Assignment 10-Data Mining and Word Clouds

Pratik Agrawal_804861

21 Januar 2020

Set up

```
#install.packages("webshot")
```

```
#install necessary packages, see txt file
if(FALSE){
  install.packages("rtweet")# access tweets

install.packages("tm") # text mining
install.packages("tidytext")  # text mining

install.packages("magrittr") # provides the pipe %>% operator

install.packages("tidyverse")# collection of packages for data analysis (ggplot2, dplyr, tidy
r, readr, purrr, tibble, stringr, forcats)

install.packages("ggplot2") # visualization
install.packages("stringr") # working with strings
install.packages("lubridate") # working with dates

install.packages("wordcloud") # word-cloud generator
install.packages("wordcloud2") # slightly different design and fun applications
install.packages("RColorBrewer") # package for the colours
install.packages("hunspell")
install.packages("SnowballC")}
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.6.2
```

```
## -- Attaching packages --------------------------------------------------------------------
---------------------------------------------- tidyverse 1.3.0 --
```

```
## <U+2713> ggplot2 3.2.1     <U+2713> purrr   0.3.3
## <U+2713> tibble  2.1.3     <U+2713> dplyr   0.8.3
## <U+2713> tidyr   1.0.0     <U+2713> stringr 1.4.0
## <U+2713> readr   1.3.1     <U+2713> forcats 0.4.0
```

```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
## Warning: package 'tidyr' was built under R version 3.6.2
## Warning: package 'readr' was built under R version 3.6.2
```

```
## Warning: package 'purrr' was built under R version 3.6.2
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

```
## Warning: package 'stringr' was built under R version 3.6.2
```

```
## Warning: package 'forcats' was built under R version 3.6.2
```

```
## -- Conflicts -----------------------------------------------------------------------------
----------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidytext)
```

```
## Warning: package 'tidytext' was built under R version 3.6.2
```

```
library(stringr) #manipulating text data
library(wordcloud2) #create wordclouds
```

```
## Warning: package 'wordcloud2' was built under R version 3.6.2
```

Accessing tweets

```
#access 1000 tweets with the hashtag #LibyaConference OR #LibyenKonferenz, do not include ret
weets, language is english
library(rtweet)
```

```
## Warning: package 'rtweet' was built under R version 3.6.2
```

```
##
## Attaching package: 'rtweet'
```

```
## The following object is masked from 'package:purrr':
##
##     flatten
```

```
# Speficy Authentification Token's provided in your Twitter App
create_token(
  app = "Sentiment_ndtv",
  consumer_key = ,
  consumer_secret = ,
  access_token = ,
  access_secret = 

)
```

```
## <Token>
## <oauth_endpoint>
##  request:   https://api.twitter.com/oauth/request_token
##  authorize: https://api.twitter.com/oauth/authenticate
##  access:    https://api.twitter.com/oauth/access_token
## <oauth_app> Sentiment_ndtv
##   key:    N7mHCm2vQ12lBsN0Q26aL6kNc
##   secret: <hidden>
## <credentials> oauth_token, oauth_token_secret
## ---
```

Exercise 1

Select a tweeter user of your interest. Briefly (1-2 sentences) reason your choice in comment of the code.

“For this Assignment I have chosen Narendra Modi’s account @Narendramodi because he is one of the most popular Politicians of india and he is also extremly active on twitter”.

a) Scrape 1000 tweets of this user. Present them as a dataframe in R and show the first 6 rows of this dataset.

```
tmls_politics <- get_timeline("narendramodi",n = 1000)
head(tmls_politics)
```

user_id <chr>	status_id <chr>	created_at <S3: POSIXct>	screen_name <chr>	▸
18839785	1220177983740219392	2020-01-23 02:54:35	narendramodi	
18839785	1220177665203888128	2020-01-23 02:53:19	narendramodi	
18839785	1220176946044243971	2020-01-23 02:50:28	narendramodi	
18839785	1219991121943621632	2020-01-22 14:32:04	narendramodi	
18839785	1219991120202981377	2020-01-22 14:32:04	narendramodi	
18839785	1219824989521637377	2020-01-22 03:31:55	narendramodi	

6 rows | 1-4 of 90 columns

```
dim(tmls_politics)
```

```
## [1] 1000   90
```

b) How many “likes“ and “retweets” average post from this user becomes? Print out top 5 most “liked” posts.

```
cols <- c("favorite_count","retweet_count")
summary(tmls_politics[cols])
```

```
##  favorite_count   retweet_count
##  Min.   :     0   Min.   :  147
##  1st Qu.: 11131   1st Qu.: 1947
##  Median : 16414   Median : 2788
##  Mean   : 22799   Mean   : 3841
##  3rd Qu.: 26500   3rd Qu.: 4298
##  Max.   :179093   Max.   :42901
```

on average a tweet gets ~22k likes and ~4k retweets from Modi.

```
# order the tweets by favorite_count variable in descending order and print top 5
cols <- c("favorite_count","text")
df <- tmls_politics[order(-tmls_politics$favorite_count),cols]
head(df,5)
```

favorite_count <int>
179093
171985
169616
147341
140891

5 rows | 1-1 of 2 columns

c) Return 10 most often referenced accounts within your sample of tweets. Referenced accounts can be recognized by @useraccount. Example from Obama tweets: Thank you for your leadership @RepHalRogers. This epidemic doesn’t discriminate between red or blue, so it’s up to all of us to do our part. Plot the frequency of 10 most often referenced accounts.

```
# first remove all NA values from the mentions column,
df <- tmls_politics[!is.na(tmls_politics$mentions_screen_name),]
head(df$mentions_screen_name)
```

```
## [[1]]
## [1] "PM_Nepal"
## 
## [[2]]
## [1] "AmitShah"  "BJP4India"
## 
## [[3]]
## [1] "JPNadda"   "BJP4India"
## 
## [[4]]
## [1] "JPNadda"   "BJP4India"
## 
## [[5]]
## [1] "examwarriors"
## 
## [[6]]
## [1] "AzmiShabana"
```

```
## now since the mentions are List, we need to unnest these mentiones into seperate rows
library(dplyr)
library(tidyr)
df <- df %>%
   unnest(mentions_screen_name)
  head(df$mentions_screen_name)
```

```
## [1] "PM_Nepal"  "AmitShah"  "BJP4India" "JPNadda"   "BJP4India" "JPNadda"
```

```
## now group by the mentions_screen_name, sort and display top 10 mentioned user names
 df2 <- df %>% group_by(mentions_screen_name) %>%  tally(sort = TRUE) %>% top_n(10)
```

