Tweets_Extract_BagOfWords

```r
library(rtweet) # harvesting tweets
```

```
## Warning: package 'rtweet' was built under R version 4.1.3
```

```r
library(tm) # text mining
```

```
## Warning: package 'tm' was built under R version 4.1.3
```

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

```r
#library(dplyr)
#library(tidyr)
```

First accessing the file that contains the api and access keys:

```r
Credential = readr:: read_csv("C:\\Users\\dhaka\\OneDrive\\Desktop\\PhD Studies\\2nd Sem\\Data Prep\\Ass4\\TwitterAPICredential.csv")
```

```
## New names:
## Rows: 1 Columns: 8
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (7): API, APIKeySecret, BearerToken, ClientID, ClientSecret, AccessToken... lgl
## (1): ...6
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...6`
```

Setting up connection from Twitter API using create_token function from rtweet package

```r
API = Credential$API
APISecretKey = Credential$APIKeySecret
AccessToken = Credential$AccessToken
AccessTokenSecret = Credential$AccessTokenSecret

Token = create_token(
  app = "PradipWorkOnTextAnalytics",
  consumer_key = API,
  consumer_secret = APISecretKey,
  access_token = AccessToken,
  access_secret = AccessTokenSecret
  )
```

Observing the current ongoing conflict between Russia and Ukraine, I decided to extract the tweets from the president of Russia and president of Ukraine.

Extracting tweets of the president of Ukraine @ZelenskyyUa

```r
PresidentUkraine <- get_timeline(
  user = "@ZelenskyyUa",
  n = 2200
)

write_as_csv(
  x = PresidentUkraine,
  file_name = "C:\\Users\\dhaka\\OneDrive\\Desktop\\PhD Studies\\2nd Sem\\Data Prep\\Ass4\\PresidentUkraine.csv"
)
```

Extracting tweets of the president of the Russia @KremlinRussia_E

```r
PresidentRussia <- get_timeline(
  user = "@KremlinRussia_E",
  n = 2200
)

write_as_csv(
  x = PresidentRussia,
  file_name = "C:\\Users\\dhaka\\OneDrive\\Desktop\\PhD Studies\\2nd Sem\\Data Prep\\Ass4\\PresidentRussia.csv"
)
```

Loading the data:

```r
setwd("C:\\Users\\dhaka\\OneDrive\\Desktop\\PhD Studies\\2nd Sem\\Data Prep\\Ass4")
tweets <- c(
  'PresidentUkraine.csv',
  'PresidentRussia.csv'
)
list_tweets <- lapply(
  X = tweets,
  FUN = read.csv,
  colClasses = "character"
)

#PresidentUkraine = list_tweets[1]
PresidentRussia = list_tweets[2]
PresidentRussia = as.data.frame(PresidentRussia)

names(list_tweets) <- tweets
```

Data Prep:

One thing I noticed on the tweets of the president of the Ukraine is that he has tweeted in English as well as in the Ukrainian language. We need to remove the one in Ukrainian language. In the data set above, the tweets in the Ukrainian language are in weird format (<U+0413><U+043E><U+043B><U+043E><U+043A><U+043E>………). I think this is because R and its packages can not read non English alphabets.

Removing the rows in the Ukrainian language.

I noticed most of the Ukrainian language tweets are starting with '<U+' in the text column value. First, I will be find the rows for which the text column value starts with '<U+'. Next, I will delete these rows from the dataset.

```r
library(stringr)

RowIndex = c()
#Function to remove tweets in Ukrainian

RemoveUkranian <- function(textarg)
  {
    First3Strings = str_sub(textarg, 1, 3)
      for (i in 1:length(First3Strings))
        {
          RowIndex = which(First3Strings == '<U+')
        }
    return(RowIndex)
  }

#Passing the text column value (one that contains all tweets) in the function

RemovingRowIndex = RemoveUkranian(list_tweets[[1]][[5]]) # list_tweets[[1]][[5]]: 5th column from Dataframe 1.

#Now we delete the rows that contains the tweets in Ukranian language

PresidentUkraineUpdated = list_tweets[[1]][-RemovingRowIndex,]
dim(PresidentUkraineUpdated)
```

```
## [1] 975  90
```

I was able to reduce 1959 rows of tweets to 846 rows of tweets excluding the ones in Ukrainian language. We can still see few rows of tweets in Ukrainian while skimming through the data set.

