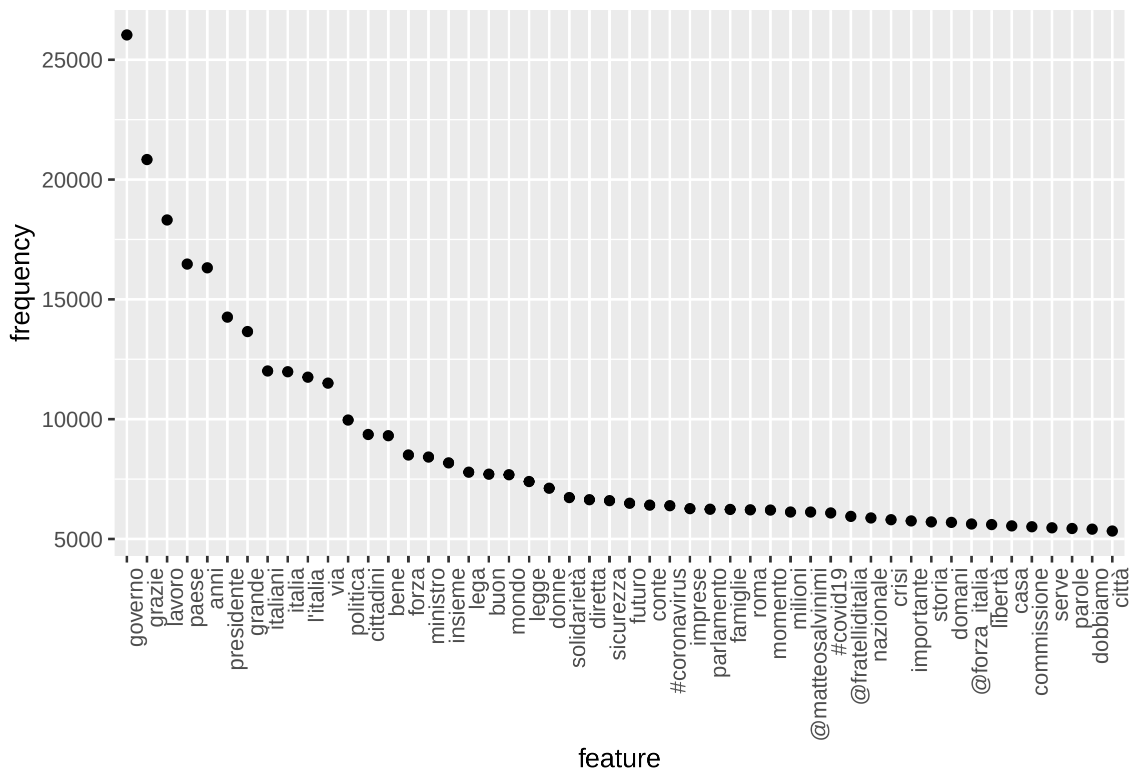
```
min(tw$date)
```

```
## [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'))
```

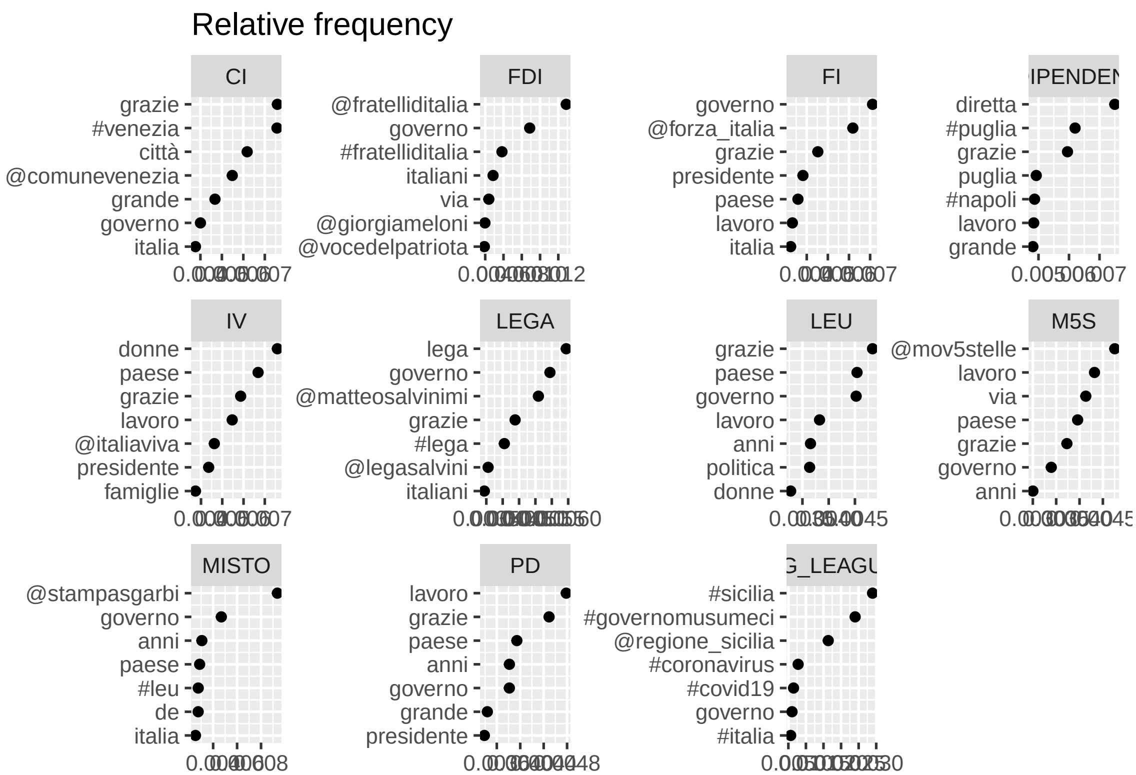

```r
# Plot frequency of the topfeatures in the DFM
features_dfm <- textstat_frequency(DFM, n = 50)
# Sort by reverse frequency order
features_dfm$feature <- with(features_dfm, reorder(feature, -frequency))
ggplot(features_dfm, aes(x = feature, y = frequency)) +
  geom_point() +
  theme(axis.text.x = 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)
```

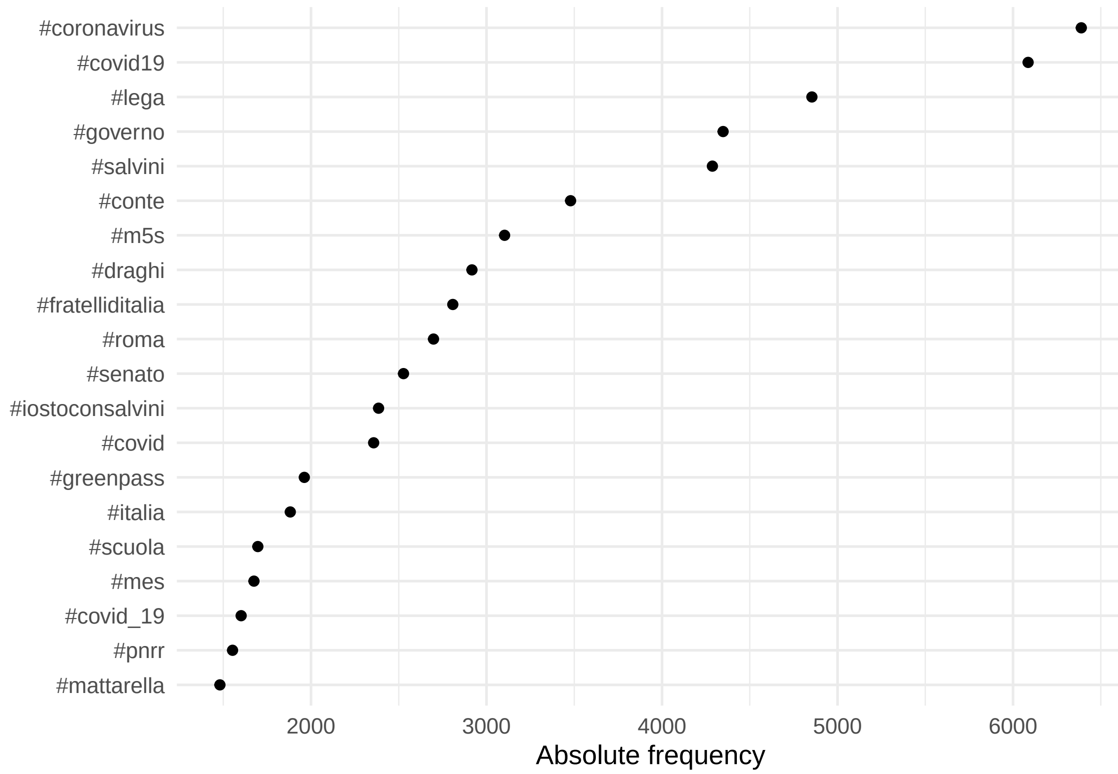


2.3 Most common hashtag

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

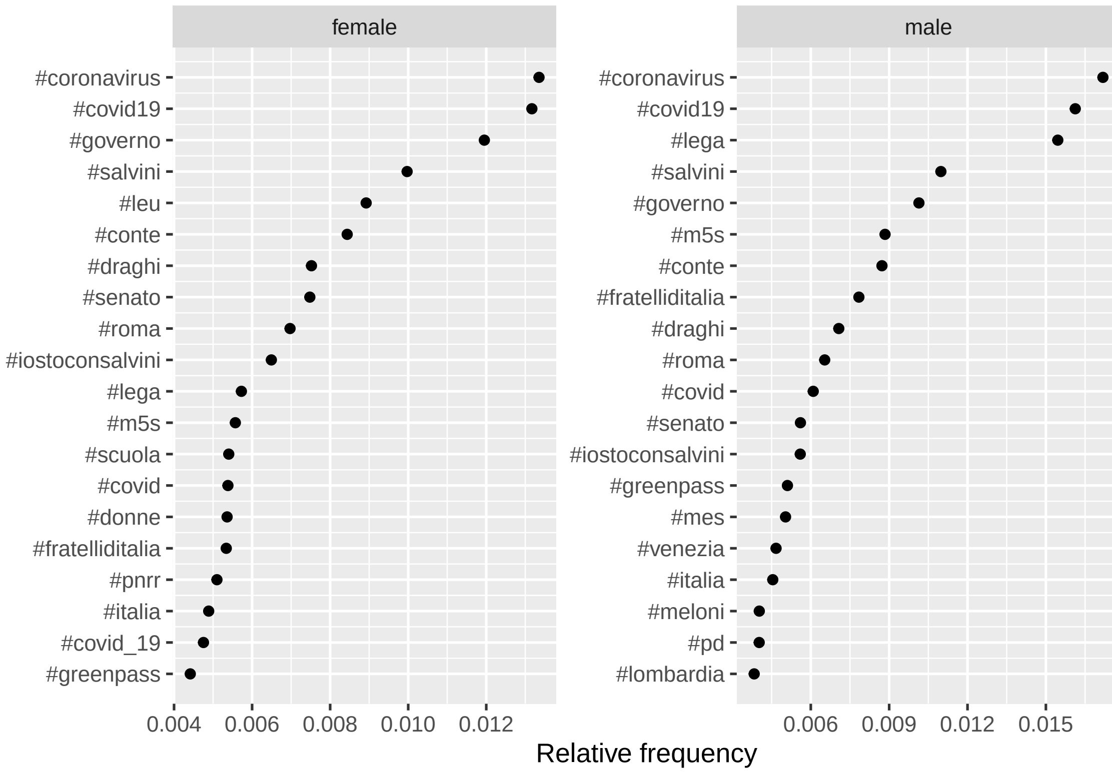
```
## [1] "#coronavirus"     "#covid19"          "#lega"             "#governo"
```