```
## Selecting by n
```

```
head(df2,10)
```

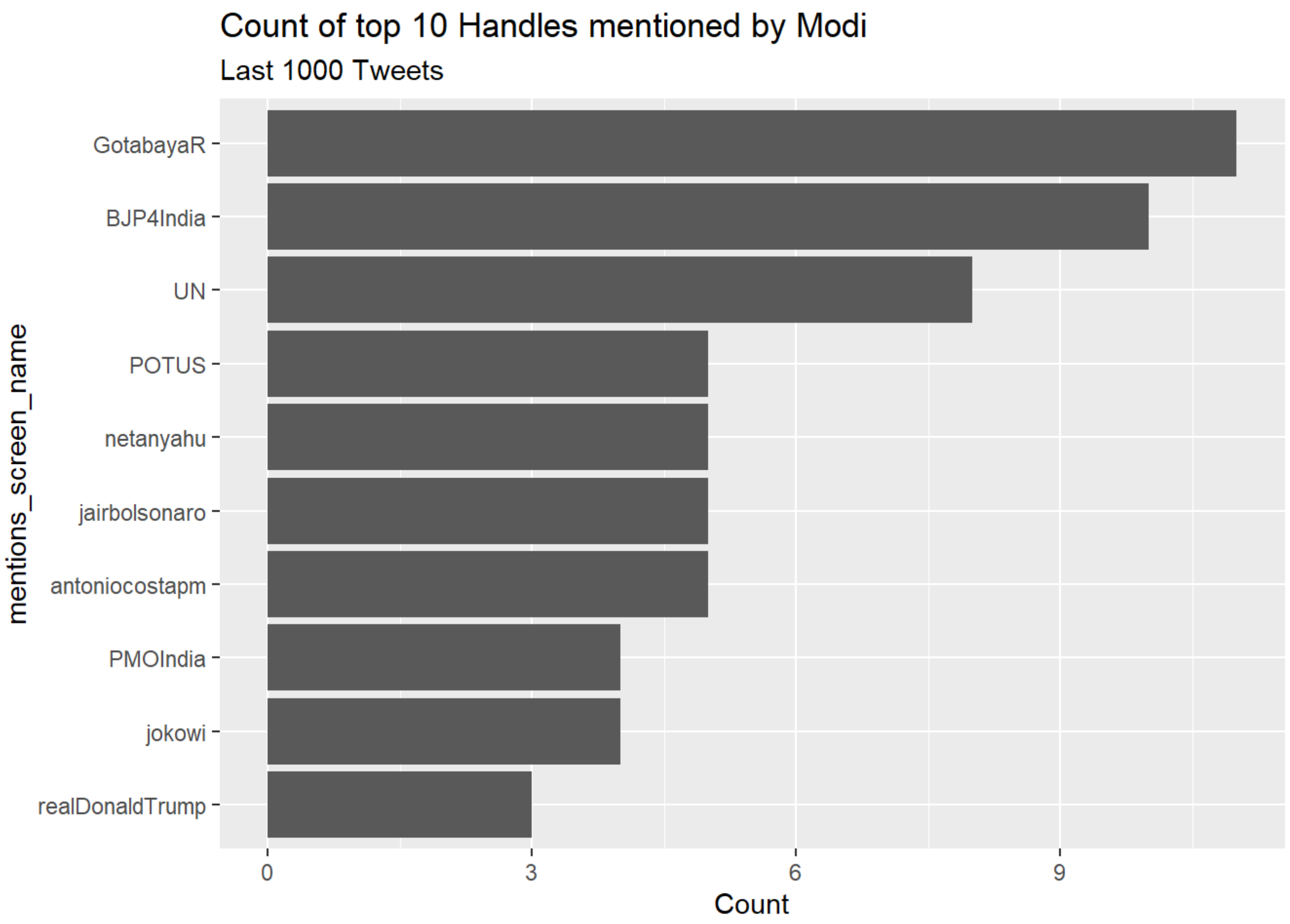
mentions_screen_name <chr>	n <int>
GotabayaR	11
BJP4India	10
UN	8
antoniocostapm	5
jairbolsonaro	5
netanyahu	5
POTUS	5
jokowi	4
PMOIndia	4

mentions_screen_name	n
<chr>	<int>
AbeShinzo	3

1-10 of 10 rows

```
#Plot
library(ggplot2)
df2   %>%
  mutate(word = reorder(mentions_screen_name, n)) %>%
  top_n(10) %>%
  ggplot(aes(x = word, y = n)) +

  geom_col() +
  xlab(NULL) +
  coord_flip() +
  labs(y = "Count",
       x = "mentions_screen_name",
       title = "Count of top 10 Handles mentioned by Modi",
       subtitle = "Last 1000 Tweets")
```

```
## Selecting by word
```


Count of top 10 Handles mentioned by Modi

Last 1000 Tweets

d) When were the extracted tweets reported? Plot a histogram of tweet counts with time on x-axis. Hint: ymd_hms() function may be useful. It transforms dates stored as character vectors in

year, month, day, hour, minute, second format to POSIXct objects

e) What devices/services (e.g., Web, Android, Iphone) were used to post tweets? Plot histogram by source (=device/service).

I have combined subtask D & E in Single histogram, I have plotted Histogram by hour and filled it with the source of tweets.

```
#Extract date out of timestamp
#install.packages("lubridate")
#library(lubridate)
df3 <- tmls_politics %>% mutate(created_at_date =as.Date(tmls_politics$created_at))
head(df3)
```

user_id	status_id	created_at	screen_name	▸
<chr>	<chr>	<S3: POSIXct>	<chr>	
18839785	1220177983740219392	2020-01-23 02:54:35	narendramodi	
18839785	1220177665203888128	2020-01-23 02:53:19	narendramodi	
18839785	1220176946044243971	2020-01-23 02:50:28	narendramodi	
18839785	1219991121943621632	2020-01-22 14:32:04	narendramodi	
18839785	1219991120202981377	2020-01-22 14:32:04	narendramodi	
18839785	1219824989521637377	2020-01-22 03:31:55	narendramodi	

6 rows | 1-4 of 91 columns

```
library(lubridate) #manipulate dates
```

```
## Warning: package 'lubridate' was built under R version 3.6.2
```

```
##
## Attaching package: 'lubridate'
```

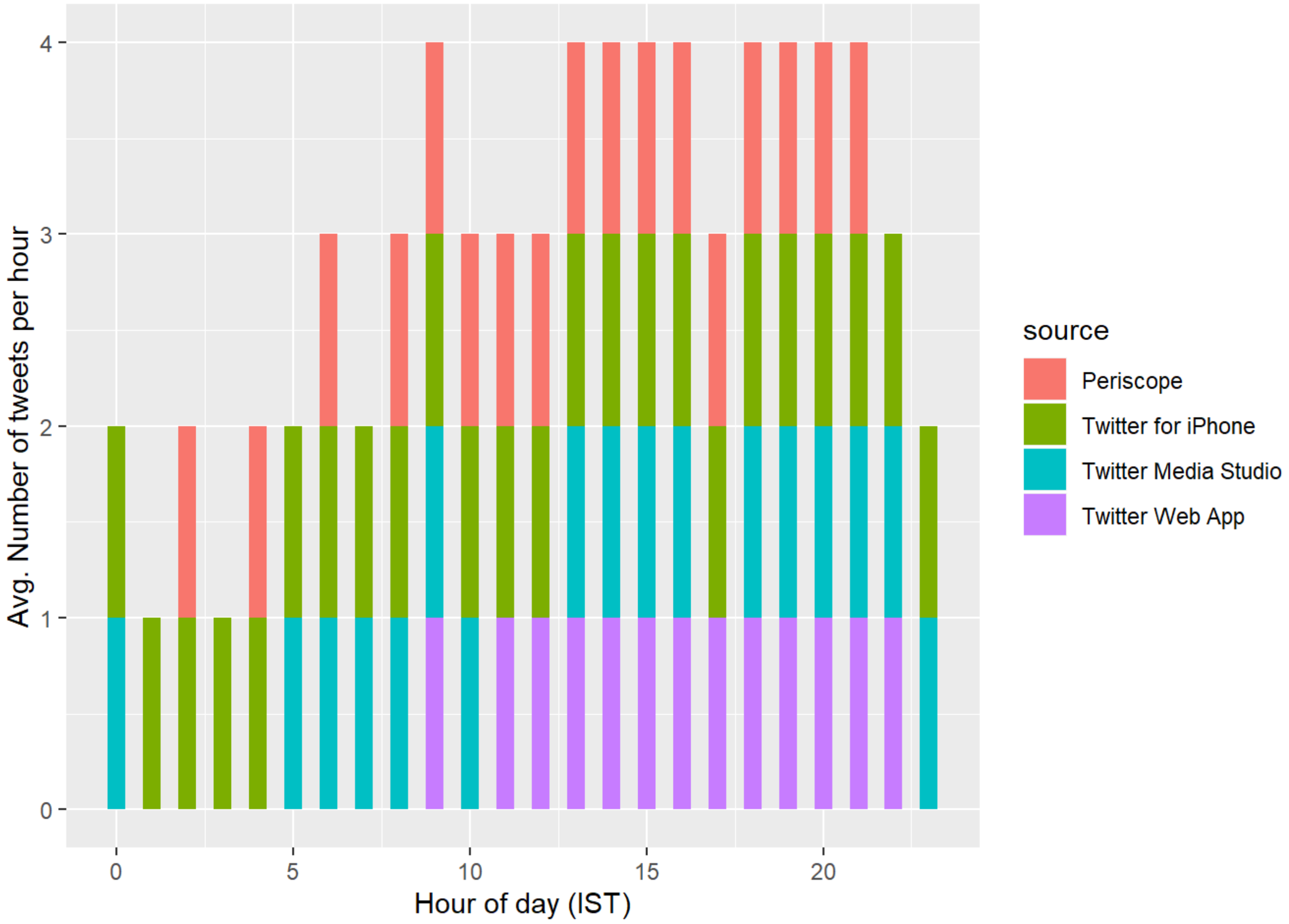
```
## The following object is masked from 'package:base':
##
##     date
```