While skimming through the dataset, I noticed that the weird formatted Ukrainian tweets are very lengthy as compared to the English tweets.

So I am going to delete rows with very lengthy text column value. (FYI: Tweets are in the text column value)

I feel removing rows with text column value over 750 will remove the Ukrainian tweets without removing the English tweets.

```r
RowIndex2 = c()
#Function to remove very lengthy tweets (tweets in Ukrainian language)

RemoveLengthyTweets <- function(textarg)
  {
      for (i in 1:length(textarg))
        {
          RowIndex2 = which(nchar(textarg) > 750)
        }
    return(RowIndex2)
  }

#Passing the text column value (one that contains all tweets) in the function

RemovingRowIndex2 = RemoveLengthyTweets(PresidentUkraineUpdated$text)

#Now we delete the rows that contains the very lengthy tweets (tweets in Ukrainian language)

PresidentUkraineUpdated2 = PresidentUkraineUpdated[-RemovingRowIndex2,]
dim(PresidentUkraineUpdated2)
```

```
## [1] 923  90
```

Now, I believe we have obtained the clean data. We have reduced from 1959 rows of mixed English and Ukrainian tweets to 799 rows of English tweets from the president of Ukraine.

String Manipulation for text analytics

Lets combine the data from both presidents by rows

```r
CombinedTweets <- dplyr::bind_rows(PresidentUkraineUpdated2, PresidentRussia)
#dim(CombinedTweets)
```

Converting all letters to lowercase.

```r
CombinedTweets$text <- stringr::str_to_lower(CombinedTweets$text)
```

Remove websites hash tags, person tags, emoji, and email.

```r
CombinedTweets$text <- qdapRegex::rm_twitter_url(
  CombinedTweets$text,
  replacement = " ",
  clean = TRUE
)
CombinedTweets$text <- qdapRegex::rm_url(
  CombinedTweets$text,
  replacement = " ",
  clean = TRUE
)
CombinedTweets$text <- qdapRegex::rm_hash(
  CombinedTweets$text,
  replacement = " ",
  clean = TRUE
)
CombinedTweets$text <- qdapRegex::rm_tag(
  CombinedTweets$text,
  replacement = " ",
  clean = TRUE
)
CombinedTweets$text <- qdapRegex::rm_emoticon(
  CombinedTweets$text,
  replacement = " ",
  clean = TRUE
)
CombinedTweets$text <- qdapRegex::rm_email(
  CombinedTweets$text,
  replacement = " ",
  clean = TRUE
)
```

Removing special characters that are particular to twitter.

```r
CombinedTweets$text <- qdapRegex::rm_between(
  CombinedTweets$text,
  left = "<",
  right = ">",
  replacement = " ",
  clean = TRUE
)
CombinedTweets$text <- stringr::str_replace_all(
  string = CombinedTweets$text,
  pattern = "â€œ",
  replacement = " "
)
CombinedTweets$text <- stringr::str_replace_all(
  string = CombinedTweets$text,
  pattern = "â€™",
  replacement = "'"
)
```

Removing punctuation, digits, and non-word characters with spaces

```r
CombinedTweets$text <- stringr::str_replace_all(
  string = CombinedTweets$text,
  pattern = "[:punct:]",
  replacement = " "
)
CombinedTweets$text <- stringr::str_replace_all(
  string = CombinedTweets$text,
  pattern = "[:digit:]",
  replacement = " "
)
CombinedTweets$text <- stringr::str_replace_all(
  string = CombinedTweets$text,
  pattern = "\\W",
  replacement = " "
)
```

