Twitter-Sentimental-Analysis

Karan Dhir

27/11/2021

```
library(twitteR)
library(ROAuth)
library(httr)
library(base64enc)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:twitteR':
##
##     id, location

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(plyr)
```

```
## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:twitteR':
##
##     id
```

```
library(dismo)
```

```
## Loading required package: raster
```

```
## Loading required package: sp
```

```
##
## Attaching package: 'raster'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(wordcloud)
```

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

```
library(RColorBrewer)
library(lubridate)
```

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

```
## The following objects are masked from 'package:raster':
##
##     intersect, union
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(stringr)
library(plotly)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'plotly'
```

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

```
## The following object is masked from 'package:raster':
##
##     select
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, mutate, rename, summarise

## The following object is masked from 'package:httr':
##
##     config

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(ggplot2)
library(tidytext)
library(leaflet)
library(exploratory)
```

```
## Package attached: exploratory v0.3.13. Most recent version available on GitHub: v0.3.16
## You have an OPTION to update the package by typing 'update_exploratory()'. If you do so, make sure t
```

```
library(sentimentr)
```

```
##
## Attaching package: 'sentimentr'

## The following object is masked from 'package:plotly':
##
##     highlight
```

```
library(devtools)
```

```
## Loading required package: usethis
```

```
library(gofastr)
library(tm)
```

```
## Loading required package: NLP

##
## Attaching package: 'NLP'

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

## The following object is masked from 'package:httr':
##
##     content
```

```r
# Setting up the credentials in-order to call Twitter API.
#' @param search_string - hashtag string.
#' @param no_of_tweets - no of tweets to be returned from the API Call.
#' @return The list of tweets.
TwitterClient <- function(search_string, no_of_tweets) {
  api_key = as.character("mpJ47SSV27W875Coe1VRbI6hs")
  api_secret = as.character("1ukBRBBESV5WZqkAsAOvqeXeUJzKmOQVvmahgfglkeX5qCXluC")
  access_token = as.character("783402164169256960-cn8FYO2fhZZpxbf10dkdiLbJIcbvA7k")
  access_token_secret = as.character("KJzwcBGQoT9s0oOXsGRIxoMsxuYXzlS7cF5rleNHViLsa")

  setup_twitter_oauth(api_key, api_secret, access_token, access_token_secret)

  inital_date <- Sys.Date() - 60
  current_date <- Sys.Date()

  MTweets <-
    searchTwitter(
      search_string,
      n = no_of_tweets,
      since = as.character(inital_date),
      until = as.character(current_date)
    )
  df <- do.call("rbind", lapply(MTweets, as.data.frame))
  return(list(t_str = MTweets, t_df = df))
}

## Cleaning the tweets removing all the punctuation, digits, stop words, extra spaces and
## special characters.
#' @param tweet - textual part of the tweet.
#' @return The list of tweets.
clean_tweets = function(tweet)
{
  # convert to lower case
  tweet = tolower(tweet)
  # remove rt
  tweet = gsub("rt", "", tweet)
  # remove at
  tweet = gsub("@\\w+", "", tweet)
  # remove punctuation
  tweet = gsub("[[:punct:]]", "", tweet)
  # remove numbers
  tweet = gsub("[[:digit:]]", "", tweet)
  # remove links http
  tweet = gsub("http\\w+", "", tweet)
  # remove tabs
  tweet = gsub("[ |\t]{2,}", "", tweet)
  # remove blank spaces at the beginning
  tweet = gsub("^ ", "", tweet)
  # remove blank spaces at the end
  tweet = gsub(" $", "", tweet)
  # some other cleaning text
  tweet = gsub('https://', '', tweet)
  tweet = gsub('http://', '', tweet)
```

```
  tweet = gsub('[^[:graph:]]', ' ', tweet)
  tweet = gsub('[[:punct:]]', '', tweet)
  tweet = gsub('[[:cntrl:]]', '', tweet)
  tweet = gsub('\\d+', '', tweet)
  return(tweet)
}

# Combining the information extracted from the user and the tweets.
#' @param tweet - textual part of the tweet.