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



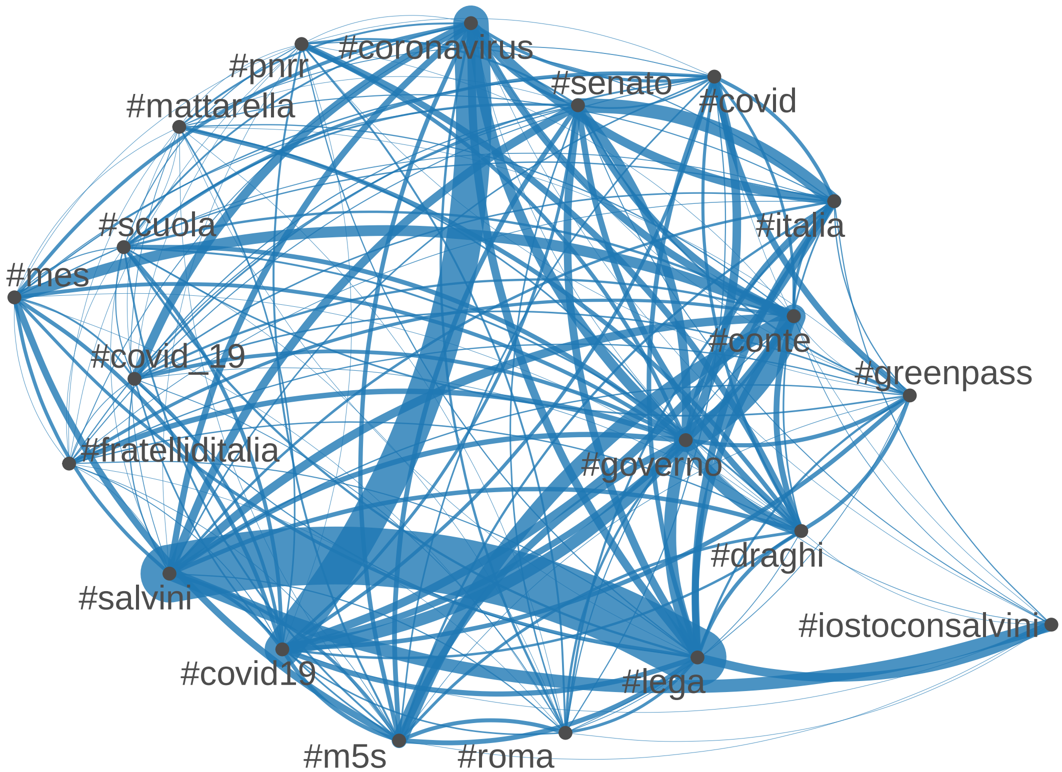
2.3.1 Most common hashtag by Gender

```
# 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

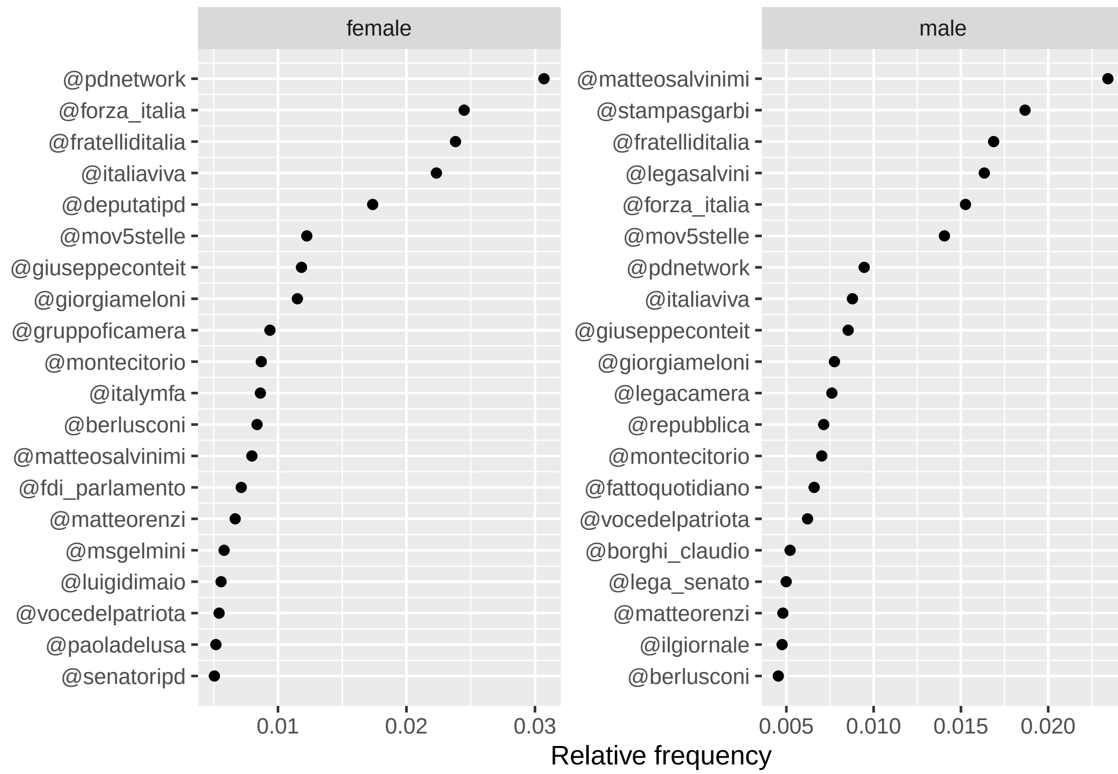
```
user_dfm <- dfm_select(DFM, pattern = "@*")
topuser <- names(topfeatures(user_dfm, 20, scheme = "docfreq"))
kable(topuser, col.names = "Most mentioned username")
```

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

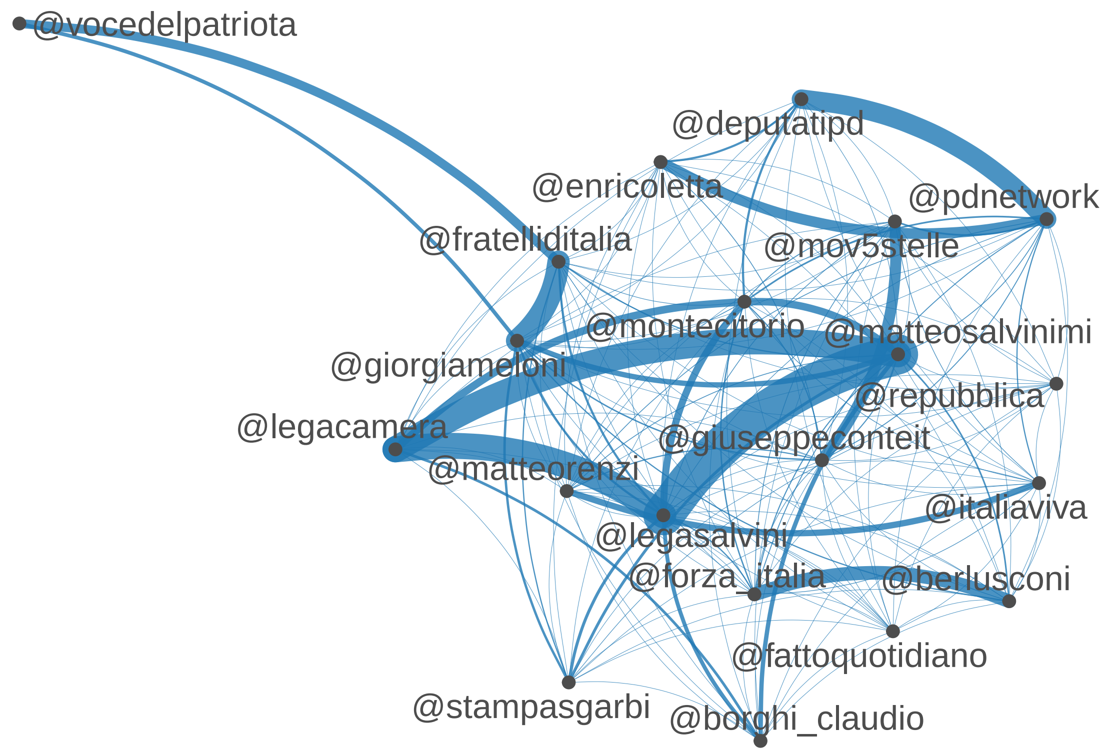
2.4.1 Most frequently mentioned usernames by gender

```
# 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",
```

```r
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

```r
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")
```

```r
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))
```