Replacing contractions if any exists

```r
CombinedTweets$text <- stringr::str_replace_all(
  string = CombinedTweets$text,
  pattern = " i'm ",
  replacement = " i am "
)
CombinedTweets$text <- stringr::str_replace_all(
  string = CombinedTweets$text,
  pattern = "'re ",
  replacement = " are "
)
CombinedTweets$text <- stringr::str_replace_all(
  string = CombinedTweets$text,
  pattern = "'t ",
  replacement = " not "
)
CombinedTweets$text <- stringr::str_replace_all(
  string = CombinedTweets$text,
  pattern = "'ve ",
  replacement = " have "
)
CombinedTweets$text <- stringr::str_replace_all(
  string = CombinedTweets$text,
  pattern = "'ll ",
  replacement = " will "
)
```

Removing stop words

```r
CombinedTweets$text <- tm::removeWords(
  x = CombinedTweets$text,
  words = tm::stopwords(kind = "SMART")
)
CombinedTweets$text <- tm::removeWords(
  x = CombinedTweets$text,
  words = tm::stopwords(kind = "english")
)
CombinedTweets$text <- tm::removeWords(
  x = CombinedTweets$text,
  words = qdapDictionaries::Top200Words
)
```

Getting rid of extra white space.

```r
CombinedTweets$text <- trimws(stringr::str_replace_all(
  string = CombinedTweets$text,
  pattern = "\\s+",
  replacement = " "
))
```

Now we split all the strings from all tweets and reduce the words to their stem. Later we again concatenate them back.

```r
library(stringr)
CombinedTweetsStrings <- strsplit(CombinedTweets$text," ") # This converts the whole sentence of tweets into individual strings for all row of tweets.
# "I am a programmer" -> "I" "am" "a" "programmer"

CombinedTweetsStrings <- lapply(
  X = CombinedTweetsStrings,
  FUN = tm::stemDocument)

# Concatenating stem words

CombinedTweetsStrings <- lapply(
  X = CombinedTweetsStrings,
  FUN = str_c,
  collapse = " " #"I" "am" "a" "programmer" -> "I am a programmer"
)

CombinedTweets$text <- unlist(CombinedTweetsStrings)
```

Next I will only use the user_id, screen_name and text and create a new dataframe.

```r
CleanedTweets = CombinedTweets[,c(1,4,5)]
PresidentUkraineTweets = CombinedTweets[1:799, 5]
PresidentRussiaTweets = CombinedTweets[800:2999, 5]

CleanCombinedTweets = CombinedTweets[,5]
```

Document-Term Matrix

Converting tweets to a Corpus.

```r
CorpusTweetsUkraine <- tm::VCorpus(tm::VectorSource(PresidentUkraineTweets))
CorpusTweetsRussia <- tm::VCorpus(tm::VectorSource(PresidentRussiaTweets))

CorpusTweetsCombined <- tm::VCorpus(tm::VectorSource(CleanCombinedTweets))
```

Creating a document-term matrix.

```r
DocTermMatTweetsUkraine <- tm::DocumentTermMatrix(CorpusTweetsUkraine)
DocTermMatTweetsRussia <- tm::DocumentTermMatrix(CorpusTweetsRussia)

DocTermMatTweetsCombined <- tm::DocumentTermMatrix(CorpusTweetsCombined)
```

Removing sparse terms.

```r
DocTermMatTweetsUkraine <- tm::removeSparseTerms(
  DocTermMatTweetsUkraine,
  0.995
)

DocTermMatTweetsRussia <- tm::removeSparseTerms(
  DocTermMatTweetsRussia,
  0.995
)

DocTermMatTweetsCombined <- tm::removeSparseTerms(
  DocTermMatTweetsCombined,
  0.995
)
dim(DocTermMatTweetsCombined)
```

```
## [1] 3120  417
```

Creating integer matrix equivalent to the document-term matrix.

```r
IntegerMatrixUkraine <- as.matrix(DocTermMatTweetsUkraine)
dim(IntegerMatrixUkraine)
```

```
## [1] 799 648
```

```r
IntegerMatrixRussia <- as.matrix(DocTermMatTweetsRussia)
dim(IntegerMatrixRussia)
```

```
## [1] 2200  297
```

```r
IntegerMatrixCombined <- as.matrix(DocTermMatTweetsCombined)
dim(IntegerMatrixCombined)
```

```
## [1] 3120  417
```