#' @return Return the combined data containing the information of user and tweets.
combine_tweet_and_userInfo <- function(tweet_obj) {
  user_info <- lookupUsers(tweet_obj$t_df$screenName)
  user_df <- twListToDF(user_info)

  tweet_df <- tweet_obj$t_df %>%
    full_join(user_df, by = c('screenName')) %>% dplyr::select(-c(longitude, latitude)) %>%
    dplyr::select(
      text,
      favorited,
      favoriteCount,
      created.x,
      statusSource,
      retweetCount,
      retweetCount,
      description,
      name,
      location
    )

  return (tweet_df)

}

# Visualization to show the number of tweets posted per hour.
#' @param tweet - Tweet Data frame.
#' @return - void
tweets_per_hour <- function(tweet_df) {
  tweet_df <- tweet_df %>%
    mutate(
      created = created.x %>%
        str_remove_all(pattern = '\\+0000') %>%
        parse_date_time(orders = '%y-%m-%d %H%M%S')
    )

  tweet_df <- tweet_df %>%
    mutate(Created_At_Round = created.x %>% round(units = 'hours') %>% as.POSIXct())

  plot_tph <- tweet_df %>%
    dplyr::count(Created_At_Round) %>%
    ggplot(mapping = aes(x = Created_At_Round, y = n)) +
    theme_light() +
    geom_line() +
    xlab(label = 'Date') +
```

```
    ylab(label = NULL) +
    ggtitle(label = 'Number of Tweets per Hour')

  plot_tph %>% ggplotly()
}

# Visualization of most frequent words corresponding to a hashtag.
#' @param tweet - Tweet Data frame.
#' @return - void
most_frequent_Cloud <- function(tweet_df) {
  tokens <- data_frame(text = tweet_df$text) %>%
    unnest_tokens(word, text) %>%
    anti_join(stop_words) %>%
    dplyr::count(word, sort = TRUE)

  hashtags <- data.frame(word = tokens$word, freq = tokens$n)
  hashtags <- hashtags[which(hashtags$freq > 30), ]
  wordcloud(
    words = hashtags$word,
    freq = hashtags$freq,
    min.freq = 1,
    max.words = 100,
    random.order = FALSE,
    rot.per = 0.35,
    colors = brewer.pal(8, "Dark2")
  )
}

# Visualize the sentiments of the tweets based on the positive and negative sentiments.
#' @param tweet - Tweet Data frame.
#' @return - void
visualize_sentiments <- function(tweet_sentiment) {
  sentiment_graph = plot_ly(
    x = tweet_sentiment$word_count,
    y = tweet_sentiment$ave_sentiment,
    mode = "markers",
    colors = c("red", "yellow"),
    size = abs(tweet_sentiment$ave_sentiment) / 3,
    color = ifelse(tweet_sentiment$ave_sentiment > 0, "Positive", "Negative")
  ) %>%
    layout(
      hovermode = "closest",
      title = "Sentiment analysis by Tweet",
      xaxis = list(title = "Number of words per Tweet", size = 18),
      yaxis = list(title = "Sentiments by Tweet", size = 18)
    )
  sentiment_graph
}

## Visualize the negative sentiment of the tweet.
#' @param tweet - Negative Data frame.