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

```r
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_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)]))
```

```r
dictionaries <- c("Rooduijn_Pauwels_Italian", "Grundl_Italian_adapted"
                  ,"Decadri_Boussalis_Grundl")
n.words <- c(
  length(Rooduijn_Pauwels_Italian$populism),
  length(Grundl_Italian_adapted$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_Grundl	77

3.1.1 Group and weight the dfm

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

Apply the dictionaries

3.2 Decadri_Boussalis_Grundl

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

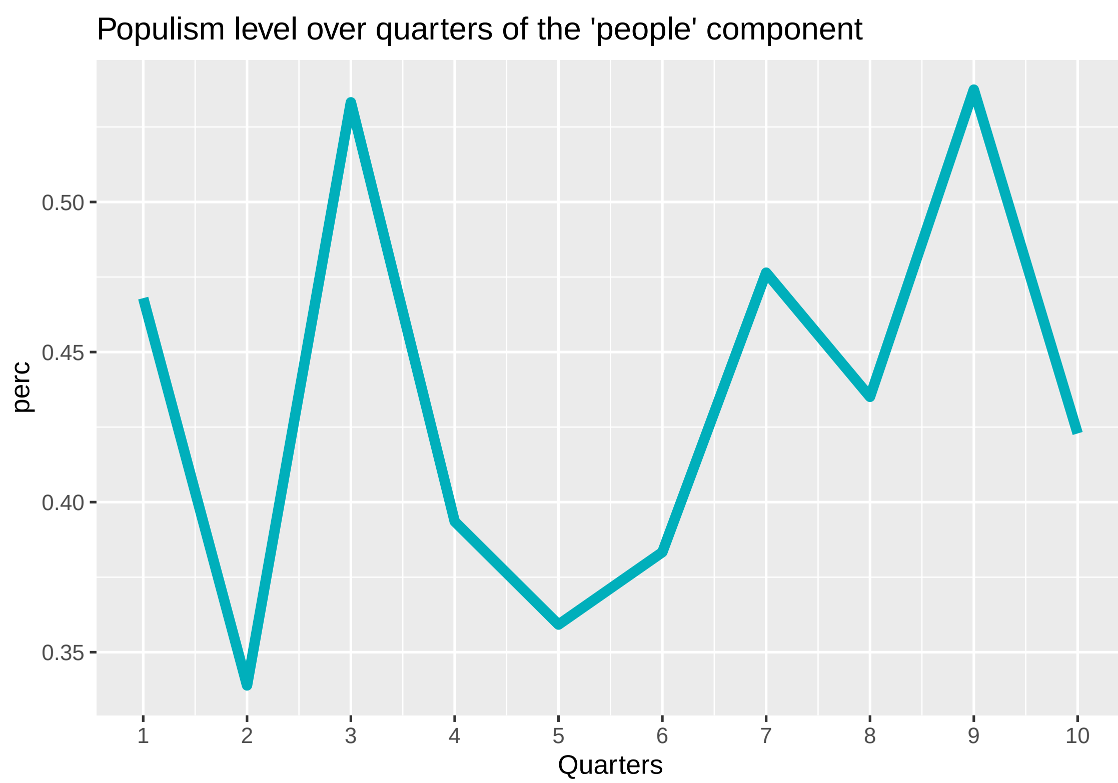
3.2.1 Transform the DFM into an ordinary dataframe

```
data_dict1 <- dfm_dict1 %>%
  quanteda::convert(to = "data.frame") %>%
  cbind(docvars(dfm_dict1))

# Add variable with general level of populism
data_dict1 <- data_dict1 %>% mutate(populism = (people + common_will + elite) * 100)
```

3.2.2 Level of populism in time

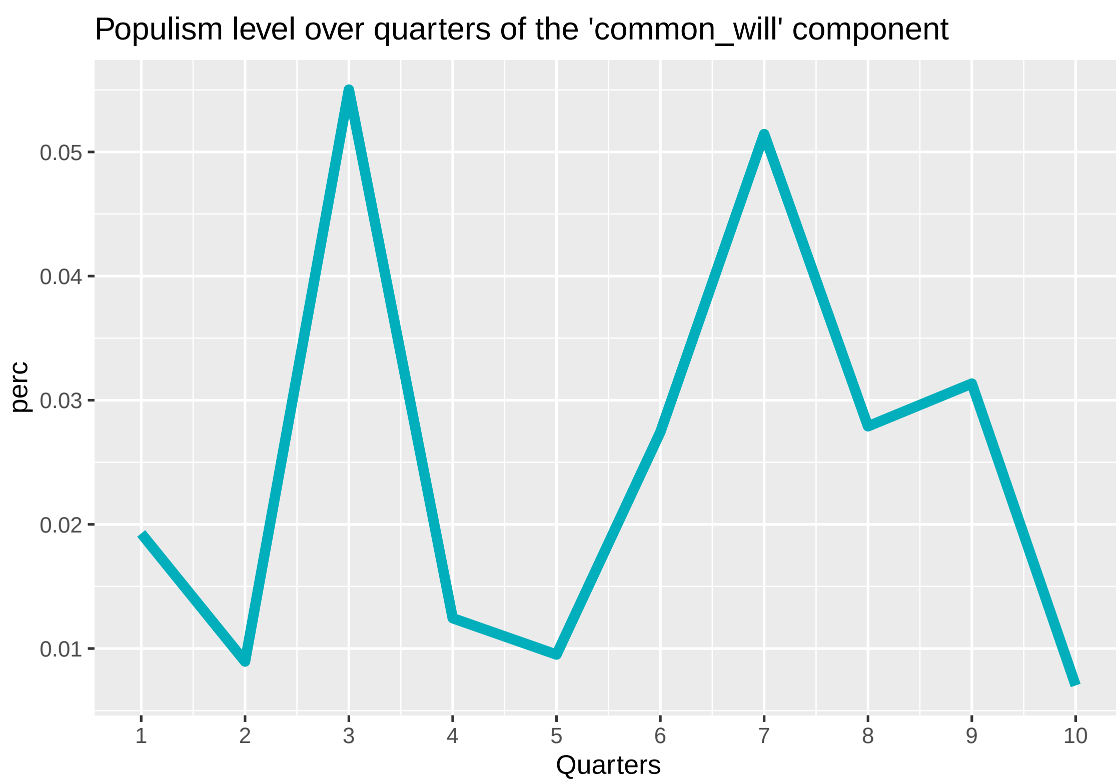
```
#Over time PEOPLE (quarters)
data_quarter_people <- aggregate(x = data_dict1$people,  # Specify data column
          by = list(data_dict1$quarter),  # Specify group indicator
          FUN = mean) # Specify function (i.e. mean)
data_quarter_people$perc <- data_quarter_people$x * 100

# plot the level of the "people" component in time
plot_people <- ggplot(data = data_quarter_people, aes(x = Group.1, y = perc))+
```

```
  geom_line(color = "#00AFBB", size = 2)+
  scale_x_continuous("Quarters", labels = as.character(data_quarter_people$Group.1)
  labs(title = "Populism level over quarters of the 'people' component")
plot_people
```



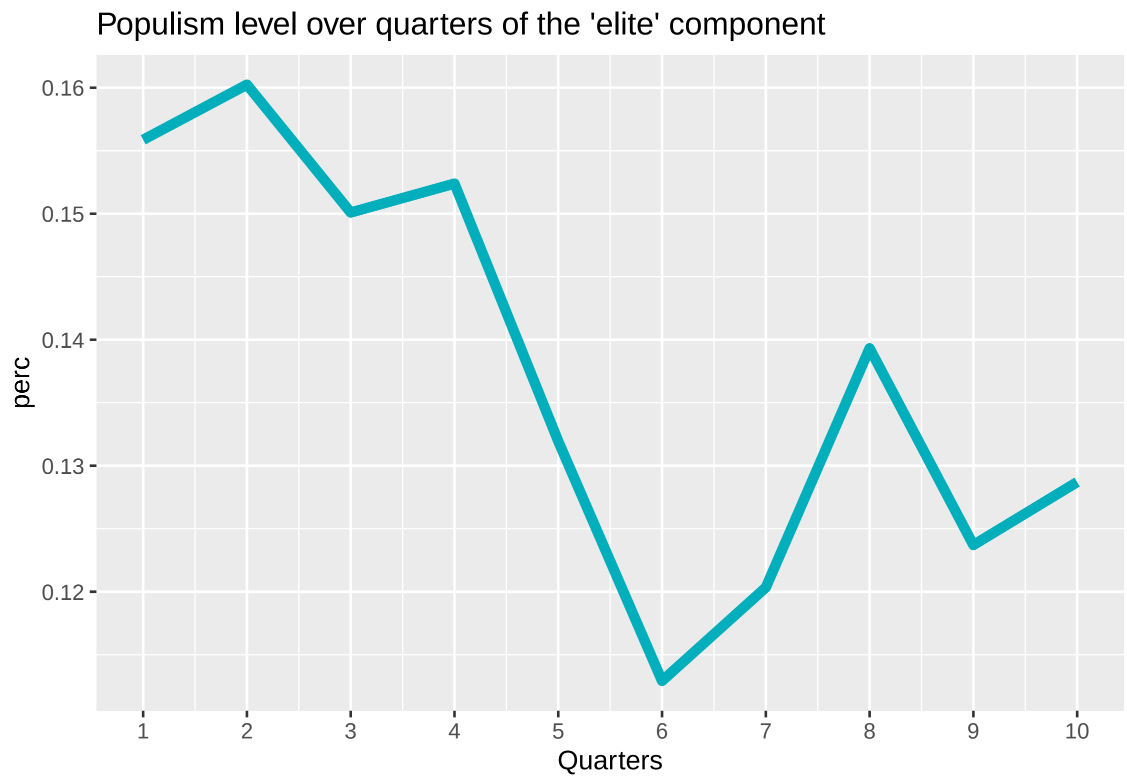

```
############
#Over time COMMON WILL (quarters)
data_quarter_common <- aggregate(x = data_dict1$common_will,  # Specify data column
          by = list(data_dict1$quarter),  # Specify group indicator
          FUN = mean) # Specify function (i.e. mean)
data_quarter_common$perc <- data_quarter_common$x * 100

# plot the level of the "common will" component in time
plot_common <- ggplot(data = data_quarter_common, aes(x = Group.1, y = perc))+
```