```
library(scales) #for plotting
```

```
##
## Attaching package: 'scales'
## The following object is masked from 'package:purrr':
##
##     discard
```

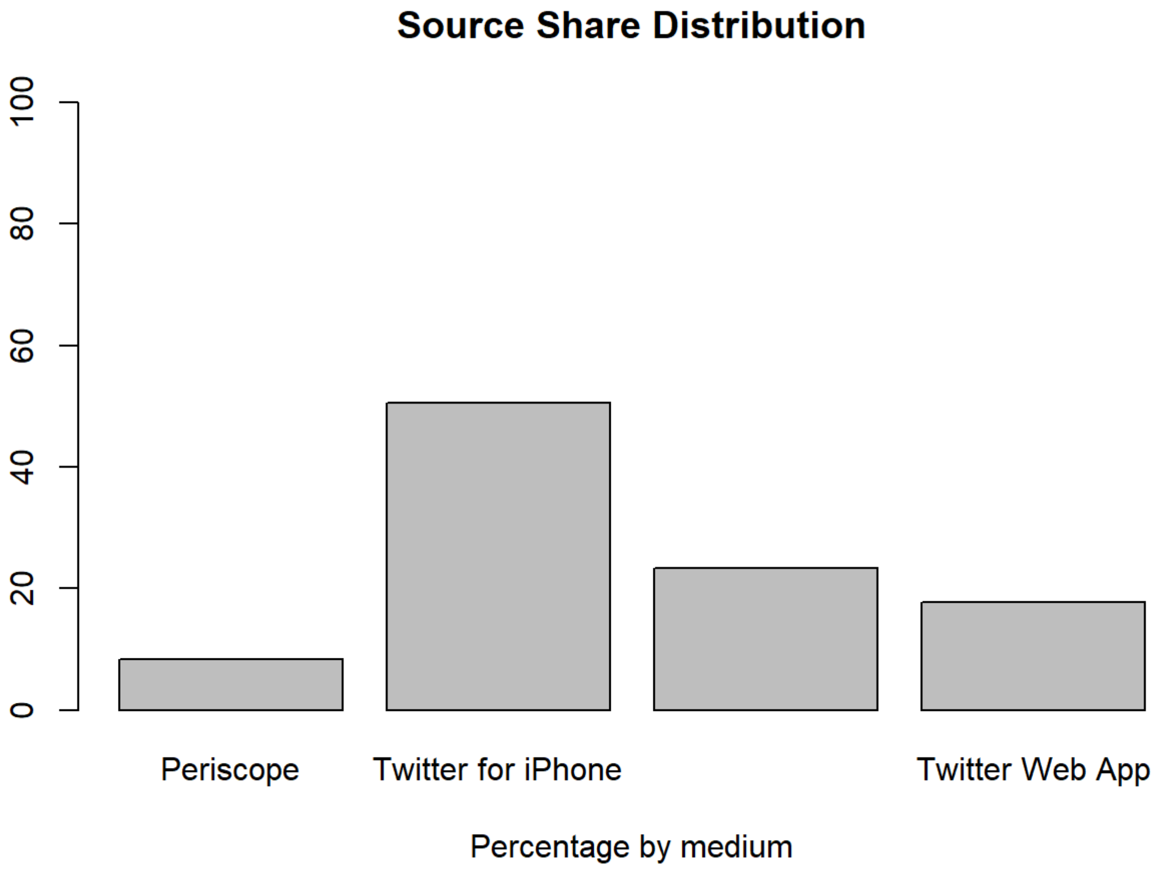
```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
#tweet time
tweets_hour <-  tmls_politics %>%
  count(source, hour = hour(with_tz(created_at, "Asia/Kolkata"))) %>% #count tweets per hour
  mutate(percent = n / sum(n)) #reformat to percent


ggplot(data=tweets_hour,aes(hour, fill = source)) + #create plot
  geom_histogram(binwidth=.5) + #add a line graph
  #scale_y_continuous(labels = percent_format()) + #format axes
  labs(x = "Hour of day (IST)", #label axes
       y = "Avg. Number of tweets per hour",
       color = "")
```



Most tweets are from Iphone followed bt Twitter Media Studio.

```
## Just for fun, i plot the bar chart of distribution of different sources below.
df_source <- tmls_politics %>%group_by(source) %>% summarise(N = n()) %>% arrange(source)
barplot((df_source$N/sum(df_source$N))*100, main="Source Share Distribution", xlab="Percentag
e by medium",names.arg=df_source$source,ylim = c(0,100) )
```



f) Inspect the content of your sample of tweets. Do necessary text transformations and clean the text as if you want to present the 1000 tweets as a word cloud. Explore the standard set of English/German stopwords, e.g. here https://github.com/arc12/Text-Mining-Weak-Signals/wiki/Standard-set-of-english-stopwords (https://github.com/arc12/Text-Mining-Weak-Signals/wiki/Standard-set-of-english-stopwords) and add at least 2 more stopwords that have in your opinion little value for your sample. Consider the language of the tweets when choosing between English/German stopwords set.

```
tweets <- tmls_politics

#removing http and https
tweets$text<-gsub("http.*","", tweets$text)
tweets$text<-gsub("https.*","", tweets$text)
# remove punctuation, convert to lowercase, add id for each tweet!
tweets_clean <- tweets%>%
dplyr::select(text) %>%
unnest_tokens(word, text)
head(tweets_clean)
```

word
<chr>
tributes
to
the
great
balasaheb
thackeray

6 rows

Stemming