#' @return - void
visualize_negative_sentiment <- function(negative) {
```

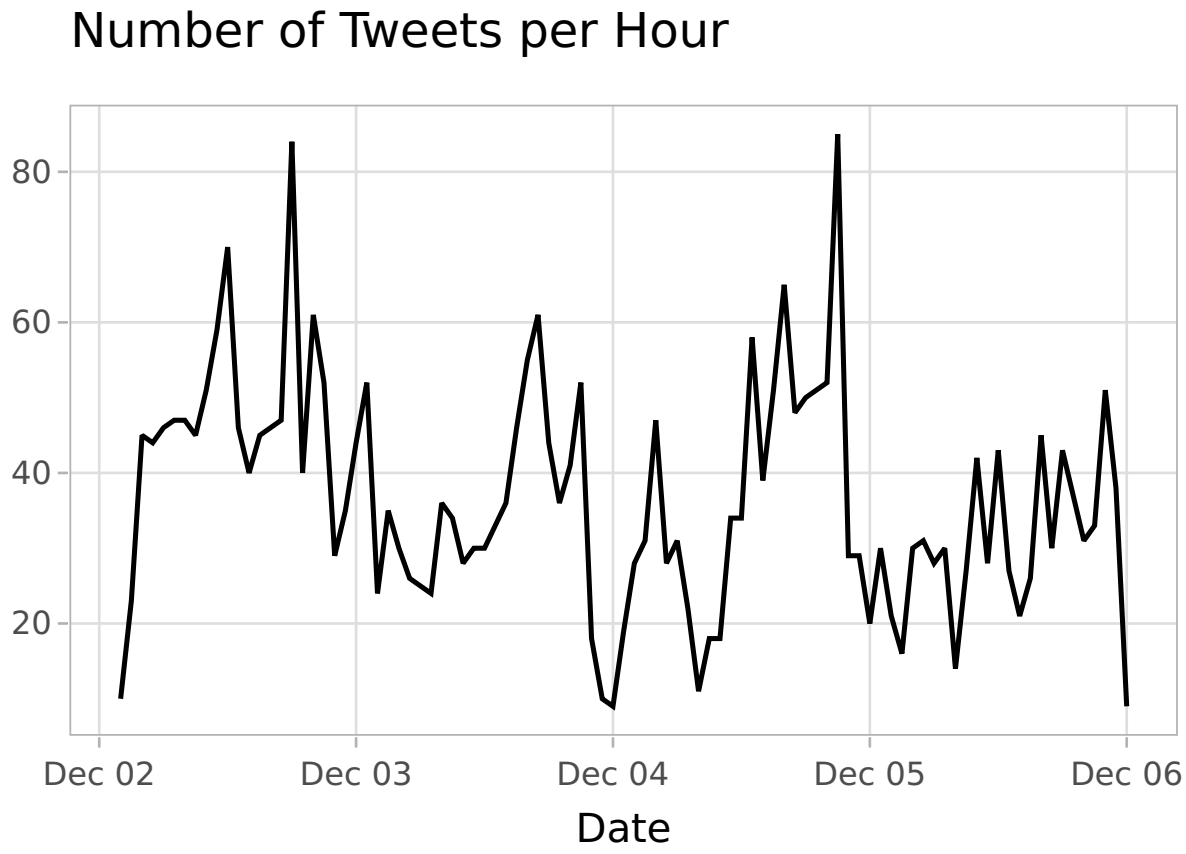
```
  negative_sentiment_graph = plot_ly(x = negative$word_count,
                                     y = negative$ave_sentiment,
                                     mode = "markers") %>%
    layout(
      hovermode = "closest",
      title = "Negatives",
      xaxis = list(title = "Number of words per Tweet", size = 18),
      yaxis = list(title = "Ave_Sentiments", size = 18)
    )
  negative_sentiment_graph
}

# visualize the positive sentiment of the tweet.
#' @param tweet - Positive Data frame.
#' @return - void
visualize_postitive_sentiment <- function(positive) {
  positive_sentiment_graph = plot_ly(x = positive$word_count,
                                     y = positive$ave_sentiment,
                                     mode = "markers") %>%
    layout(
      hovermode = "closest",
      title = "Positives",
      xaxis = list(title = "Number of words per Tweet", size = 18),
      yaxis = list(title = "Ave_Sentiments", size = 18)
    )
  positive_sentiment_graph
}

# Visualize those countries on map having highest frequency of tweets.
# Since Google API is paid so hard-coding the latitudes and longitudes of the countries.
#' @param tweet - Tweet Dataframe.