```r
  geom_line(color = "#00AFBB", size = 2)+
  scale_x_continuous("Quarters", labels = as.character(data_quarter_common$Group.1)
  labs(title = "Populism level over quarters of the 'common_will' component")
plot_common
```



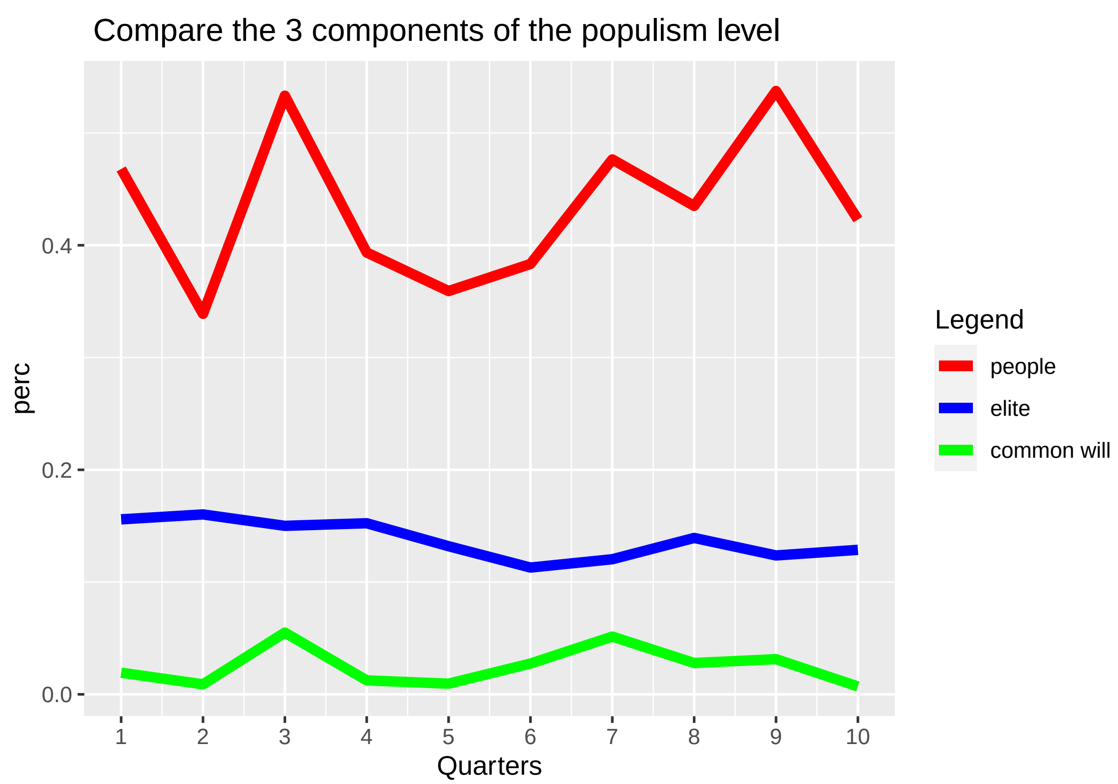

```r
############
#Over time ELITE (quarters)
data_quarter_elite <- aggregate(x = data_dict1$elite,  # Specify data column
          by = list(data_dict1$quarter),  # Specify group indicator
          FUN = mean) # Specify function (i.e. mean)
data_quarter_elite$perc <- data_quarter_elite$x * 100

# plot the level of the "ELITE" component in time
plot_elite <- ggplot(data = data_quarter_elite, aes(x = Group.1, y = perc))+
```

```
  geom_line(color = "#00AFBB", size = 2)+
  scale_x_continuous("Quarters", labels = as.character(data_quarter_elite$Group.1),
  labs(title = "Populism level over quarters of the 'elite' component")
plot_elite
```



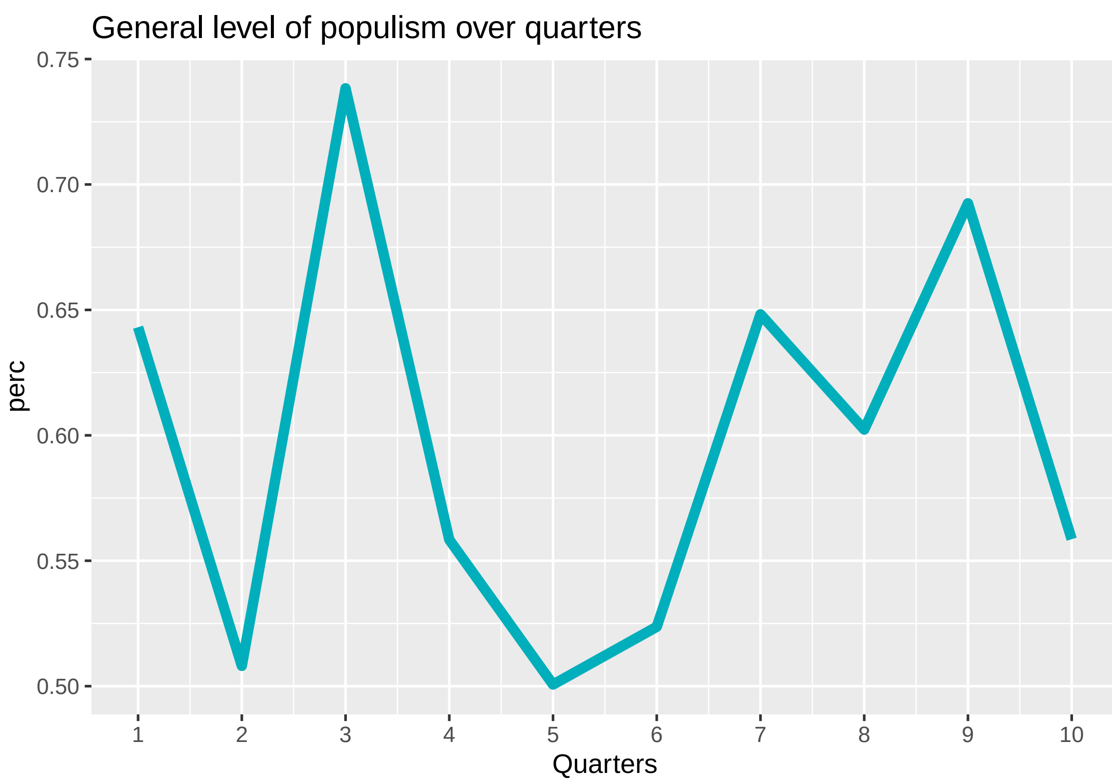

```
#########
# compare the levels
p <- ggplot() +
  # plot people
  geom_line(data = data_quarter_people, aes(x = Group.1, y = perc, color = "people")
  # plot common will
  geom_line(data = data_quarter_common, aes(x = Group.1, y = perc, color = "common w
  # plot elite
  geom_line(data = data_quarter_elite, aes(x = Group.1, y = perc, color = "elite"),
```

```
  scale_color_manual(name='Legend',
                     breaks=c('people', 'elite', 'common will'),
                     values=c('people'='red', 'elite'='blue', 'common will'='green')

  scale_x_continuous("Quarters", labels = as.character(data_quarter_people$Group.1)

  labs(title = " Compare the 3 components of the populism level")

p
```

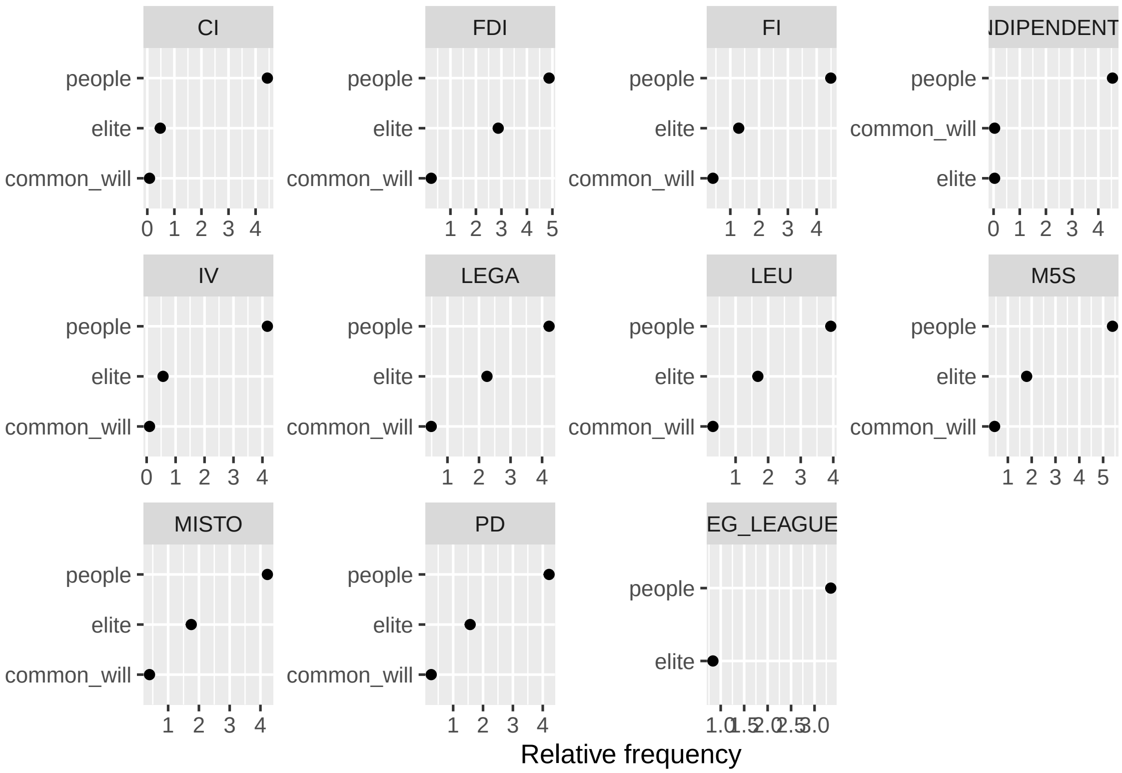



```
################
#Over time general level populism (quarters)
data_quarter_general <- aggregate(x = data_dict1$populism,  # Specify data column
          by = list(data_dict1$quarter),  # Specify group indicator
```

```
          FUN = mean) # Specify function (i.e. mean)
data_quarter_general$perc <- data_quarter_general$x

# plot the level of populism
plot_general <- ggplot(data = data_quarter_general, aes(x = Group.1, y = perc))+
  geom_line(color = "#00AFBB", size = 2)+
  scale_x_continuous("Quarters", labels = as.character(data_quarter_general$Group.1)
  labs(title = "General level of populism over quarters")
plot_general
```



3.2.3 Main component for each parliamentary group

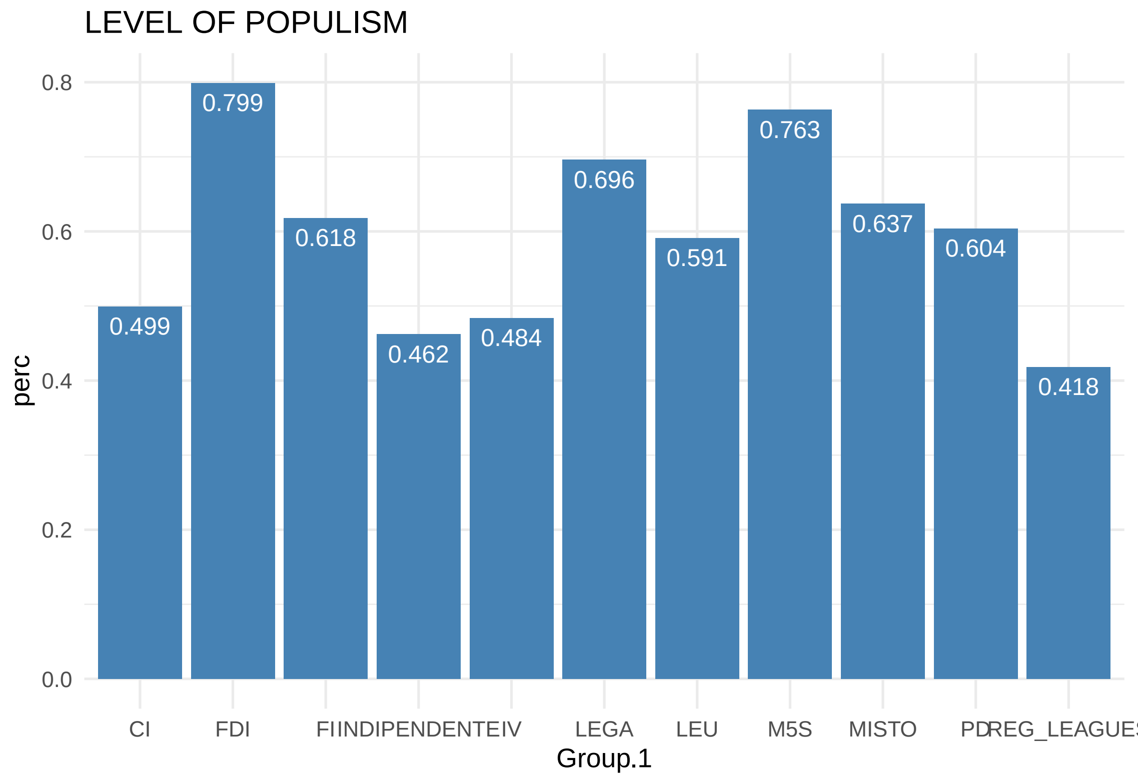


3.2.4 Most populist parliamentary group