```
#stemming is not always recommended with unnest_tokens()
#Option 1 wordStem() function
library(SnowballC)
tweets_clean1 <- tweets %>%
  dplyr::select(text) %>%
  unnest_tokens(word, text)%>%
  mutate(word_stem = wordStem(word))
head(tweets_clean1)
```

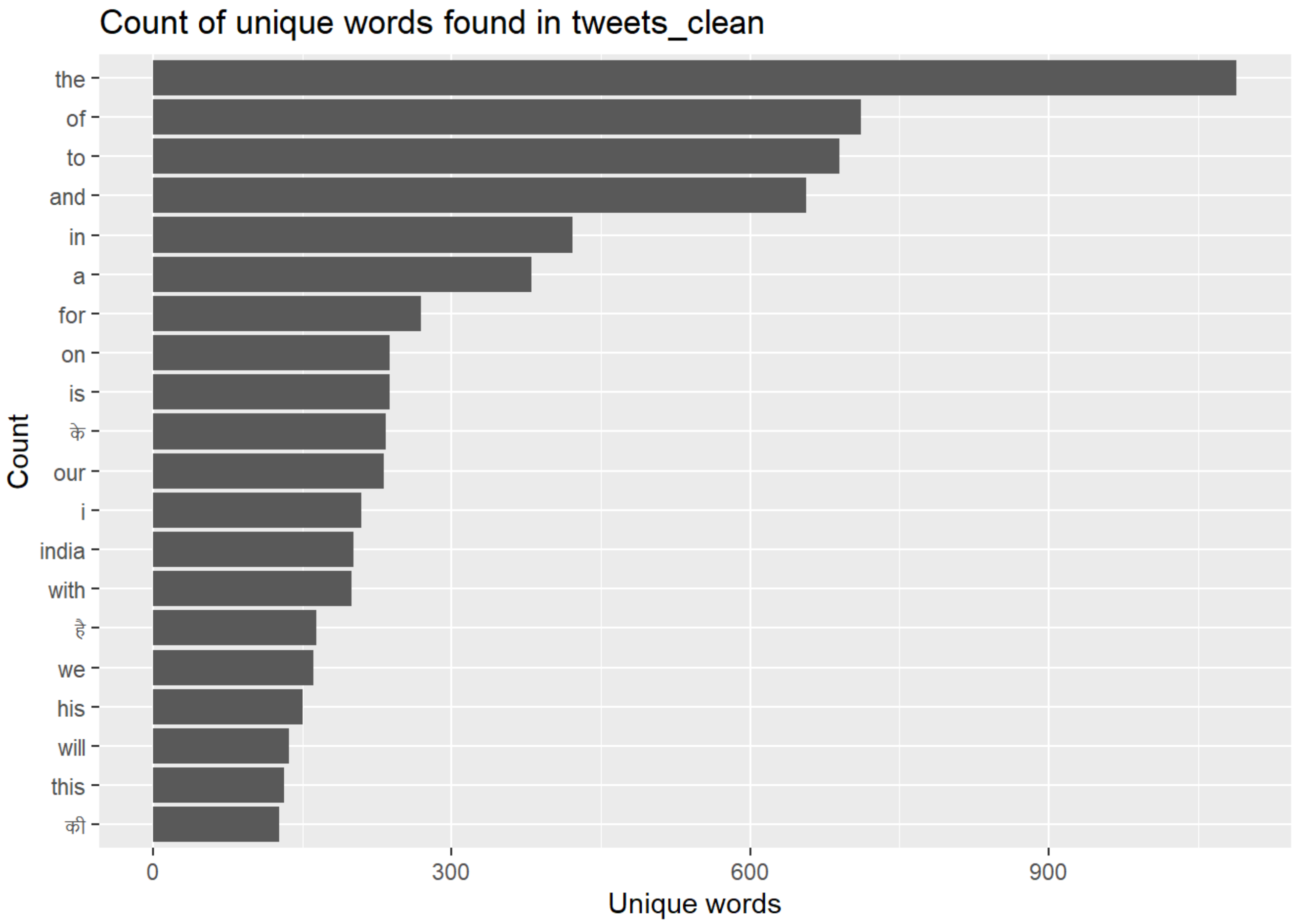
word	word_stem
<chr>	<chr>
tributes	tribut
to	to
the	the
great	great
balasaheb	balasaheb
thackeray	thackerai

6 rows

here is a plot of most frequent words without removing the stopwords. we see lot of hindi and english stopwords.

```
# plot the top 20
library (dplyr)
tweets_clean1 %>%
  count(word, sort = TRUE) %>% #Term document matrix
  top_n(20) %>%
  mutate(word = reorder(word, n)) %>%

  ggplot(aes(x = word, y = n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip() +
  labs(x = "Count",
       y = "Unique words",
       title = "Count of unique words found in tweets_clean")
```

```
## Selecting by n
```



```
stop_words_en <-get_stopwords()
stop_words_hindi <- get_stopwords(language = "hi",source = "stopwords-iso")
stop_words <- rbind.data.frame(stop_words_en,stop_words_hindi)
head(stop_words)
```

word	lexicon
<chr>	<chr>
i	snowball
me	snowball

word	**lexicon**
<chr>	<chr>
my	snowball
myself	snowball
we	snowball
our	snowball

6 rows

load list of stop words & combine the stop words from english and hindi, since tweets contain both languages

```
# remove stop words from your list of words
tweets_words <- tweets_clean1%>%
anti_join(stop_words)
```

```
## Joining, by = "word"
```

```
head(tweets_words,25)
```

word	**word_stem**
<chr>	<chr>
tributes	tribut
great	great
balasaheb	balasaheb
thackeray	thackerai
jayanti	jayanti
courageous	courag
indomitable	indomit
never	never
hesitated	hesit
raising	rais

1-10 of 25 rows Previous **1** 2 3 Next

g) Generate a term-document matrix and print out 10 most frequently used words as a table.

```
tweets_words %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  mutate(word = reorder(word, n))
```

```
## Selecting by n
```

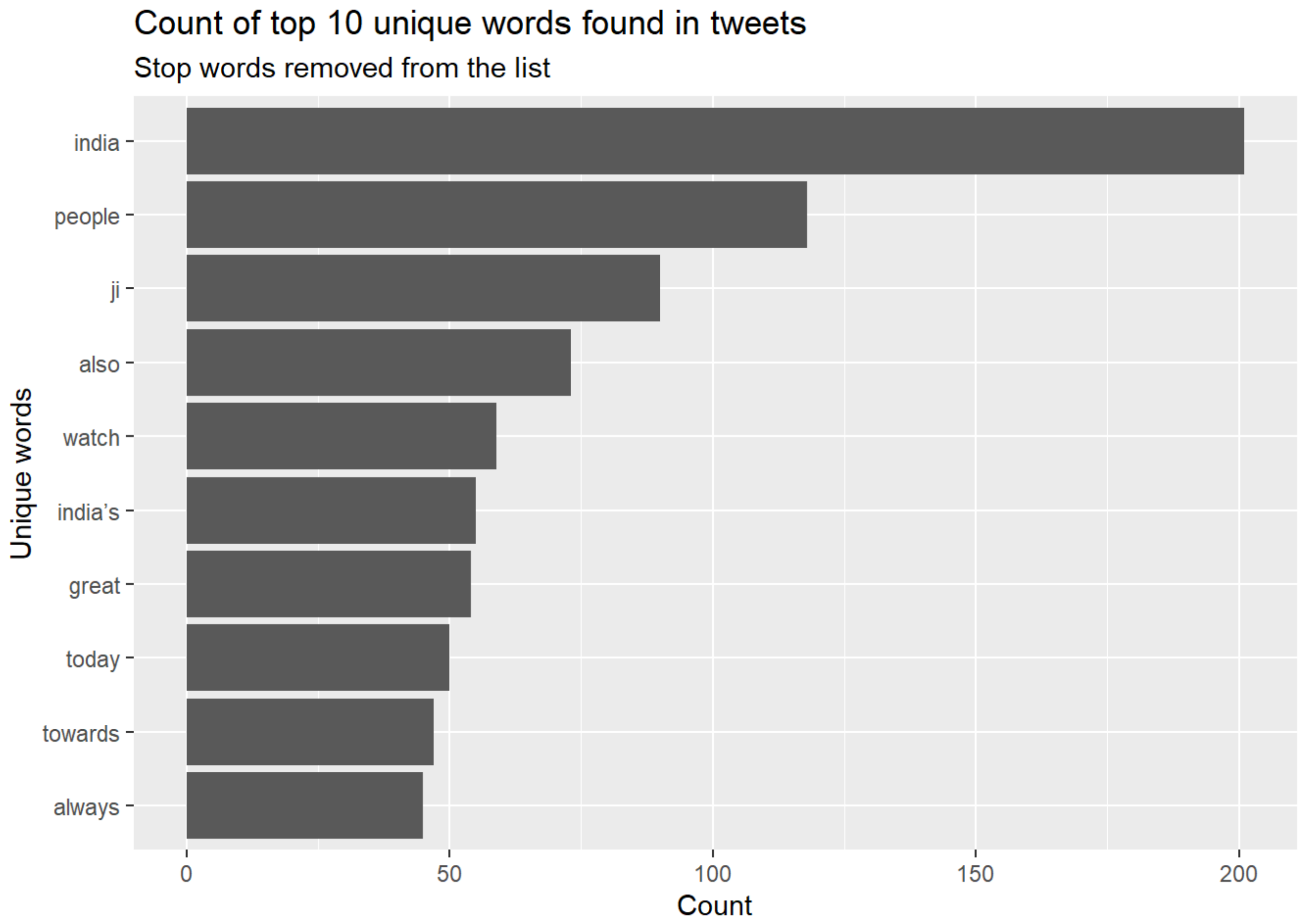
word <fctr>	n <int>
india	201
people	118
ji	90
also	73
watch	59
india’s	55
great	54
today	50
towards	47
always	45

1-10 of 10 rows

h) Create a bar plot for the 10 most frequently used words.