#' @return - void
show_countries_map <- function(tweet_df) {
  location_all  <- tweet_df %>%
    filter(is.na(location) == FALSE & location != "") %>%
    dplyr::count(location, sort = TRUE)

  location_filtered <-
    as.data.frame(sapply(location_all, grepl, pattern = ",", fixed = TRUE))
  location_new <-
    location_all[location_filtered$location == TRUE, ] %>% slice(1:10)

  lat <-
    c(
      0.347596,
      37.0902,
      35.3395,
      40.7128,
      37.9060,
      52.5200,
      45.4869,
      36.9741,
      37.0902,
```

```
      37.0902
    )
  long <-
    c(
      32.582520,
      95.7129,
      97.4867,
      74.0060,
      122.5450,
      13.4050,
      122.8040,
      122.0308,
      95.7129,
      95.7129
    )
  location_df <-
    data.frame(location_new$location, location_new$n, lat, long)

  getColor <- function(location_df) {
    sapply(location_df$n, function(n) {
      if (n <= 60) {
        "green"
      } else if (n <= 80) {
        "orange"
      } else {
        "red"
      }
    })
  }

  icons <- awesomeIcons(
    icon = 'ios-close',
    iconColor = 'black',
    library = 'ion',
    markerColor = getColor(location_df)
  )

  leaflet(location_df) %>% addTiles() %>%
    addAwesomeMarkers( ~ long,
                       ~ lat,
                       icon = icons,
                       label = (location_df$location_new.n))
}

## API call-out to get all the tweets belonging to a particular hash-tag.
searchString <- "#globalwarming"
no_of_tweets <- 3500
tweet_obj <- TwitterClient(searchString, no_of_tweets)
```

```
## [1] "Using direct authentication"
```

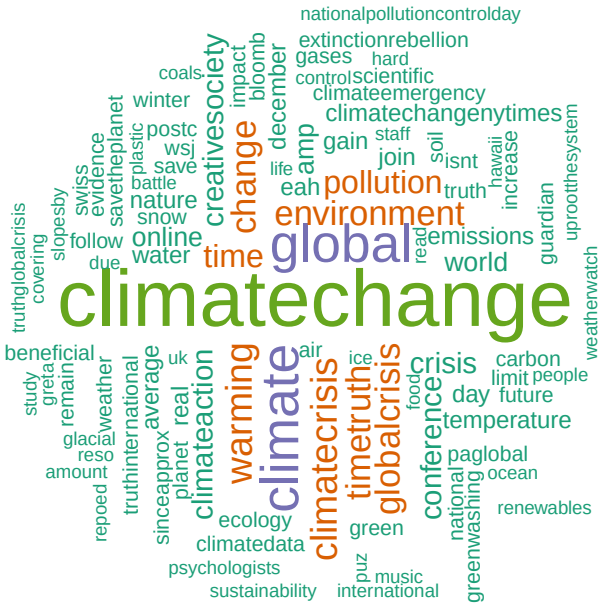
```
## Get the user information of the tweets.
tweet_df <- combine_tweet_and_userInfo(tweet_obj)
```

```
## Clean all the tweets by removing all the punctuation, stop words, links http, tabs, blank spaces.
filter_tweets <- clean_tweets(tweet_df$text)

tweet_df <- tweet_df %>%
  dplyr::select(-c(text)) %>%
  mutate(text = filter_tweets)

## Plot number of tweets per day.
tweets_per_hour(tweet_df)
```




```
## Plot tag-Cloud corresponding to words having highest frequency.
most_frequent_Cloud(tweet_df)
```