Visualizations

Visualize tweets from each presidents with a word cloud.

```r
#Combined
TermFrequencyCombined <- data.frame(
  Term = colnames(IntegerMatrixCombined),
  Frequency = colSums(IntegerMatrixCombined),
  stringsAsFactors = FALSE
)
TermFrequencyCombined <- TermFrequencyCombined[order(TermFrequencyCombined$Frequency),]
wordcloud::wordcloud(
  words = TermFrequencyCombined$Term,
  freq = TermFrequencyCombined$Frequency,
  max.words = 25,
  random.order = FALSE,
  colors = viridis::viridis(100)
)
title('Word Cloud for tweets from both of the presidents')
```

Word Cloud for tweets from both of the presidents


```r
#Ukraine
TermFrequencyUkraine <- data.frame(
  Term = colnames(IntegerMatrixUkraine),
  Frequency = colSums(IntegerMatrixUkraine),
  stringsAsFactors = FALSE
)
TermFrequencyUkraine <- TermFrequencyUkraine[order(TermFrequencyUkraine$Frequency),]
wordcloud::wordcloud(
  words = TermFrequencyUkraine$Term,
  freq = TermFrequencyUkraine$Frequency,
  max.words = 25,
  random.order = FALSE,
  colors = viridis::viridis(100)
)
title('Word Cloud for tweets from president of Ukraine')
```

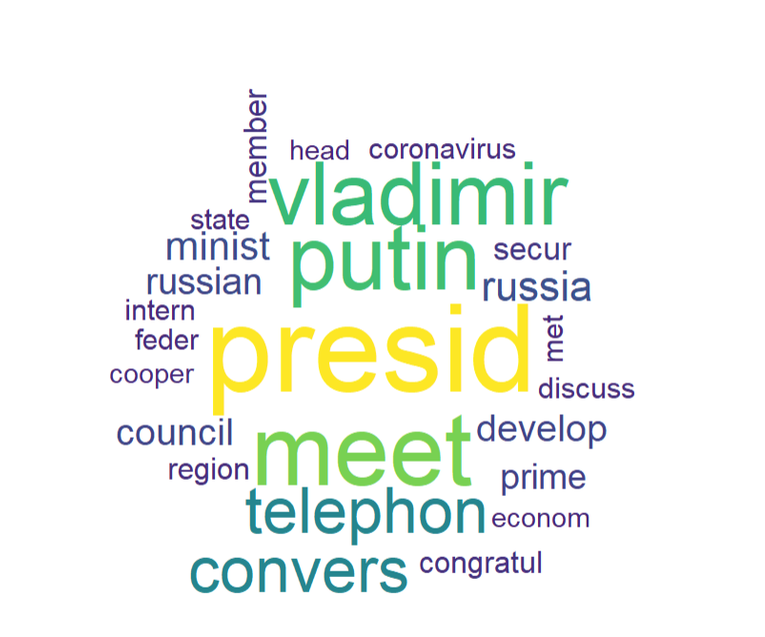
Word Cloud for tweets from president of Ukraine


```r
#Down sampling Ukrainian terms in equivalent to Russia. i.e, downsampling to 281 terms

TermFrequencyUkraine <- tail(TermFrequencyUkraine, n = 281)


#Russia
TermFrequencyRussia <- data.frame(
  Term = colnames(IntegerMatrixRussia),
  Frequency = colSums(IntegerMatrixRussia),
  stringsAsFactors = FALSE
)
TermFrequencyRussia <- TermFrequencyRussia[order(TermFrequencyRussia$Frequency),]
wordcloud::wordcloud(
  words = TermFrequencyRussia$Term,
  freq = TermFrequencyRussia$Frequency,
  max.words = 25,
  random.order = FALSE,
  colors = viridis::viridis(100)
)
title('Word Cloud for tweets from president of Russia')
```