```
tweets_words %>%
  count(word, sort = TRUE) %>%
  top_n(10) %>%
  mutate(word = reorder(word, n)) %>%

  ggplot(aes(x = word, y = n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip() +
  labs(y = "Count",
       x = "Unique words",
       title = "Count of top 10 unique words found in tweets",
       subtitle = "Stop words removed from the list")
```

```
## Selecting by n
```



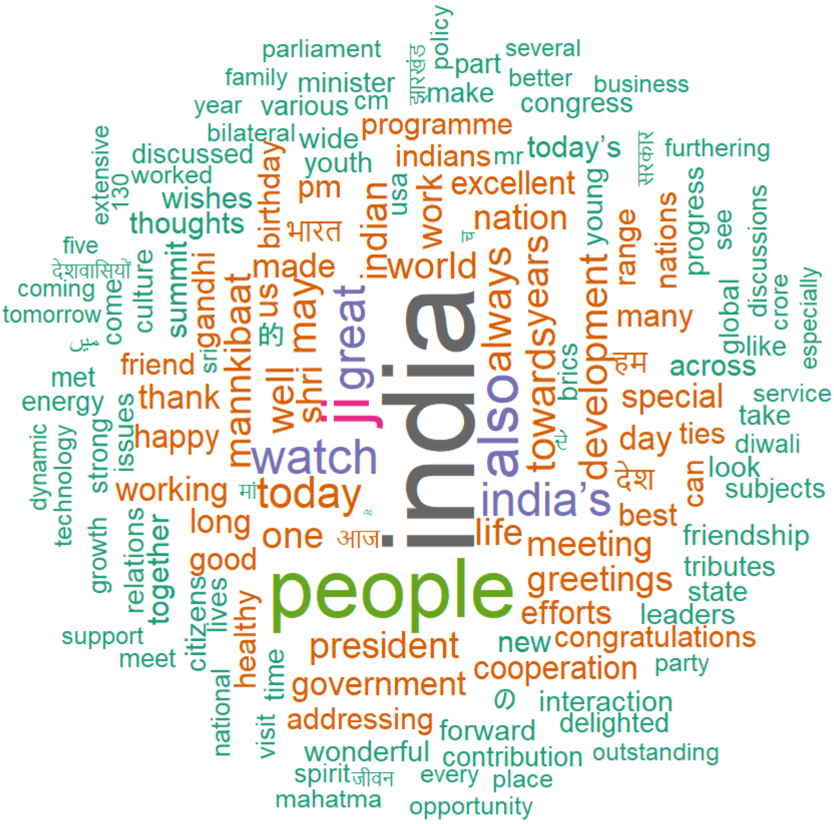
Word Cloud (Solution for f)

```
#create Term document matrix
library(tidyverse)
  words_wc <- tweets_words %>%
  count(word, sort = TRUE) %>%

  mutate(word = reorder(word, n))

library(wordcloud)
```

```
## Warning: package 'wordcloud' was built under R version 3.6.2
```

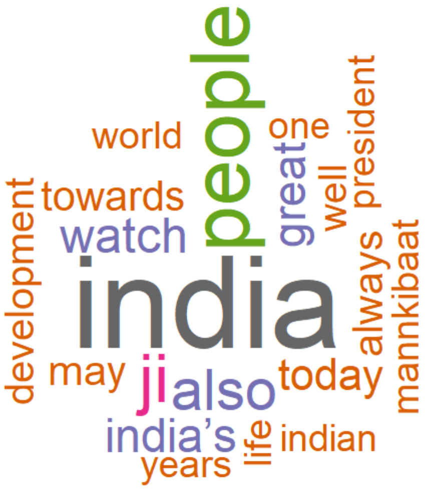
```
## Loading required package: RColorBrewer
```

```
library(RColorBrewer)
set.seed(1234) # for reproducibility

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



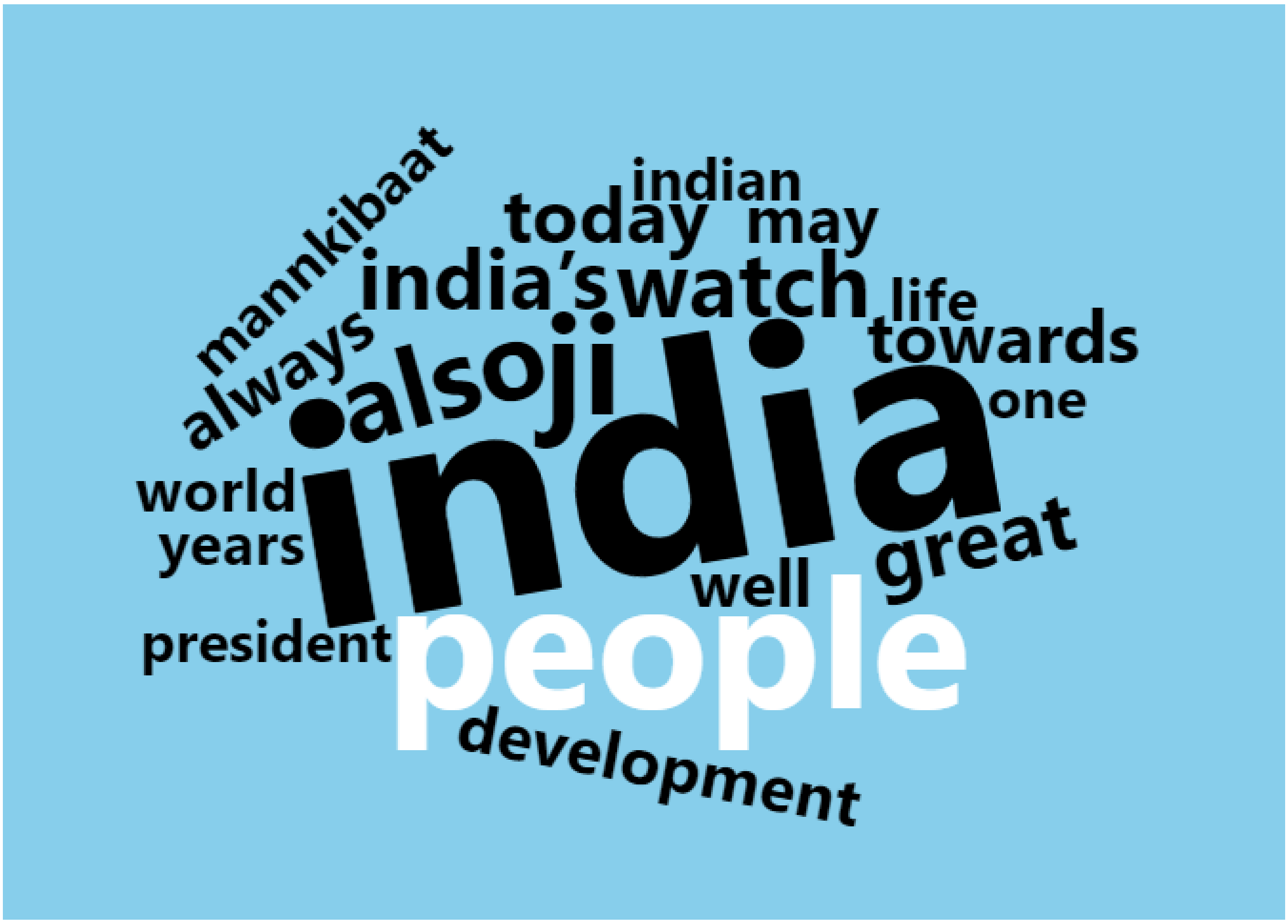
i) Generate the word cloud so that maximum number of words to be plotted is 20 and minimum frequency of the plotted word is 2. Print out 1 word cloud in the color palette of your choice and 1 word cloud in black-and-white colors.

```
wordcloud(words = words_wc$word, freq = words_wc$n, min.freq = 2, max.words=20, random.order=
FALSE, rot.per=0.35,colors=brewer.pal(8, "Dark2"))
```



```
library(wordcloud2)
#top 20 words
wordcloud2(data=words_wc %>% top_n(20), size = 0.9, color = c("black","white"),backgroundColo
r = "skyblue")
```

```
## Selecting by n
```