```
## Warning: ‘data_frame()‘ was deprecated in tibble 1.1.0.
## Please use ‘tibble()‘ instead.

## Joining, by = "word"

## Warning in wordcloud(words = hashtags$word, freq = hashtags$freq, min.freq =
## 1, : globalwarming could not be fit on page. It will not be plotted.
```




```
## Plot the sentiment of the tweet based on its polarity.
sentiments_df <- sentiment_attributes(tweet_df$text)
```

```
## Warning: 'dfm.character()' is deprecated. Use 'tokens()' first.
```

```
## Warning: '...' should not be used for tokens() arguments; use 'tokens()' first.
```

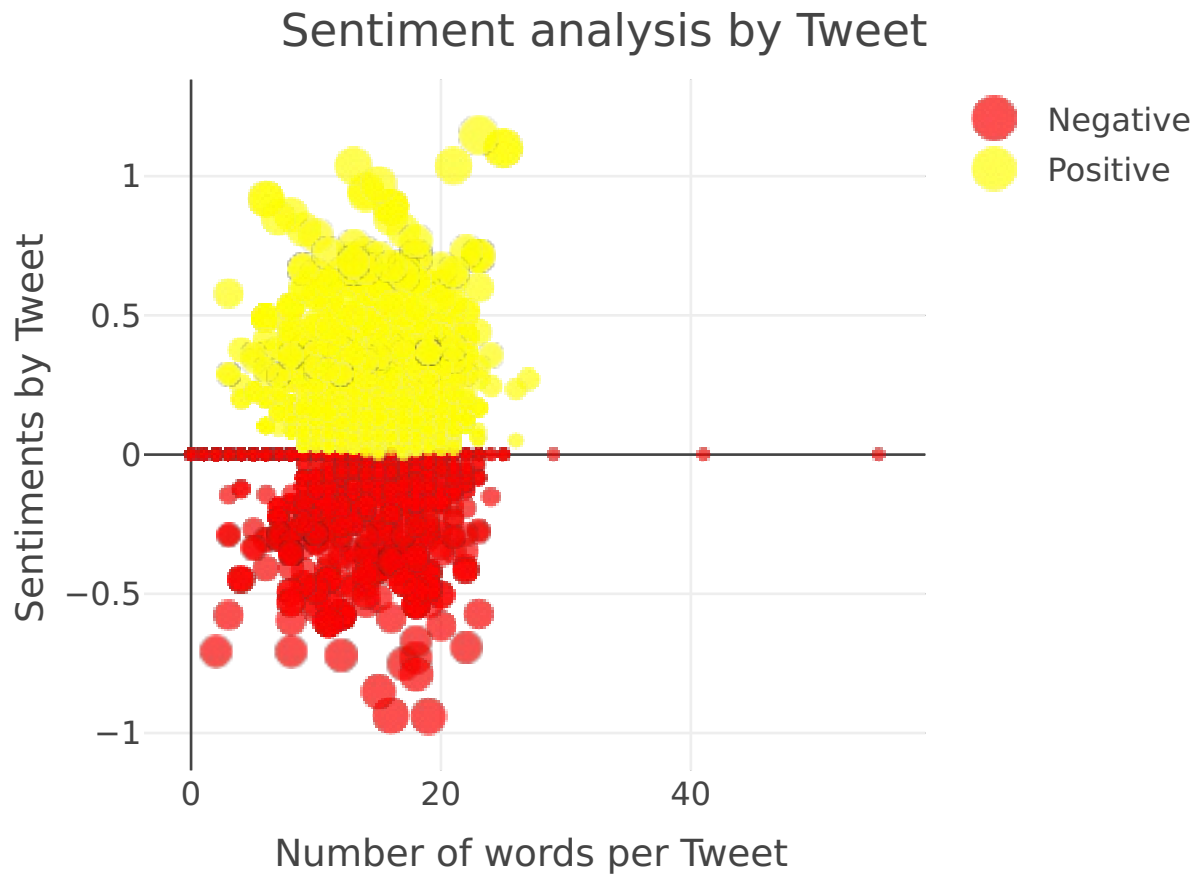
```
## Warning: 'stem' is deprecated; use dfm_wordstem() instead
```

```
new_2 <- get_sentences(tweet_df$text)
tweet_sentiment <-
  sentiment_by(new_2, averaging.function = average_weighted_mixed_sentiment)

visualize_sentiments(tweet_sentiment)
```

```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plotly.com/r/reference/#scatter
```

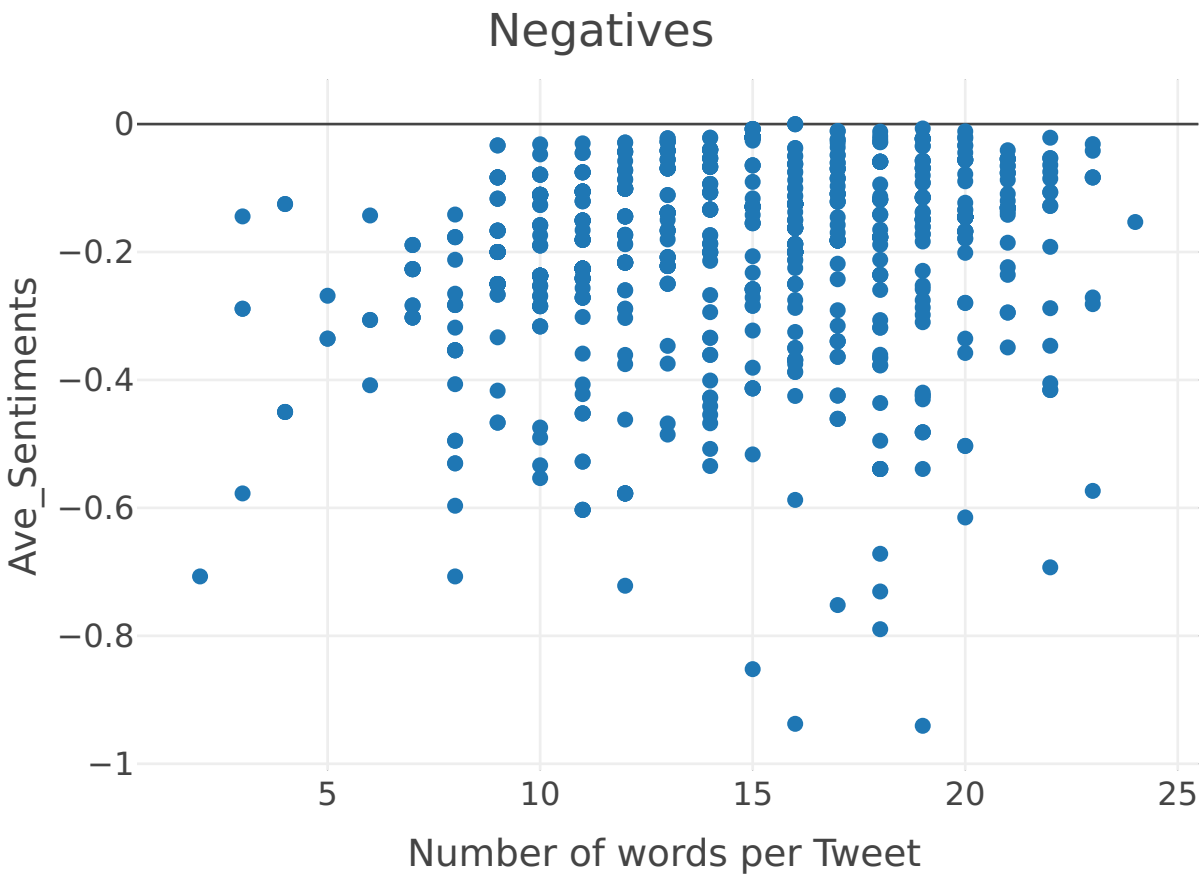
```
## Warning: `line.width` does not currently support multiple values.
```

