Twitter Sentiment Analysis

Install packages

```r
#Load libraries
library('quanteda')
```

```
## Package version: 3.2.0
## Unicode version: 13.0
## ICU version: 67.1

## Parallel computing: 8 of 8 threads used.

## See https://quanteda.io for tutorials and examples.
```

```r
library('readtext')
library('tidyverse')
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1

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

```r
library('quanteda.textstats')
library("rtweet")
```

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

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

```r
# #library("tidytext")
library("textstem")
```

```
## Loading required package: koRpus.lang.en

## Loading required package: koRpus

## Loading required package: sylly

## For information on available language packages for 'koRpus', run
##
##   available.koRpus.lang()
##
## and see ?install.koRpus.lang()

##
```

```
## Attaching package: 'koRpus'

## The following object is masked from 'package:readr':
##
##     tokenize

## The following objects are masked from 'package:quanteda':
##
##     tokens, types
```

```
library("httr")
library("jsonlite")
```

```
##
## Attaching package: 'jsonlite'

## The following object is masked from 'package:rtweet':
##
##     flatten

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

```
library("dplyr")
library("wordcloud")
```

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

Bringing in Tweets

Tweets have been saved as a csv and are imported here

```
# Getting twitter trends for Ireland
# ire_trends <- get_trends(woeid = "Ireland")
# head(ire_trends)
# ire_trends_7mar <- get_trends(woeid = "23424803")

# Getting twitter trends for the world
# wor_trends <- get_trends(woeid = "World", lang = "en")
# Bringing in 18000 tweets with the hashtag #StandWithUkraine, tweets in English (5th Mar)
# swu_18000 <- search_tweets("#StandWithUkraine", n = 18000, include_rts = FALSE, type = "mixed", lang
# readr::write_csv(swu_18000,"swu_18000.csv") # Saving the results to file
swu_18000 <- as.data.frame(readr::read_csv("/Users/garethmoen/Documents/Data Science/Portfolio/Sentimen
```

```
## Warning: One or more parsing issues, see `problems()` for details

## Rows: 16961 Columns: 90

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (27): screen_name, text, source, reply_to_screen_name, lang, quoted_tex...
## dbl  (19): user_id, status_id, display_text_width, reply_to_status_id, reply...
## lgl  (41): is_quote, is_retweet, quote_count, reply_count, hashtags, symbols...
## dttm  (3): created_at, quoted_created_at, account_created_at

##
## 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.
```

Full project used 400,000 tweets but is too big for Github storage

```
# Bringing in 400,000 tweets with the hashtag #StandWithUkraine, tweets in English (5th Mar)
# swu_400000 <- search_tweets("#StandWithUkraine", n = 400000, include_rts = FALSE, type = "mixed", ret
# swu_400000 <- subset(swu_400000, is.na(swu_400000$reply_to_status_id))
# readr::write_csv(swu_400000,"swu_400000.csv") # Saving the results to file
# swu_400000 <- readr::read_csv("swu_400000.csv") # Now the file is save, only read it in
```

Cleaning the tweets

```
# Tweets should be original tweets, not retweets, as the original API request filtered them out
# Selecting columns
df <- swu_18000 %>%
  select(status_id,
         created_at,
         text,
         source,
         screen_name, # used to remove irrelevant tweets later
         favorite_count,
         retweet_count
  ) # %>%
  #as.data.frame()

# Remove large swu_18000 tibble
#rm(swu_18000)

# Removal of irrelevant sources
df <- df[-which(df$screen_name == "ArvadaRadio"),] # radio station
df <- df[-which(df$screen_name == "EstellaBell1"),] # other news
df <- df[-which(df$screen_name == "AlenaKazakevich"),] # repeated tweets
df <- df[-which(df$screen_name == "Fidget02"),] # only hashtags

# Remove duplicated status IDs if needed
# df <- df[-which(duplicated(df$status_id)),]

# Filter for popular tweets with 1 or more in the 'favourite_count'
df <- df[df$favorite_count >= 1,]

# Randomise the order of the tweets
set.seed(1234)
rows <- sample(nrow(df))
df_shuf  <- df[rows, ]

# Looking at tweets related to Putin
#df_putin <- dplyr::filter(df_shuf, grepl('\\bputin\\b', text) | grepl('\\bPutin\\b', text))

# Looking at tweets related to 'ukraine'  or'ukrainians'
df_ukr <- dplyr::filter(df_shuf, grepl('\\ukraine\\b', text) | grepl('\\bukrainians\\b', text)| grepl('

# Conversion to corpus
corp <- corpus(df_ukr,
               docid_field = "status_id",
               text_field = "text",
               unique_docnames = TRUE)

# summary(corp)

prep_toks <- function(text_corpus){
  toks <- quanteda::tokens(text_corpus,
                 include_docvars = TRUE) %>%
    tokens_tolower() %>%
    tokens_remove(stopwords("english"), padding = TRUE) %>%
    tokens_remove('[\\p{P}\\p{S}]', valuetype = 'regex', padding = TRUE)
  return(toks)
}

toks <- corp %>%
  prep_toks()

# head(toks)
```