Exercise 2

a) Select a tweeter user, somehow related to a twitter user of your interest in Task 1. It may be a friend, business partner, rival / competitor, advisor or any other person from a similar field. E.g. Cristiano Ronaldo and Lionel Messi, Kim Kardashian and Beyoncé. Shortly comment on your choice. Scrape 1000 tweets of this second user.

I am selecting “Rahul Gandhi” as second user, he is main opponent of first user, i.e. Narendra Modi.

```
tweets_rg <- get_timeline("RahulGandhi",n = 1000)
head(tweets_rg)
```

user_id	status_id	created_at	screen_name
<chr>	<chr>	<S3: POSIXct>	<chr>
3171712086	1220257424155627525	2020-01-23 08:10:15	RahulGandhi
3171712086	1219939498391265280	2020-01-22 11:06:56	RahulGandhi
3171712086	1219242429804617729	2020-01-20 12:57:02	RahulGandhi
3171712086	1219183878125961217	2020-01-20 09:04:22	RahulGandhi
3171712086	1218045521157185537	2020-01-17 05:40:57	RahulGandhi

user_id <chr>	status_id <chr>	created_at <S3: POSIXct>	screen_name <chr>
3171712086	1217789363460820992	2020-01-16 12:43:04	RahulGandhi

6 rows | 1-4 of 90 columns

b) Inspect the content, do necessary text transformations and clean the text using the standard set of English (German) stopwords thus preparing to present word clouds for both speeches. Print out the word cloud for the tweets of the 1st user in red colors and the word cloud for the 2nd user in blue colors, set maximum number of words to be plotted and minimum frequency by yourself. Shortly #comment on your choices.

Preproccessing

```
#removing http and https
tweets_rg$text<-gsub("http.*","", tweets_rg$text)
tweets_rg$text<-gsub("https.*","", tweets_rg$text)

## Stemming
tweets_clean_rg1 <- tweets_rg %>%
  dplyr::select(text) %>%
  unnest_tokens(word, text)%>%
  mutate(word_stem = wordStem(word))
tail(tweets_clean_rg1)
```

word <chr>	word_stem <chr>
for	for
your	your
wishes	wish
lal	lal
thanhawla	thanhawla
ji	ji

6 rows

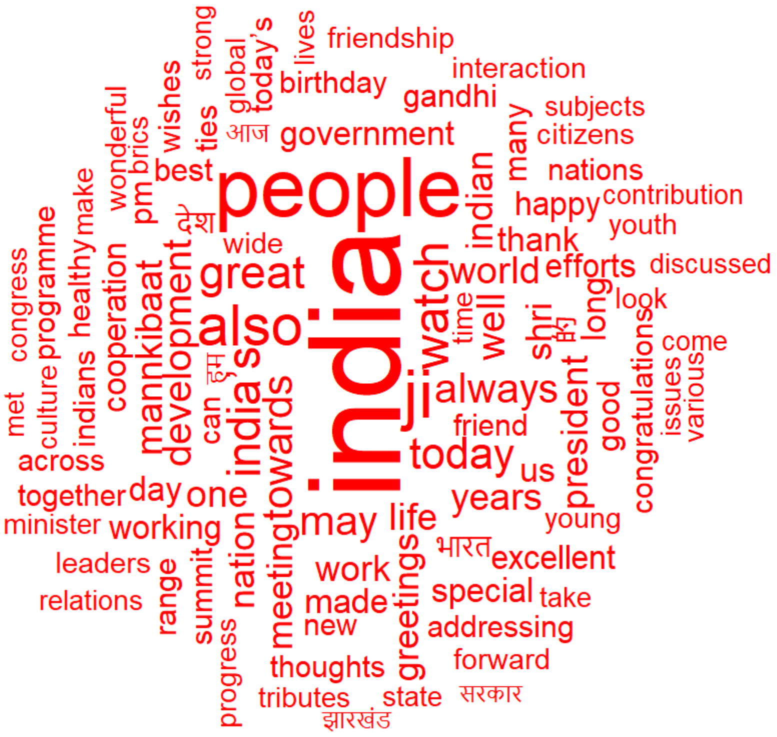
```
# remove stop words from your list of words
tweets_words_rg <- tweets_clean_rg1%>%
anti_join(stop_words)
```

```
## Joining, by = "word"
```

Plot Wordclouds for both users

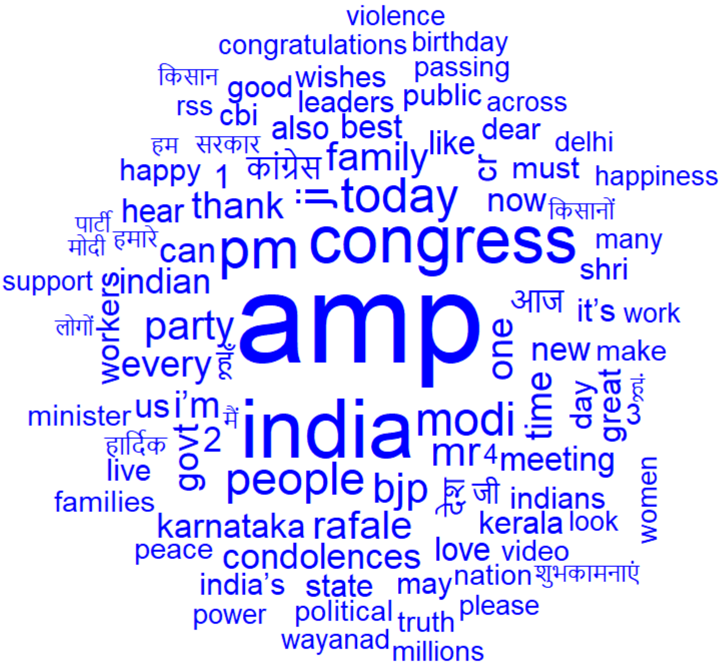
Narendra Modi

```
wordcloud(words = words_wc$word, freq = words_wc$n, min.freq = 20, max.words=100, random.orde
r=FALSE, rot.per=0.35,colors=c("red"))
```