```r
#By party no time (quarters)


# POPULISM
data_party <- aggregate(x = data_dict1$populism,  # Specify data column
          by = list(data_dict1$party_id),  # Specify group indicator
          FUN = mean) # Specify function (i.e. mean)
data_party$perc <- round(data_party$x,3)
kable(data_party %>% select(Group.1, perc) %>%  arrange(desc(perc)))
```

Group.1	perc
FDI	0.799
M5S	0.763
LEGA	0.696
MISTO	0.637
FI	0.618
PD	0.604
LEU	0.591
CI	0.499
IV	0.484
INDIPENDENTE	0.462
REG_LEAGUES	0.418

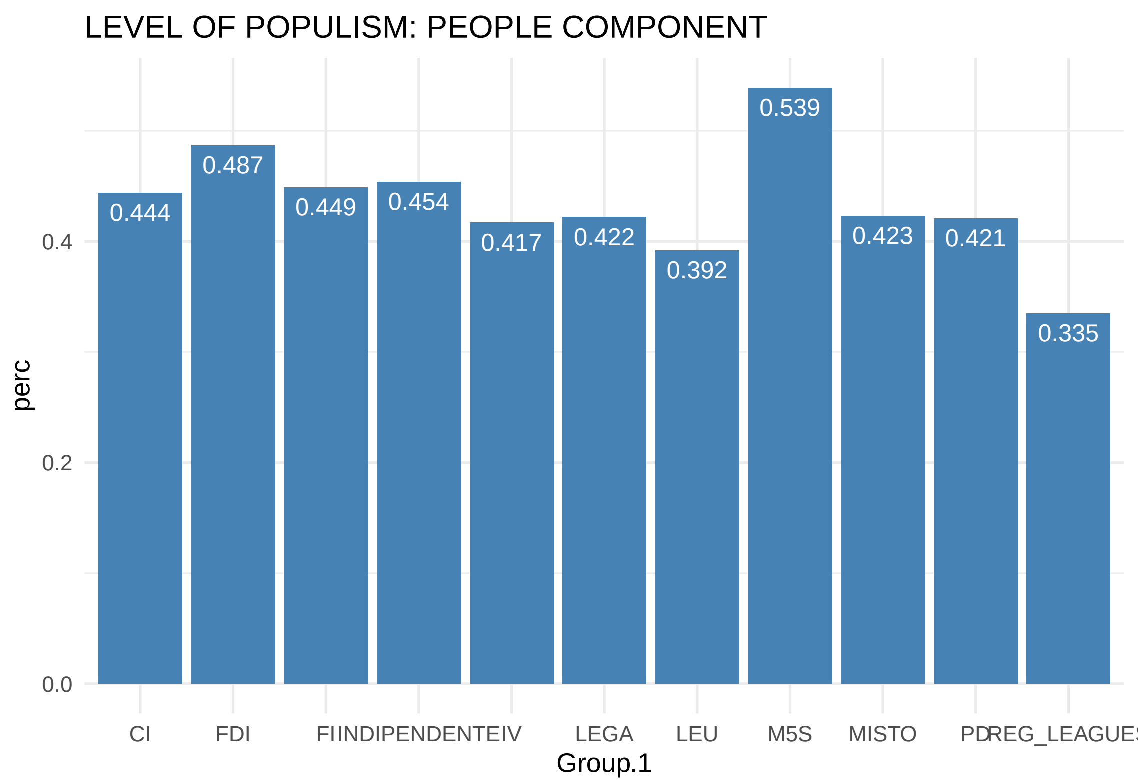
```
ggplot(data=data_party, aes(x=Group.1, y=perc)) +
  geom_bar(stat="identity", fill="steelblue")+
  geom_text(aes(label=perc), vjust=1.6, color="white", size=3.5)+
  theme_minimal()+
  labs(title = "LEVEL OF POPULISM")
```




```
# PEOPLE
data_party_people <- aggregate(x = data_dict1$people,  # Specify data column
          by = list(data_dict1$party_id),  # Specify group indicator
          FUN = mean) # Specify function (i.e. mean)
data_party_people$perc <- round(data_party_people$x * 100,3)
kable(data_party_people %>% select(Group.1, perc)%>% arrange(desc(perc)))
```

Group.1	perc
M5S	0.539
FDI	0.487
INDIPENDENTE	0.454
FI	0.449
CI	0.444
MISTO	0.423
LEGA	0.422
PD	0.421
IV	0.417
LEU	0.392
REG_LEAGUES	0.335

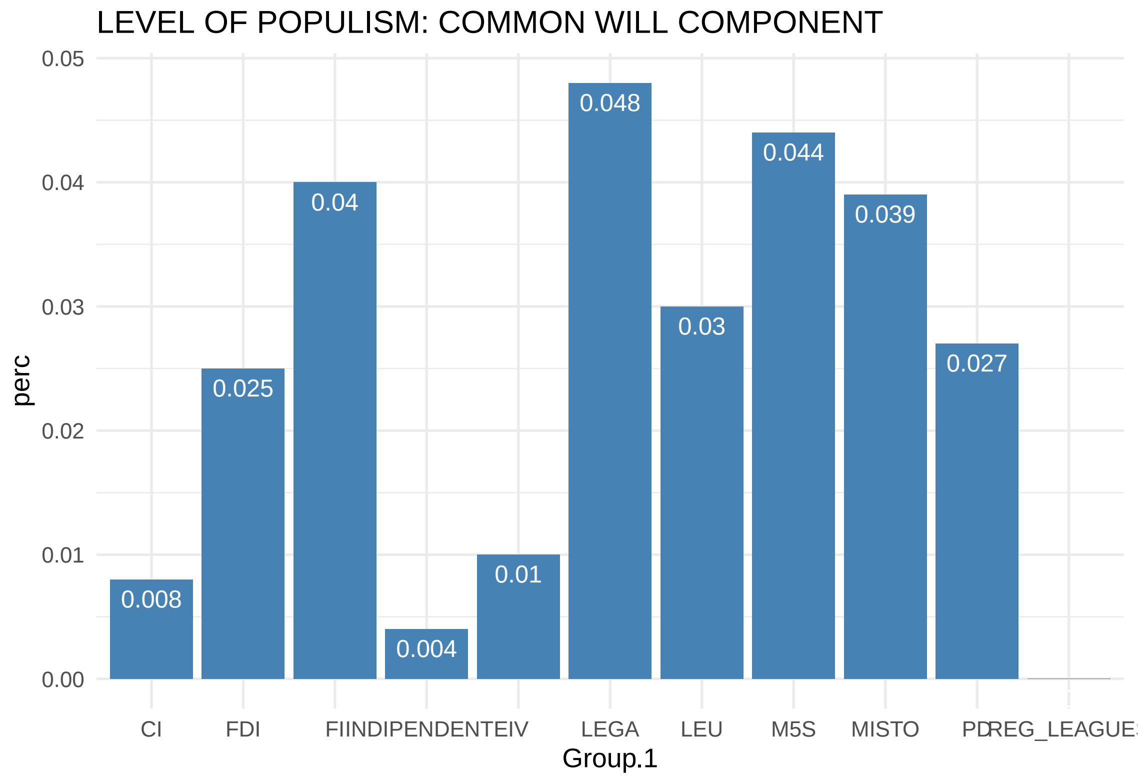
```
ggplot(data=data_party_people, aes(x=Group.1, y=perc)) +
  geom_bar(stat="identity", fill="steelblue")+
  geom_text(aes(label=perc), vjust=1.6, color="white", size=3.5)+
  theme_minimal()+
  labs(title = "LEVEL OF POPULISM: PEOPLE COMPONENT")
```