Creating collocations

```
get_coll <- function(tokens){
  unsup_col <- textstat_collocations(tokens,
                                     method = "lambda",
                                     size = 2,
                                     min_count = 5,
                                     smoothing = 0.5)

  unsup_col <- unsup_col[order(-unsup_col$count),] # sort detected collocations by count (descending)
  return(unsup_col)
}
collocations <- get_coll(toks) # create collocations

toks <- tokens_compound(toks, pattern = collocations[collocations$z > 5]) # merge collocations into tok

toks <- tokens_remove(toks, c("amp", "come", "months", "weeks", "analysts_said", "can", "today", "now",

toks <- quanteda::tokens(toks,
               remove_numbers = TRUE,
               remove_punct = TRUE,
               remove_symbols = TRUE,
               remove_hyphens = TRUE,
               remove_separators = TRUE,
               remove_url = TRUE) # remove other uninformative text
```

```
## Warning: remove_hyphens argument is not used.
```

Creating collocations and returning them back to the main corpus may not be useful as some of the words with sentiment may be lost. So for example, ‘aggressor’ would be considered a negative sentiment, but ‘russian aggressor’ may not be considered such. So it’s best to leave the words as they are for the purpose of sentiment analysis.

```
# Creating a dfm
dfm <- dfm(toks) # create DFM
dfm <- dfm_trim(dfm, min_docfreq = 20) # trim DFM
```

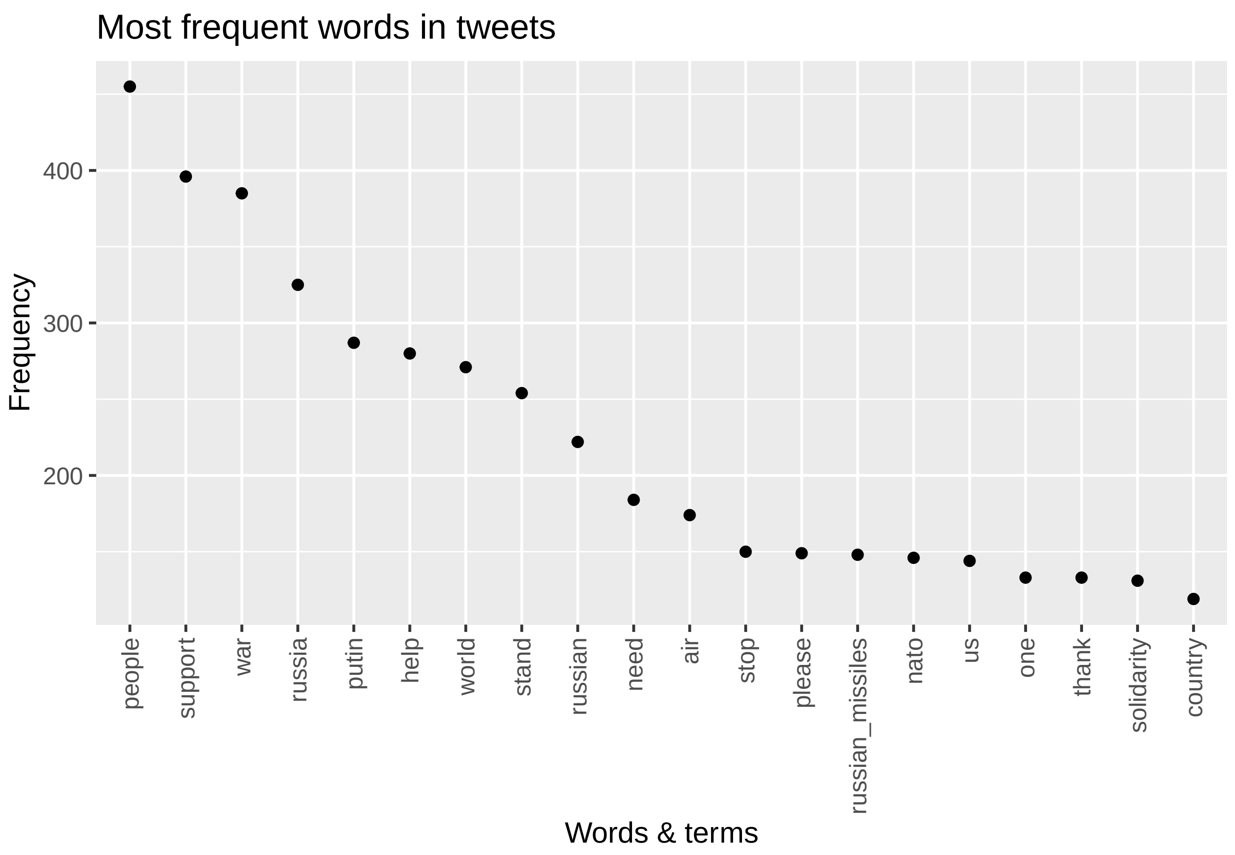
```r
#dfm <- dfm_tfidf(dfm) # weight DFM

# Top 20 most frequent words / expressions
dfm_plot <- textstat_frequency(dfm, n = 20)
 # use textstat_frequency(dfm, n = 30, force = TRUE) if the DFM is already weighted
#dfm_plot <- dfm_plot[-1,]
dfm_plot
```

```
##              feature frequency rank docfreq group
## 1             people       455    1     411   all
## 2            support       396    2     370   all
## 3                war       385    3     373   all
## 4             russia       325    4     308   all
## 5              putin       287    5     274   all
## 6               help       280    6     265   all
## 7              world       271    7     247   all
## 8              stand       254    8     246   all
## 9            russian       222    9     215   all
## 10              need       184   10     179   all
## 11               air       174   11     137   all
## 12              stop       150   12     138   all
## 13            please       149   13     142   all
## 14 russian_missiles       148   14     127   all
## 15              nato       146   15     136   all
## 16                us       144   16     137   all
## 17             thank       133   17     117   all
## 18               one       133   17     125   all
## 19        solidarity       131   19     131   all
## 20           country       119   20     117   all
```