```
## Warning: `line.width` does not currently support multiple values.
```



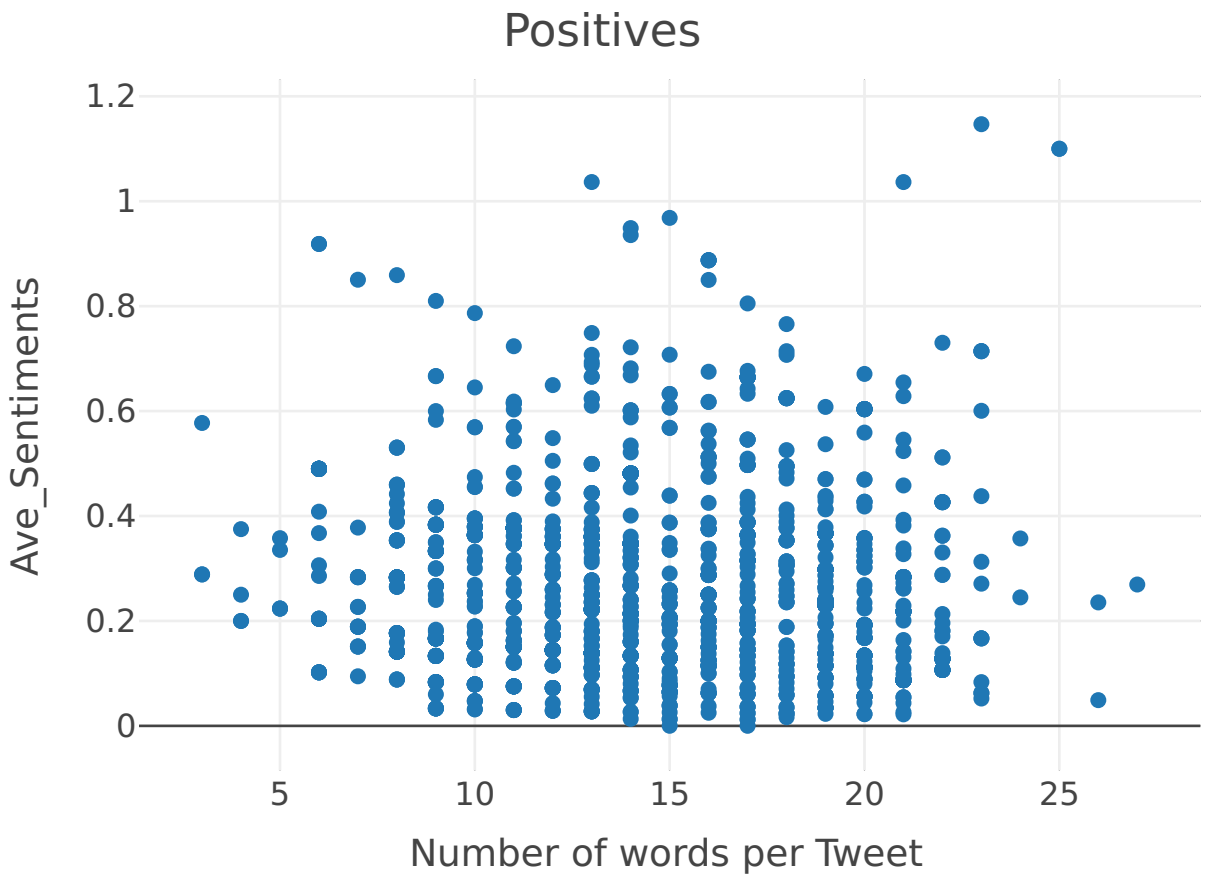