```
# COMMON WILL
data_party_common <- aggregate(x = data_dict1$common_will,  # Specify data column
          by = list(data_dict1$party_id),  # Specify group indicator
          FUN = mean) # Specify function (i.e. mean)
data_party_common$perc <- round(data_party_common$x * 100,3)
kable(data_party_common %>% select(Group.1, perc)%>% arrange(desc(perc)))
```

Group.1	perc
LEGA	0.048
M5S	0.044
FI	0.040
MISTO	0.039
LEU	0.030
PD	0.027
FDI	0.025
IV	0.010
CI	0.008
INDIPENDENTE	0.004
REG_LEAGUES	0.000

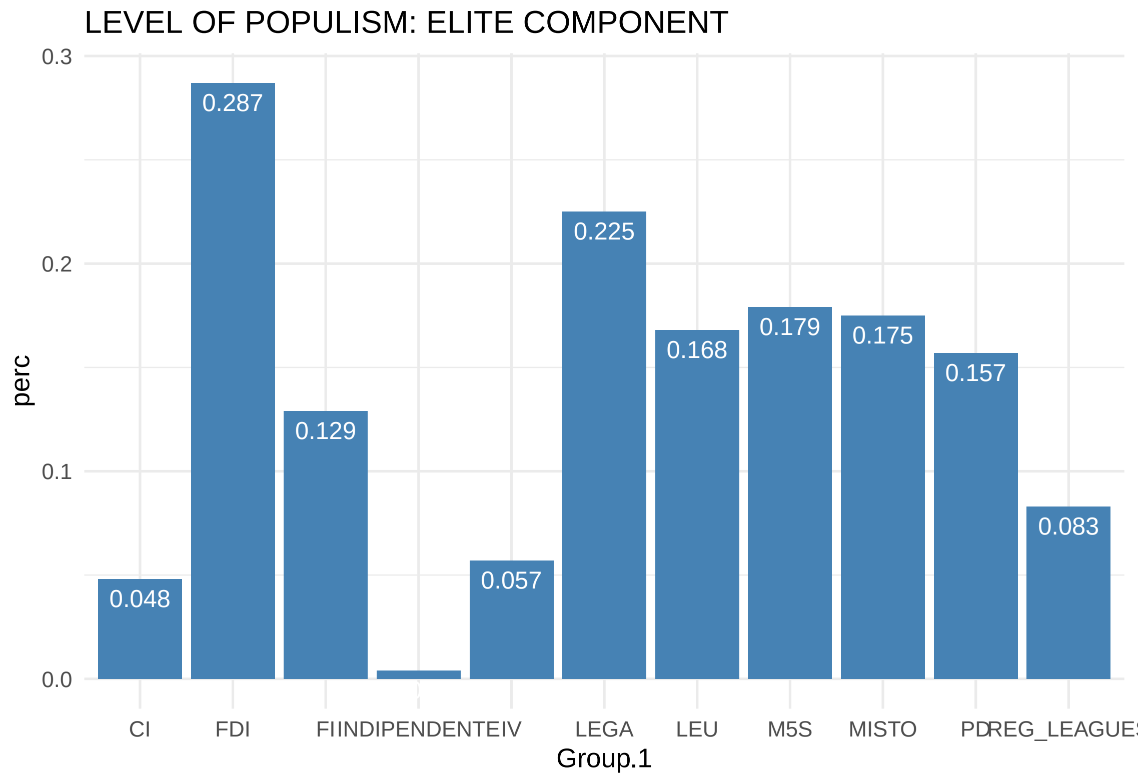
```
ggplot(data=data_party_common, aes(x=Group.1, y=perc)) +
  geom_bar(stat="identity", fill="steelblue")+
  geom_text(aes(label=perc), vjust=1.6, color="white", size=3.5)+
  theme_minimal()+
  labs(title = "LEVEL OF POPULISM: COMMON WILL COMPONENT")
```




```
# ELITE
data_party_elite <- aggregate(x = data_dict1$elite,  # Specify data column
          by = list(data_dict1$party_id),  # Specify group indicator
          FUN = mean) # Specify function (i.e. mean)
data_party_elite$perc <- round(data_party_elite$x * 100,3)
kable(data_party_elite %>% select(Group.1, perc)%>% arrange(desc(perc)))
```

Group.1	perc
FDI	0.287
LEGA	0.225
M5S	0.179
MISTO	0.175
LEU	0.168
PD	0.157
FI	0.129
REG_LEAGUES	0.083
IV	0.057
CI	0.048
INDIPENDENTE	0.004

```
ggplot(data=data_party_elite, aes(x=Group.1, y=perc)) +
  geom_bar(stat="identity", fill="steelblue")+
  geom_text(aes(label=perc), vjust=1.6, color="white", size=3.5)+
  theme_minimal()+
  labs(title = "LEVEL OF POPULISM: ELITE COMPONENT")
```



Are the average values of populism for each party statistically different from each other? The reference category is PD

```
# bivariate regression for check t-test
data_dict1$factor_party <- as.factor(data_dict1$party_id)
data_dict1$factor_party <- relevel(data_dict1$factor_party, ref = "PD")

data_dict1$factor_quarter <- as.factor(data_dict1$quarter)
data_dict1$factor_quarter <- relevel(data_dict1$factor_quarter, ref = "8")

a3 <- lm(populism ~  factor_quarter + factor_party, data_dict1 )

summary(a3)
```

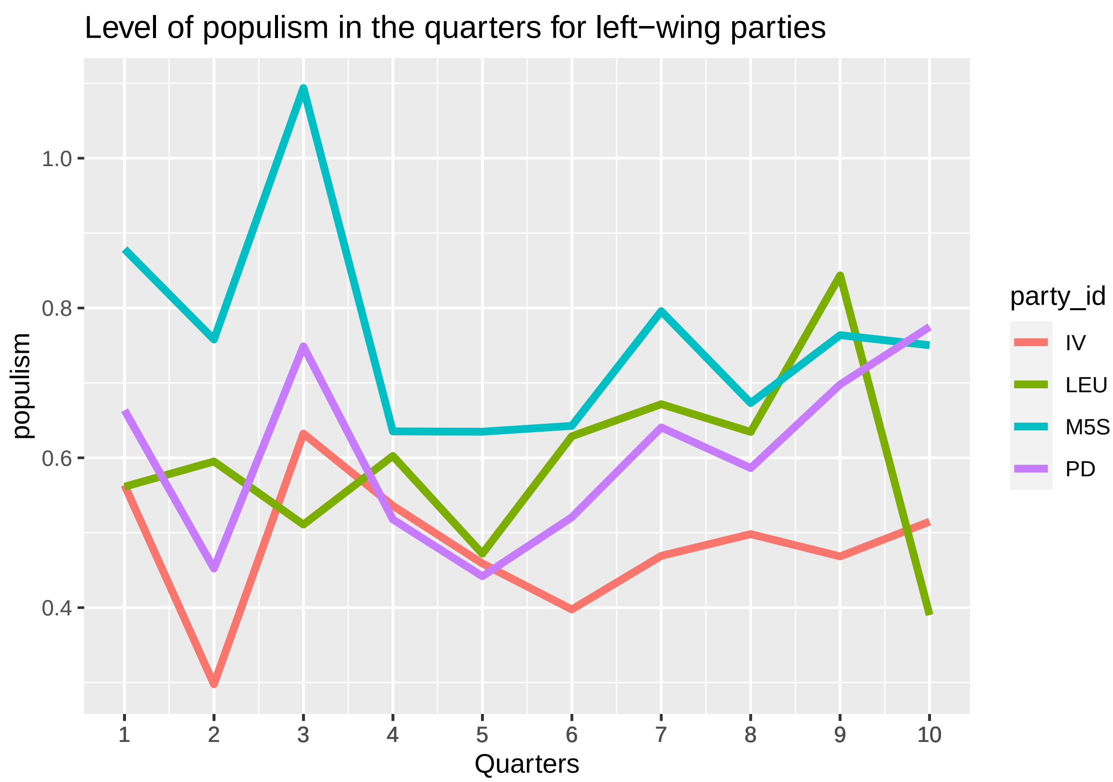
```
##
```

```
## Call:
## lm(formula = populism ~ factor_quarter + factor_party, data = data_dict1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.30617 -0.06571  0.00588  0.05535  0.32599
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 0.60934    0.05058  12.046  < 2e-16 ***
## factor_quarter1             0.04082    0.05058   0.807 0.421838
## factor_quarter2            -0.09418    0.05058  -1.862 0.065878 .
## factor_quarter3             0.13606    0.05058   2.690 0.008522 **
## factor_quarter4            -0.04390    0.05058  -0.868 0.387769
## factor_quarter5            -0.10164    0.05058  -2.009 0.047500 *
## factor_quarter6            -0.07861    0.05058  -1.554 0.123684
## factor_quarter7             0.04596    0.05058   0.909 0.365971
## factor_quarter9             0.09022    0.05058   1.783 0.077879 .
## factor_quarter10           -0.04369    0.05058  -0.864 0.390079
## factor_partyCI             -0.10503    0.05305  -1.980 0.050793 .
## factor_partyFDI             0.19458    0.05305   3.668 0.000414 ***
## factor_partyFI              0.01356    0.05305   0.256 0.798859
## factor_partyINDIPENDENTE -0.14233    0.05305  -2.683 0.008687 **
## factor_partyIV             -0.12078    0.05305  -2.277 0.025184 *
## factor_partyLEGA            0.09147    0.05305   1.724 0.088134 .
## factor_partyLEU            -0.01339    0.05305  -0.252 0.801282
## factor_partyM5S             0.15814    0.05305   2.981 0.003698 **
## factor_partyMISTO           0.03265    0.05305   0.615 0.539799
## factor_partyREG_LEAGUES    -0.18644    0.05305  -3.514 0.000693 ***
```

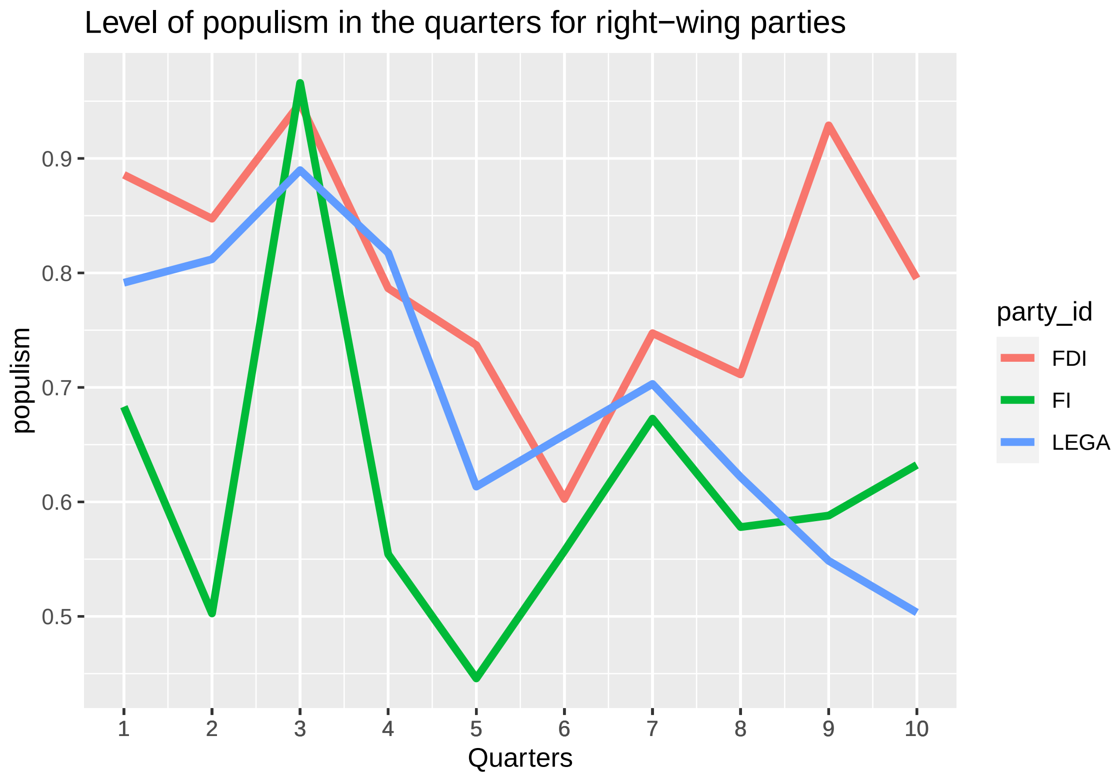
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1186 on 90 degrees of freedom
## Multiple R-squared:  0.6326, Adjusted R-squared:  0.5551
## F-statistic: 8.157 on 19 and 90 DF,  p-value: 1.35e-12
```