Plot the most frequent words & expressions

```r
dfm_plot %>%
  ggplot(aes(x = reorder(feature, -frequency), y = frequency)) +
  geom_point() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
    labs(title = "Most frequent words in tweets",
       x = "Words & terms",
       y = "Frequency")
```



### Adding sentiment to the data

```
# General sentiment
dfm_sentiment <- dfm_lookup(dfm(toks), data_dictionary_LSD2015) %>%
    dfm_group(groups = dfm@docvars$created_at)
# dfm_sentiment <- dfm_lookup(dfm, data_dictionary_LSD2015) %>%
#   dfm_group(groups = date)
```

Plot of sentiment over time

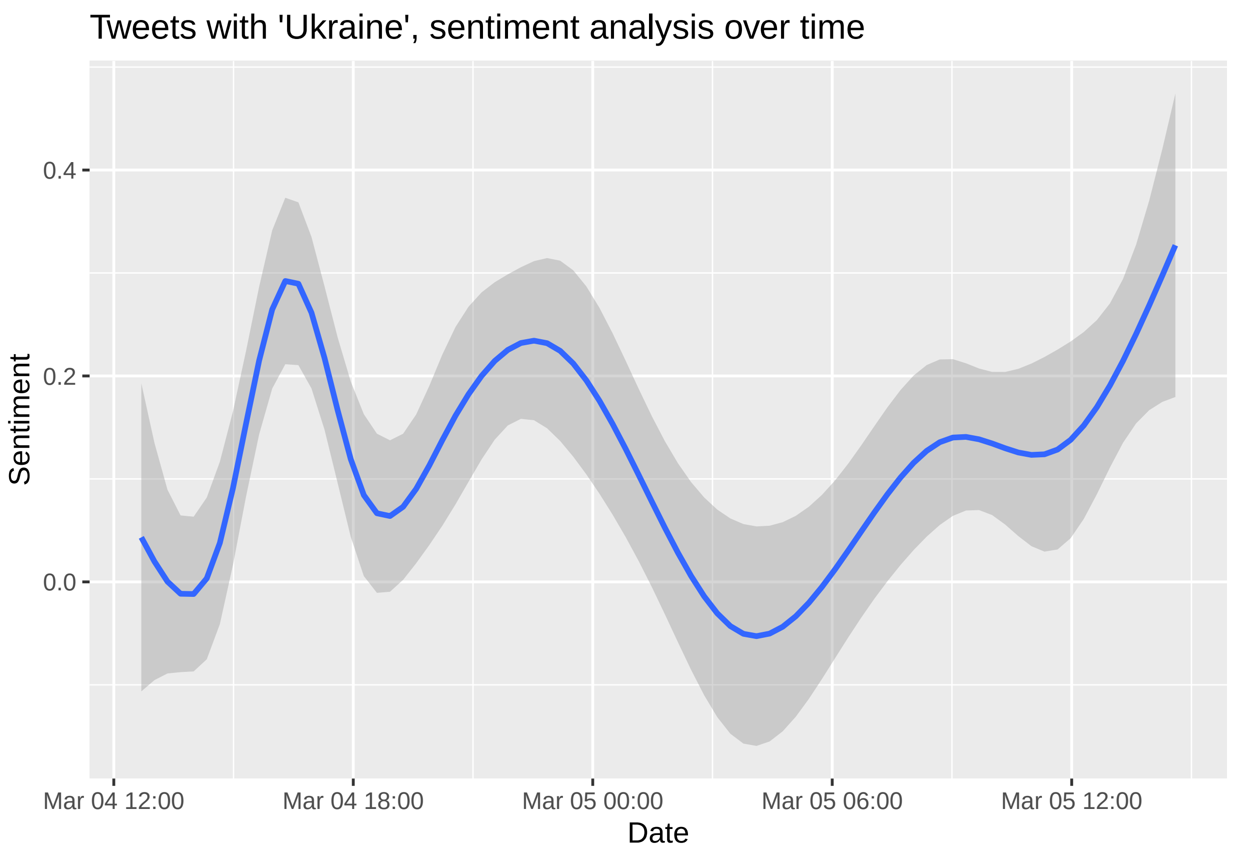
```
# Only keep column 1 - 2
dfm_sentiment <- dfm_lookup(dfm(toks), data_dictionary_LSD2015[1:2])
# Next prepare to plot
docvars(dfm_sentiment, "prop_negative") <- as.numeric(dfm_sentiment[,1] / ntoken(dfm_sentiment))