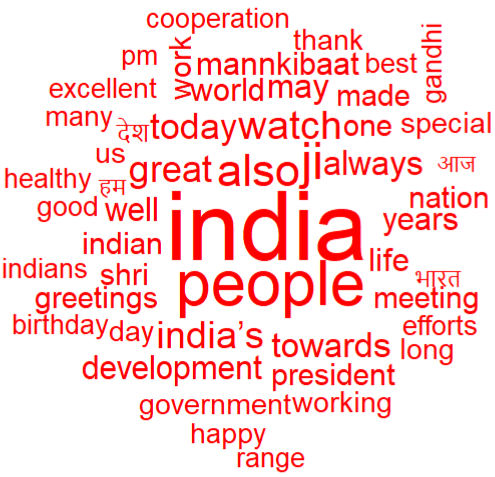
Rahul Gandhi

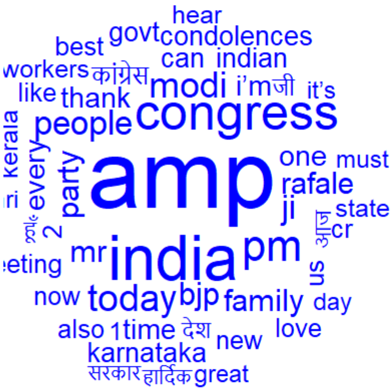
```
#create Term document matrix
library(tidyverse)
  words_wc_rg <- tweets_words_rg %>%
  count(word, sort = TRUE) %>%

  mutate(word = reorder(word, n))
wordcloud(words = words_wc_rg$word, freq = words_wc$n, min.freq =20, max.words=100, random.or
der=FALSE,colors=c("blue"))
```



```
library(wordcloud)
library(RColorBrewer)
set.seed(1234) # for reproducibility
```

```
#Create two panels to add the word clouds to
par(mfrow=c(1,2))
wordcloud(words = words_wc$word, freq = words_wc$n, min.freq = 10, max.words=50, random.order=FALSE,  scale=c(3, .5),colors=c("red"))
wordcloud(words = words_wc_rg$word, freq = words_wc$n, min.freq = 8, max.words=50, random.order=FALSE, scale=c(3, .4),colors=c("blue"))
```





c) Generate a term-document matrix for the tweets of each user and print out 6 most frequently used words for the tweets of each user.

```
#create Term document matrix - Rahul Gandhi
  words_wc_rg <- tweets_words_rg %>%
  count(word, sort = TRUE) %>%
  mutate(word = reorder(word, n))
head(words_wc_rg)
```

word <fctr>	n <int>
amp	482
india	149
congress	125
pm	122
today	113
modi	98

6 rows

```
#create Term document matrix - Narendra Modi
  words_wc <- tweets_words %>%
  count(word, sort = TRUE) %>%
  mutate(word = reorder(word, n))
head(words_wc)
```

word <fctr>	n <int>
india	201
people	118
ji	90
also	73
watch	59
india’s	55

6 rows

d) Go to https://rpubs.com/brandonkopp/creating-word-clouds-in-r (https://rpubs.com/brandonkopp/creating-word-clouds-in-r) or use any other source to get acquainted with Comparison Cloud. Generate Comparison Cloud using comparison.cloud() function for the tweets of 2 users. Set the argument max.words on your own. Shortly #comment on the output, e.g.: “#From the comparison cloud, we can see that issues like Iraq were more front-and-center in 2008 than in 2016. We also see ISIL, which didn’t exist (at least by that name) in 2008, pop up in President Obama’s speech. “Change” was used more by President Obama and, interestingly, “hope” was used more often in President Bush’s 2008 speech“

```
## first we need to create combined term document matrix for both these users
words_Modi<- words_wc %>% mutate(Modi = n)
drops <- c("n")
words_Modi <- words_Modi[ , !(names(words_Modi) %in% drops)]
```

```
head(words_Modi)
```

word <fctr>	Modi <int>
india	201
people	118
ji	90

word <fctr>	Modi <int>
also	73
watch	59
india’s	55

6 rows

```
words_Raga<- words_wc_rg %>% mutate(RAGA = n)
drops <- c("n")
words_Raga <- words_Raga[ , !(names(words_Raga) %in% drops)]
head(words_Raga)
```

word <fctr>	RAGA <int>
amp	482
india	149
congress	125
pm	122
today	113
modi	98

6 rows

```
#Create single term document matrix
Words_both <- merge(words_Modi,words_Raga,by = 'word')

tail(Words_both)
```

	word <fctr>	Modi <int>	RAGA <int>
2204	you’re	1	7
2205	young	24	14
2206	youngsters	11	3
2207	youth	24	20
2208	zealand	1	1
2209	zero	1	1

6 rows

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(tm)
```

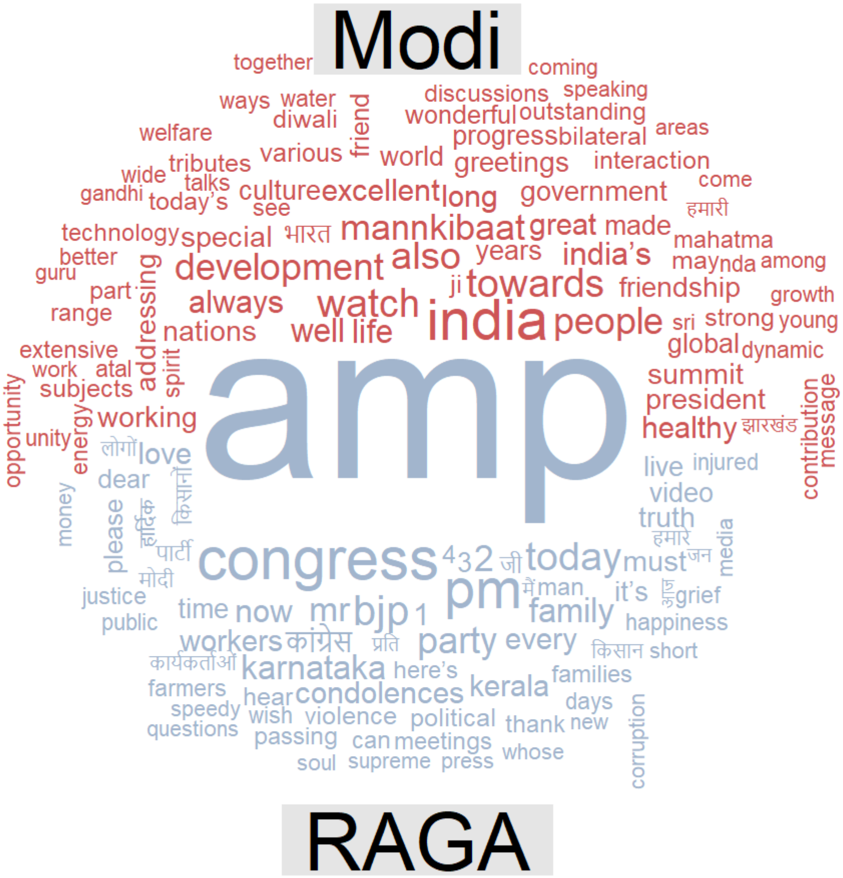
```
## Warning: package 'tm' was built under R version 3.6.2
```

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

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

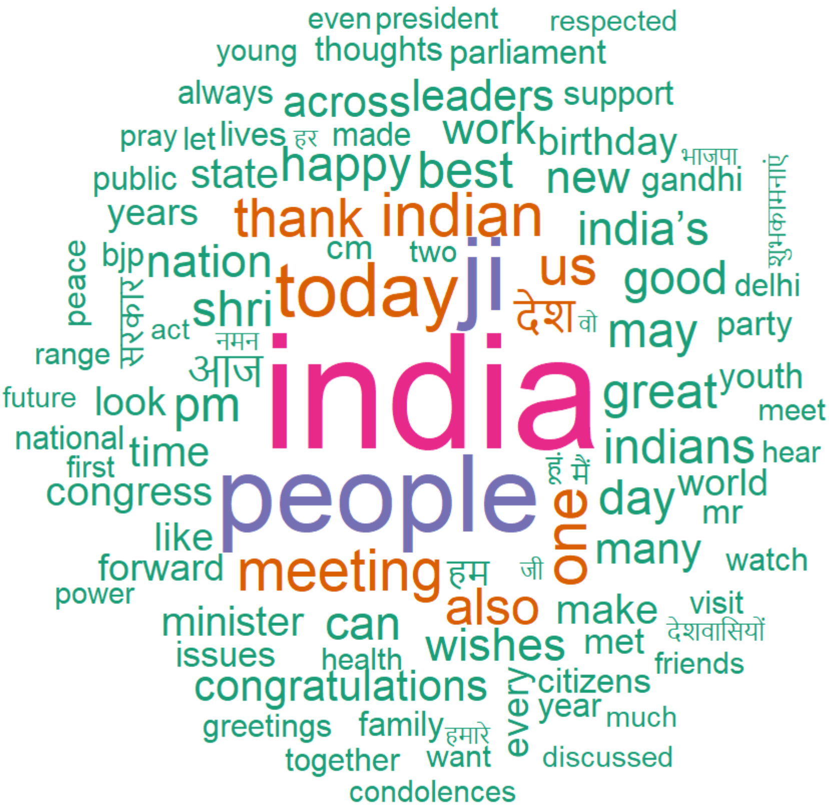
```
library(dplyr)
library(wordcloud)
Words_both$word <-as.character(Words_both$word)