3.2.5 Trends in the level of populism for each parliamentary group over time

```r
#By party & time (quarters)
parties_time <- data_dict1 %>% select(populism, party_id, quarter)

right_party <- data_dict1 %>% select(populism, party_id, quarter) %>%
  filter(party_id == "FDI"|party_id =="FI"|party_id =="LEGA")
left_party <- data_dict1 %>% select(populism, party_id, quarter) %>%
  filter(party_id == "LEU"|party_id =="M5S"|party_id =="PD"|party_id =="IV")

# Left parties in time
ggplot(left_party, aes(x=quarter, y=populism, color=party_id)) +
  geom_line(size=1.5)+
  scale_x_continuous("Quarters", labels = as.character(left_party$quarter), breaks =
  ggtitle("Level of populism in the quarters for left-wing parties")
```




```
# Right parties in time
ggplot(right_party, aes(x=quarter, y=populism, color=party_id)) +
  geom_line(size=1.5)+
  scale_x_continuous("Quarters", labels = as.character(right_party$quarter), breaks
  ggtitle("Level of populism in the quarters for right-wing parties")
```


Level of populism in the quarters for right−wing parties
populism
0.9
0.8
0.7
0.6
0.5
1
2
3
4
5
6
7
8
9
10
Quarters
party_id
FDI
FI
LEGA

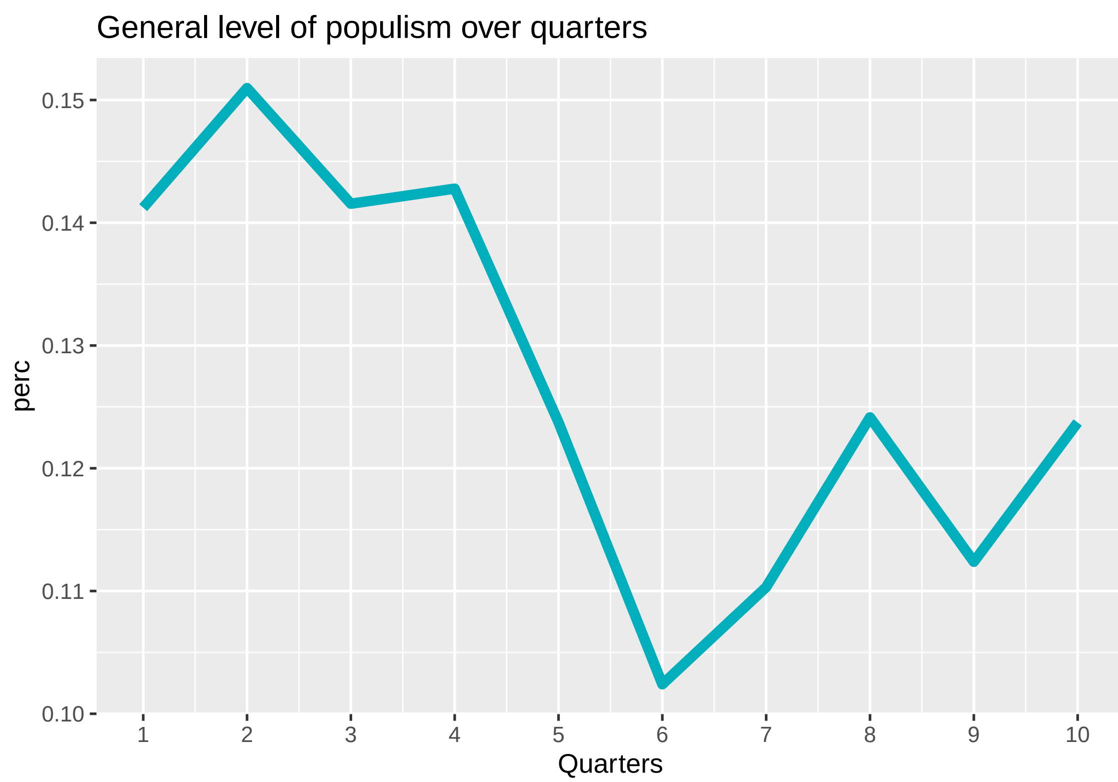
3.3 Rooduijn_Pauwels_Italian

```
# Dictionary analysis with Rooduijn_Pauwels_Italian
# By quarter
dfm_dict2  <- dfm_lookup(dfm_weigh_p_quart, dictionary = Rooduijn_Pauwels_Italian)
```

```
data_dict2 <- dfm_dict2 %>%
  quanteda::convert(to = "data.frame") %>%
  cbind(docvars(dfm_dict2))

# Add variable with general level of populism
#data_dict2 <- data_dict2 %>% mutate(populism = (people + common_will + elite) * 1
```

3.3.1 Level of populism in time

```
#Over time general level populism (quarters)
data_quarter_general2 <- aggregate(x = data_dict2$populism,  # Specify data column
          by = list(data_dict2$quarter),  # Specify group indicator
          FUN = mean) # Specify function (i.e. mean)
data_quarter_general2$perc <- data_quarter_general2$x * 100

# plot the level of populism
plot_general2 <- ggplot(data = data_quarter_general2, aes(x = Group.1, y = perc))+
  geom_line(color = "#00AFBB", size = 2)+
  scale_x_continuous("Quarters", labels = as.character(data_quarter_general2$Group.1
  labs(title = "General level of populism over quarters")
plot_general2
```

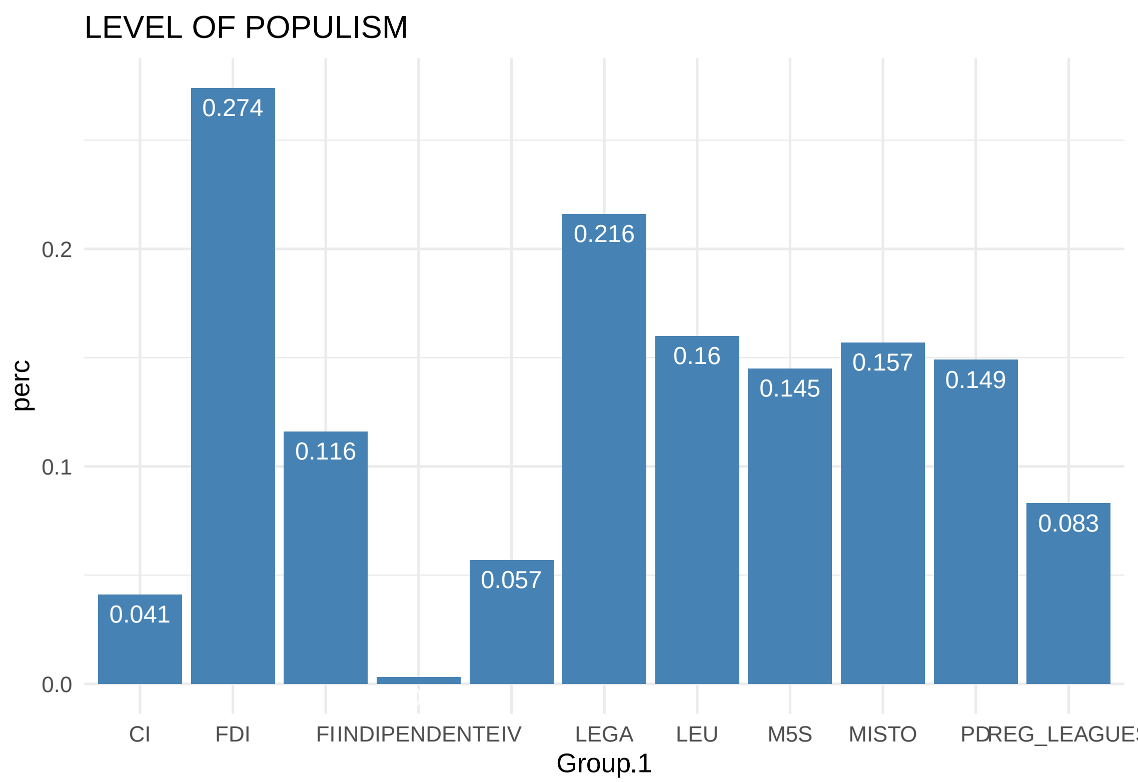


3.3.2 Most populist parliamentary group

```
# POPULISM
data_party2 <- aggregate(x = data_dict2$populism,  # Specify data column
          by = list(data_dict2$party_id),  # Specify group indicator
          FUN = mean) # Specify function (i.e. mean)
data_party2$perc <- round(data_party2$x * 100 ,3)
kable(data_party2 %>% select(Group.1, perc) %>%  arrange(desc(perc)))
```

Group.1	perc
FDI	0.274
LEGA	0.216
LEU	0.160
MISTO	0.157
PD	0.149
M5S	0.145
FI	0.116
REG_LEAGUES	0.083
IV	0.057
CI	0.041
INDIPENDENTE	0.003

```
ggplot(data=data_party2, aes(x=Group.1, y=perc)) +
  geom_bar(stat="identity", fill="steelblue")+
  geom_text(aes(label=perc), vjust=1.6, color="white", size=3.5)+
  theme_minimal()+
  labs(title = "LEVEL OF POPULISM")
```



LEVEL OF POPULISM
0.274
0.216
0.16
0.145
0.157
0.149
0.116
0.083
0.057
0.041
0.2
0.1
0.0
perc
CI
FDI
FI
INDIPENDENTE
IV
LEGA
LEU
M5S
MISTO
PD
Group.1