docvars(dfm_sentiment, "prop_positive") <- as.numeric(dfm_sentiment[,2] / ntoken(dfm_sentiment))

docvars(dfm_sentiment, "net_sentiment") <- docvars(dfm_sentiment, "prop_positive") - docvars(dfm_sentim

docvars(dfm_sentiment) %>%
    ggplot(aes(x = dfm@docvars$created_at, y = net_sentiment)) +
    geom_smooth() +
    labs(title = "Tweets with 'Ukraine', sentiment analysis over time",
        x = "Date",
        y = "Sentiment")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

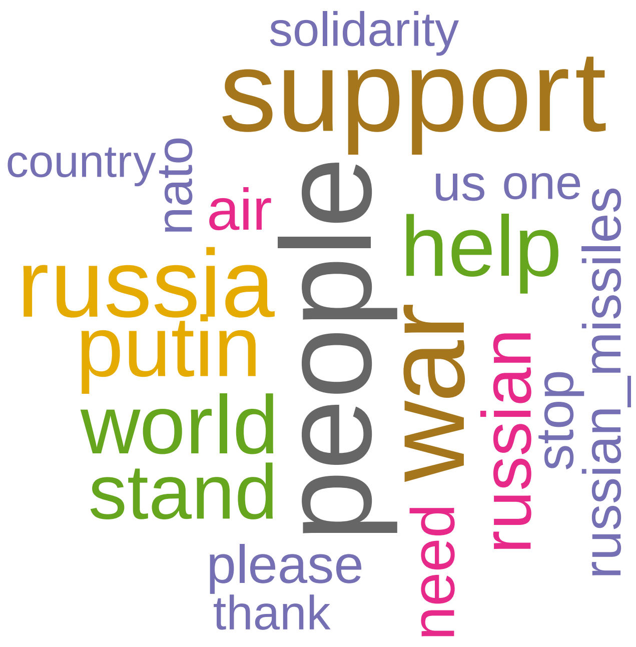
```
## Warning: Removed 668 rows containing non-finite values (stat_smooth).
```



Wordcloud of most frequent words

```
dfwc <- as.data.frame(dfm_plot)

wordcloud(words = dfwc$feature, freq = dfwc$frequency, min.freq = 1, max.words=150, random.order=FALSE,
```


solidarity
support
country
nato
air
us one
help
russia
people
putin
war
russian
stop
russian_missiles
world
stand
please
thank
need