```
## Plot the sentiment of the tweet (Negative polarity).
negative_sentiment <- filter(tweet_sentiment, ave_sentiment < 0)
positive_sentiment <- filter(tweet_sentiment, ave_sentiment > 0)

visualize_negative_sentiment(negative_sentiment)
```

```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plotly.com/r/reference/#scatter
```

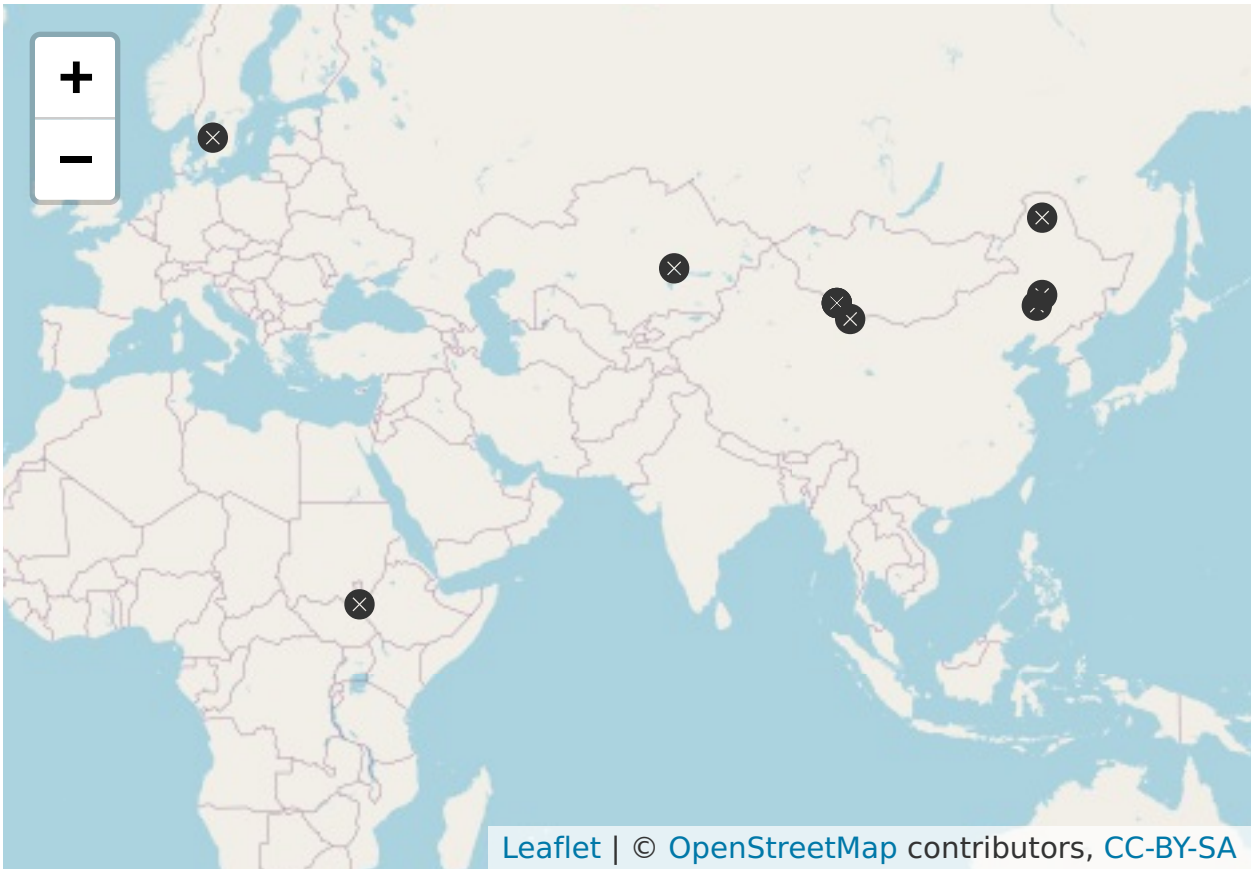



```
visualize_postitive_sentiment(positive_sentiment)
```

```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plotly.com/r/reference/#scatter
```




```
# Plotting the map based on the tweets collected from the different regions etc.
show_countries_map(tweet_df)
```


+
−
Leaflet | © OpenStreetMap contributors, CC-BY-SA