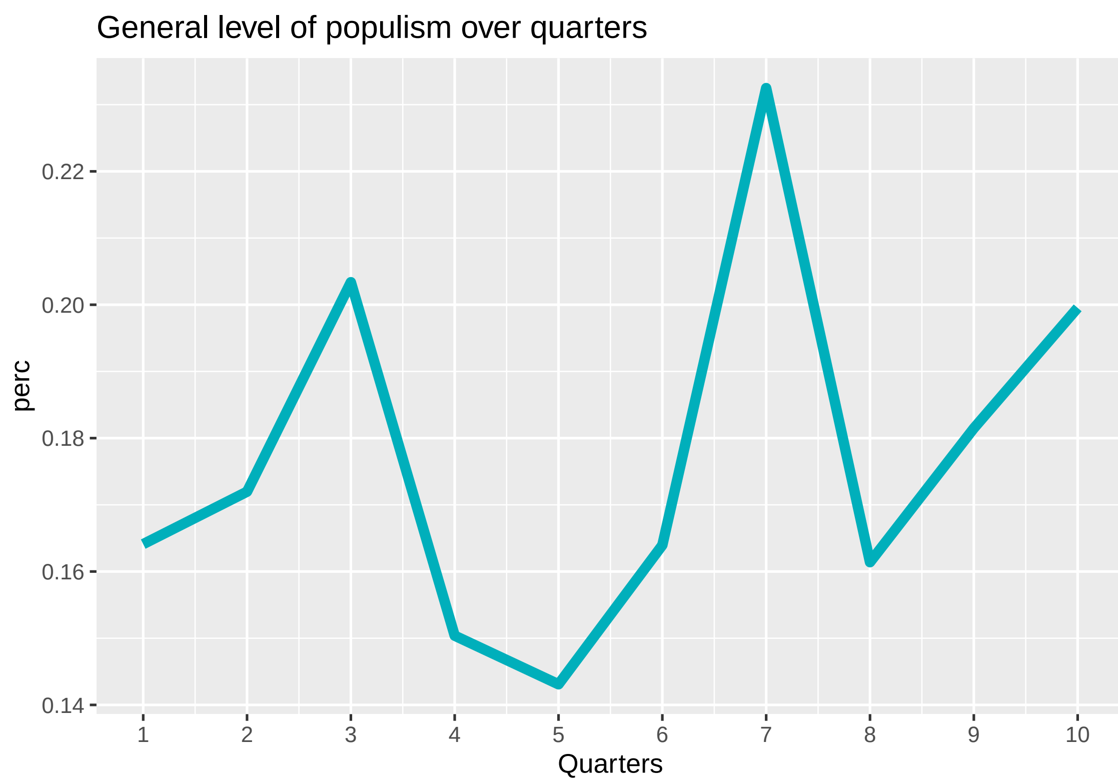
3.4 Grundl_Italian_adapted

```
# Dictionary analysis with Rooduijn_Pauwels_Italian
# By quarter
dfm_dict3  <- dfm_lookup(dfm_weigh_p_quart, dictionary = Grundl_Italian_adapted)
```

```
data_dict3 <- dfm_dict3 %>%
  quanteda::convert(to = "data.frame") %>%
  cbind(docvars(dfm_dict3))

# Add variable with general level of populism
#data_dict2 <- data_dict2 %>% mutate(populism = (people + common_will + elite) * 1
```

3.4.1 Level of populism in time

```
#Over time general level populism (quarters)
data_quarter_general3 <- aggregate(x = data_dict3$populism,  # Specify data column
          by = list(data_dict3$quarter),  # Specify group indicator
          FUN = mean) # Specify function (i.e. mean)
data_quarter_general3$perc <- data_quarter_general3$x * 100

# plot the level of populism
plot_general3 <- ggplot(data = data_quarter_general3, aes(x = Group.1, y = perc))+
  geom_line(color = "#00AFBB", size = 2)+
  scale_x_continuous("Quarters", labels = as.character(data_quarter_general3$Group.1
  labs(title = "General level of populism over quarters")
plot_general3
```

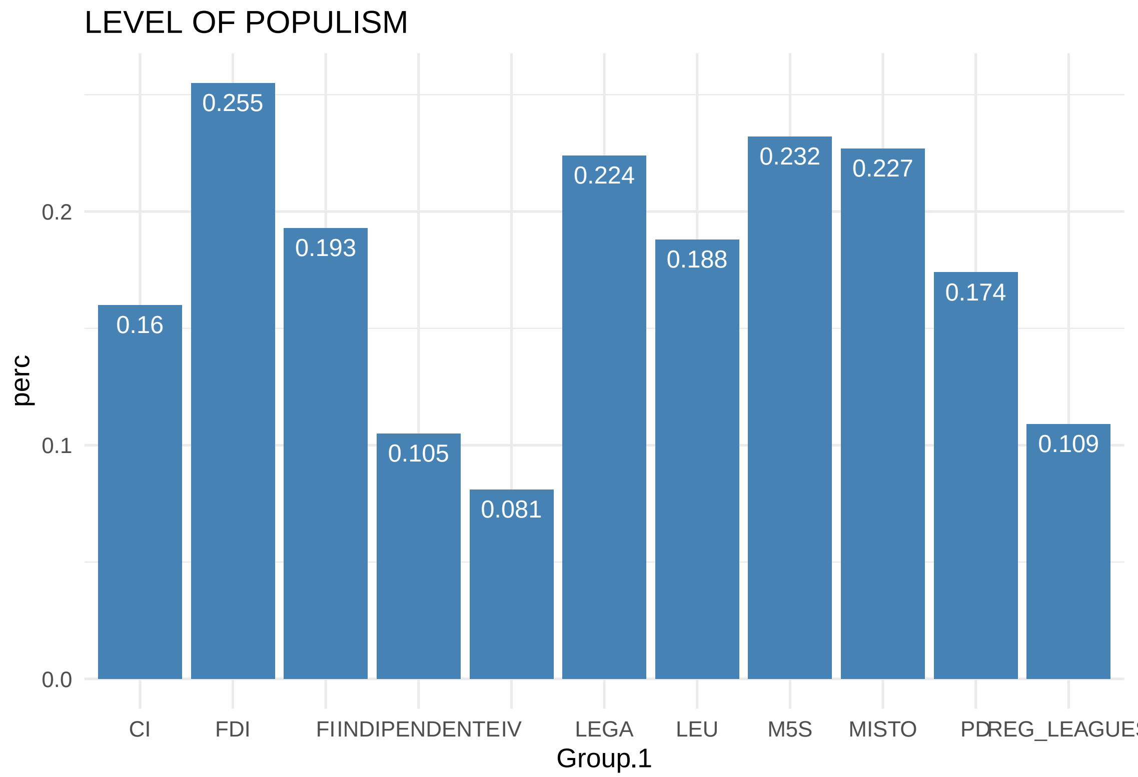


3.4.2 Most populist parliamentary group

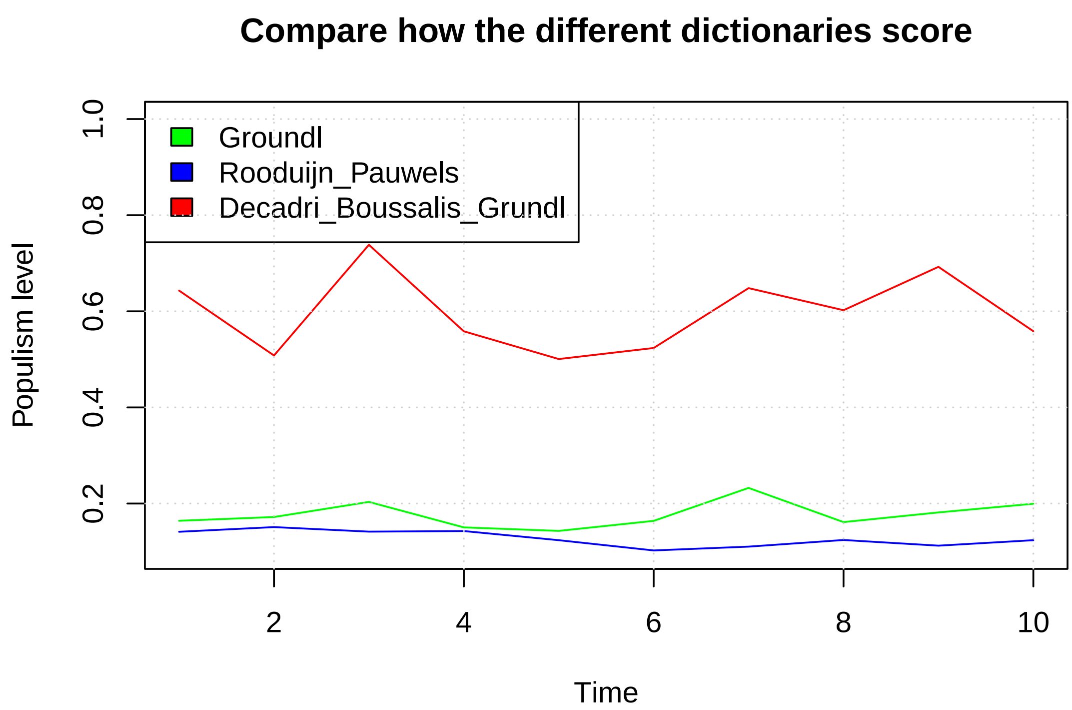
```
# POPULISM
data_party3 <- aggregate(x = data_dict3$populism,  # Specify data column
          by = list(data_dict3$party_id),  # Specify group indicator
          FUN = mean) # Specify function (i.e. mean)
data_party3$perc <- round(data_party3$x * 100 ,3)
kable(data_party3 %>% select(Group.1, perc) %>%  arrange(desc(perc)))
```

Group.1	perc
FDI	0.255
M5S	0.232
MISTO	0.227
LEGA	0.224
FI	0.193
LEU	0.188
PD	0.174
CI	0.160
REG_LEAGUES	0.109
INDIPENDENTE	0.105
IV	0.081

```
ggplot(data=data_party3, aes(x=Group.1, y=perc)) +
  geom_bar(stat="identity", fill="steelblue")+
  geom_text(aes(label=perc), vjust=1.6, color="white", size=3.5)+
  theme_minimal()+
  labs(title = "LEVEL OF POPULISM")
```


LEVEL OF POPULISM
perc
0.0
0.1
0.2
0.16
0.255
0.193
0.105
0.081
0.224
0.188
0.232
0.227
0.174
0.109
CI
FDI
FI
INDIPENDENTE
IV
LEGA
LEU
M5S
MISTO
PD
REG_LEAGUE
Group.1

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



3.6 DA SISTEMARE LA COMPARAZIONE TRA DIZIONARI !

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$my_rank <- rank(dfm_dict1_tstat_party_filtered$popul
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)
```

```r
# 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)), col.names = c("Party",
                                                               "Dec_Bous_Grun",
                                                               "Rood_Pau_it",
                                                               "Grun_it",
                                                               "Dec_Bous",
                                                               "Total"))
```

4 FER: Facial Emotion Recognition Analysis

4.1 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)