df_n <- Words_both %>% gather()
corpus_my<-Corpus(VectorSource(df_n))
#tdm<-as.matrix(TermDocumentMatrix(corpus_my))
df2<-Words_both %>%
  gather("Origin","Freq",c(2,3)) %>%
  acast(word~Origin,fill=0,value.var = "Freq")
comparison.cloud(df2, random.order=FALSE, colors = c("indianred3","lightsteelblue3"),
                 title.size=2.5,max.words=150,scale = c(7,.5))
```



we could notice that Being Prime minister, Modi has more often used words like ‘India’, ‘People’ etc. compared to Rahul Gandhi. While rahul Gandhi has used words like ‘BJP’, ‘PM’, ‘Modi’, ‘Congress’ etc as he is in opposition and need to oppose the ruling party on varios issues.

e) Get acquainted with Commonality Cloud. Generate Commonality Cloud for the tweets of 2 users. Shortly comment on the output.

```
library(RColorBrewer)
commonality.cloud(df2, random.order=FALSE, scale=c(5, .5),colors = brewer.pal(4, "Dark2"), ma
x.words=100)
```



Both leaders use expected words like ‘India’, Poeple, indian,nation, names of the both political parties.

Exercise 3.

Create a word cloud of a web page: https://www.uni-potsdam.de/de/social-media-krasnova.html (https://www.uni-potsdam.de/de/social-media-krasnova.html)

Do text cleaning, if necessary. Set the color palette, maximum number of words to be plotted and minimum frequency by yourself. Shortly comment on the output.

```
## function is taken from here: http://www.sthda.com/english/wiki/print.php?id=159

rquery.wordcloud <- function(x, type=c("text", "url", "file"),
                          lang="de", excludeWords=NULL,
                          textStemming=FALSE,  colorPalette="Dark2",
                          min.freq=3, max.words=200)
{
  library("tm")
  library("SnowballC")
  library("wordcloud")
  library("RColorBrewer")

  if(type[1]=="file") text <- readLines(x)
  else if(type[1]=="url") text <- html_to_text(x)
  else if(type[1]=="text") text <- x

  # Load the text as a corpus
  docs <- Corpus(VectorSource(text))
  # Convert the text to lower case
  docs <- tm_map(docs, content_transformer(tolower))
  # Remove numbers
  docs <- tm_map(docs, removeNumbers)
  # Remove stopwords for the language
  docs <- tm_map(docs, removeWords, stopwords(lang))
  # Remove punctuations
  docs <- tm_map(docs, removePunctuation)
  # Eliminate extra white spaces
  docs <- tm_map(docs, stripWhitespace)
  # Remove your own stopwords
  if(!is.null(excludeWords))
    docs <- tm_map(docs, removeWords, excludeWords)
  # Text stemming
  if(textStemming) docs <- tm_map(docs, stemDocument)
  # Create term-document matrix
  tdm <- TermDocumentMatrix(docs)
  m <- as.matrix(tdm)
  v <- sort(rowSums(m),decreasing=TRUE)
  d <- data.frame(word = names(v),freq=v)
  # check the color palette name
  if(!colorPalette %in% rownames(brewer.pal.info)) colors = colorPalette
  else colors = brewer.pal(8, colorPalette)
  # Plot the word cloud
  set.seed(1234)
  wordcloud(d$word,d$freq, min.freq=min.freq, max.words=max.words,
            random.order=FALSE, rot.per=0.35,
            use.r.layout=FALSE, colors=colors)

  invisible(list(tdm=tdm, freqTable = d))
}
#++++++++++++++++++++++
# Helper function
#++++++++++++++++++++++
# Download and parse webpage
html_to_text<-function(url){
  library(RCurl)
  library(XML)
  # download html
  html.doc <- getURL(url)
  #convert to plain text
  doc = htmlParse(html.doc, asText=TRUE)
 # "//text()" returns all text outside of HTML tags.
 # We also don’t want text such as style and script codes
  text <- xpathSApply(doc, "//text()[not(ancestor::script)][not(ancestor::style)][not(ancesto
r::noscript)][not(ancestor::form)]", xmlValue)
  # Format text vector into one character string
  return(paste(text, collapse = " "))
}
```

```
#install.packages(c( "RCurl", "XML"))
library(RCurl)
```

```
## Warning: package 'RCurl' was built under R version 3.6.2
```

```
## Loading required package: bitops
```

```
## 
## Attaching package: 'RCurl'
```

```
## The following object is masked from 'package:tidyr':
## 
##     complete
```

```
library(XML)
url = "https://www.uni-potsdam.de/de/social-media-krasnova.html"
rquery.wordcloud(x=url, type="url", min.freq = 3, max.words = 110)
```

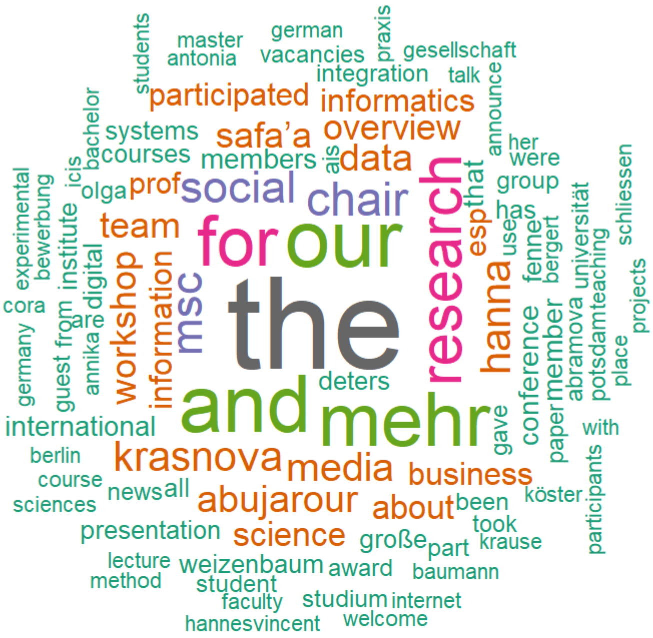
```
## Warning in tm_map.SimpleCorpus(docs, content_transformer(tolower)):
## transformation drops documents
```

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

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

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

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



as the URL text is in german. we applied stopwords of german. this results in english stopwords like “the” and “and” being included.if we set minimum freq. to 5 & 10, there are very less words in the cloud so i have set the min. freq as 3 to include more words.