Word Cloud for tweets from president of Russia



Bar Plots

```r
library(ggplot2)
```

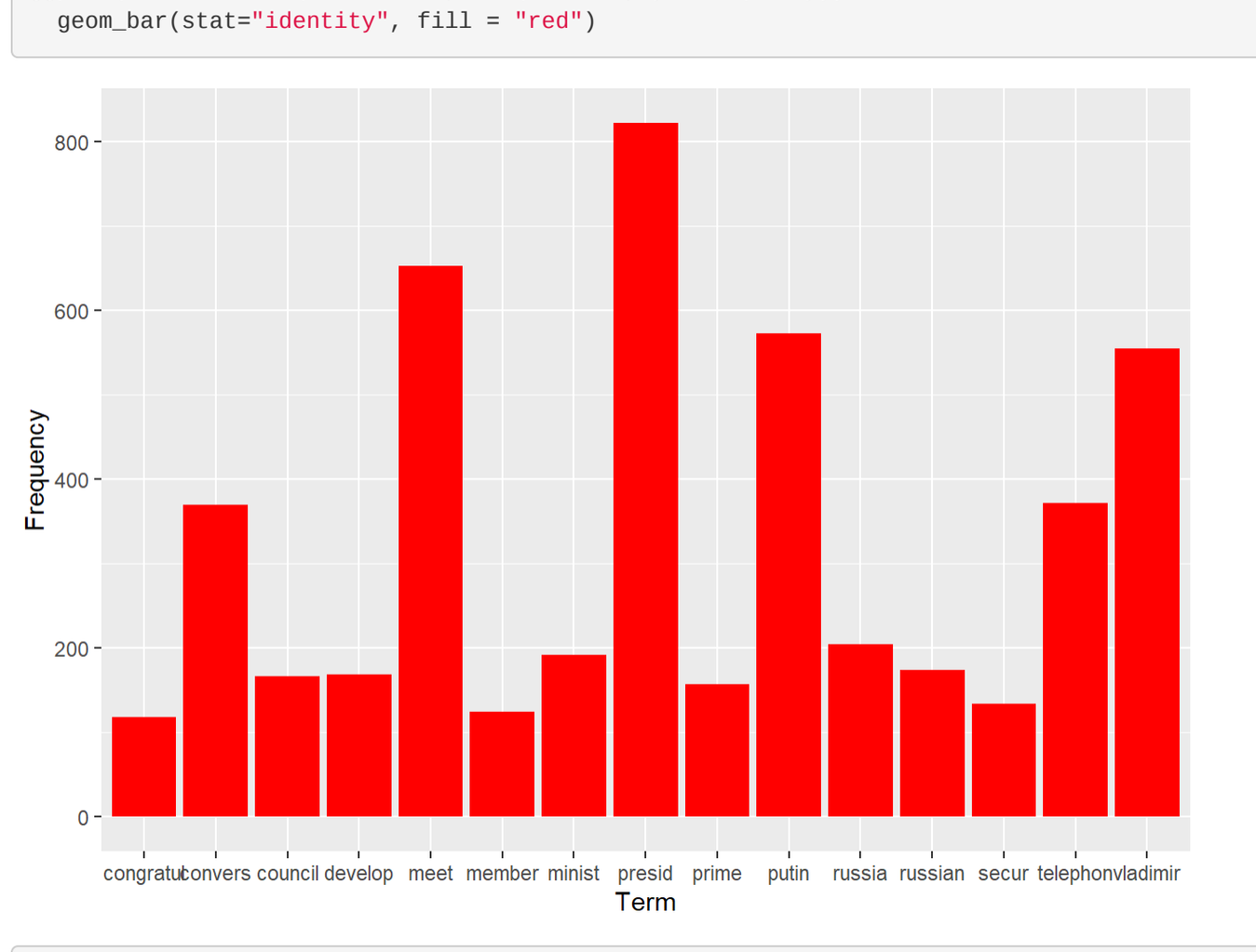
```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
##
## Attaching package: 'ggplot2'
```

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

```r
MostFrequentRussia = tail(TermFrequencyRussia, n = 15)
MostFrequentUkraine = tail(TermFrequencyUkraine, n = 15)

#Russia
ggplot(data=MostFrequentRussia, aes(x= Term, y= Frequency)) +
  geom_bar(stat="identity", fill = "red")
```




```r
#Ukraine
ggplot(data=MostFrequentUkraine, aes(x= Term, y= Frequency)) +
  geom_bar(stat="identity", fill = "blue")
```