

# Lab 7: Sentiment analysis

## Clinton-Trump Election

Isabel Casas  
[icasas@deusto.es](mailto:icasas@deusto.es)

### Learning objectives

1. Learn text mining with R
2. Cleaning text data
3. Visualising text data
4. Sentiment analysis

### Tweets in the Clinton-Trump election

We are going to see some text mining techniques. This is part of a popular discipline called natural language processing (NLP) which uses models, commonly deep neural networks, to teach machines understand texts.

- Download *tweet.csv* from [Kaggle Notebook](#)
- This dataset contains 3000 tweets posted by Hillary Clinton and Donald Trump between January 2016 and the end of September 2016, just before the elections.
- The data was collected by ([Hammer 2017](#))
- Reproduce the analysis in the R notebook by ([Bruin 2018](#))

We unload the packages we are going to use in this document.

```
packages = c("knitr", "tidyr", "dplyr", "readr", "ggplot2", "tibble", "stringr", "gridExtra",
, "scales", "lubridate", "ggrepel", "reshape2", "kableExtra", "tm", "wordcloud",
"tidytext", "broom", "topicmodels", "data.table")
## Now load or install&load all
package.check <- lapply(packages, FUN = function(x){
  if (!require(x, character.only = TRUE)) {
    install.packages(x, dependencies = TRUE)
    library(x, character.only = TRUE)
  }
  else
    library(x, character.only = TRUE)
})
```

### Data

We have been using functions *read.table()* and *read.csv()* to upload files into our memory. The result is a *data.frame*. Handling big datasets have motivated the creating of new objects that handle it more efficiently,



for example *data.table* and *tibble*. Part of this is package *data.table* which contains function *fread()* that uploads large datasets into the R memory faster than *read.table*. This function

The chunk below, uploads the data into memory. Encoding UTF-8 is the classical and very simple system to store text into a machine. Remember machines only have 0 and 1 in their memory. We can see that tweets have 28 variables and 6434 messages.

```
tweets <- data.table::fread("tweets.csv", encoding = "UTF-8")
names(tweets)
```

```
## [1] "id"           "handle"
## [3] "text"         "is_retweet"
## [5] "original_author" "time"
## [7] "in_reply_to_screen_name" "in_reply_to_status_id"
## [9] "in_reply_to_user_id"    "is_quote_status"
## [11] "lang"              "retweet_count"
## [13] "favorite_count"     "longitude"
## [15] "latitude"          "place_id"
## [17] "place_full_name"    "place_name"
## [19] "place_type"         "place_country_code"
## [21] "place_country"      "place_contained_within"
## [23] "place_attributes"   "place_bounding_box"
## [25] "source_url"         "truncated"
## [27] "entities"           "extended_entities"
```

```
dim(tweets)
```

```
## [1] 6434 28
```

```
class(tweets)
```

```
## [1] "data.table" "data.frame"
```

**Question 1:** How can see you the different variable types inside *tweets*? In particular, we are interested invariable *time*.

**Questions 2** What does function *ymd\_hms()* do?

```
head(tweets$time)
?ymd_hms
```

## Exploratory Data Analysis

**Do Clinton or Trump also tweet in Languages other than English?**

Variable *lang* inside *tweet* has the answer.

```
names(tweets)
```

```
## [1] "id"           "handle"
## [3] "text"         "is_retweet"
## [5] "original_author" "time"
## [7] "in_reply_to_screen_name" "in_reply_to_status_id"
## [9] "in_reply_to_user_id"    "is_quote_status"
## [11] "lang"              "retweet_count"
## [13] "favorite_count"     "longitude"
## [15] "latitude"          "place_id"
## [17] "place_full_name"    "place_name"
## [19] "place_type"         "place_country_code"
## [21] "place_country"      "place_contained_within"
```



```
## [23] "place_attributes"      "place_bounding_box"
## [25] "source_url"            "truncated"
## [27] "entities"              "extended_entities"
```

```
tweets %>% group_by(lang)
```

```
## # A tibble: 6,434 x 28
## # Groups:   lang [8]
##       id handle      text is_re~1 origi~2 time                in_re~3 in_re~4
##   <int64> <chr>      <chr> <lgl>  <chr>   <dtm>                <chr>    <dbl>
## 1  7e17 HillaryCli~ "The~ FALSE   ""      2016-09-28 00:22:34 ""      NA
## 2  7e17 HillaryCli~ "Las~ TRUE    "timka~ 2016-09-27 23:45:00 ""      NA
## 3  7e17 HillaryCli~ "Cou~ TRUE    "POTUS" 2016-09-27 23:26:40 ""      NA
## 4  7e17 HillaryCli~ "If ~ FALSE   ""      2016-09-27 23:08:41 ""      NA
## 5  7e17 HillaryCli~ "Bot~ FALSE   ""      2016-09-27 22:30:27 ""      NA
## 6  7e17realDonaldTrump~ "Joi~ FALSE   ""      2016-09-27 22:13:24 ""      NA
## 7  7e17 HillaryCli~ "Thi~ FALSE   ""      2016-09-27 21:35:28 ""      NA
## 8  7e17 HillaryCli~ "Whe~ FALSE   ""      2016-09-27 21:25:31 ""      NA
## 9  7e17realDonaldTrump~ "Onc~ FALSE   ""      2016-09-27 21:08:22 ""      NA
## 10 7e17 HillaryCli~ "3) ~ TRUE    "mcuba~ 2016-09-27 21:00:13 ""      NA
## # ... with 6,424 more rows, 20 more variables: in_reply_to_user_id <dbl>,
## #   is_quote_status <lgl>, lang <chr>, retweet_count <int>,
## #   favorite_count <int>, longitude <dbl>, latitude <dbl>, place_id <chr>,
## #   place_full_name <chr>, place_name <chr>, place_type <chr>,
## #   place_country_code <chr>, place_country <chr>,
## #   place_contained_within <chr>, place_attributes <chr>,
## #   place_bounding_box <chr>, source_url <chr>, truncated <lgl>, ...
```

```
tweets %>% group_by(lang) %>% count()
```

```
## # A tibble: 8 x 2
## # Groups:   lang [8]
##   lang      n
##   <chr> <int>
## 1 da         3
## 2 en      6238
## 3 es       105
## 4 et         1
## 5 fi         1
## 6 fr         2
## 7 tl         2
## 8 und       82
```

```
tweets %>% group_by(lang) %>% count() %>% rename(Language = lang, 'Number of Tweets' = n)
```

```
## # A tibble: 8 x 2
## # Groups:   Language [8]
##   Language `Number of Tweets`
##   <chr>          <int>
## 1 da              3
## 2 en            6238
## 3 es            105
## 4 et              1
## 5 fi              1
## 6 fr              2
## 7 tl              2
```



## 8 und

82

```
kable(tweets %>% group_by(lang) %>% count() %>%  
      rename(Language = lang, 'Number of Tweets' = n))
```

Language	Number of Tweets
da	3
en	6238
es	105
et	1
fi	1
fr	2
tl	2
und	82

**Question 3** What does function *kable* do? What package does it belong to?

```
#Choose those tweets that are not in "es"  
tweets <- tweets %>% filter(lang != "es")  
names(tweets)
```

```
## [1] "id" "handle"  
## [3] "text" "is_retweet"  
## [5] "original_author" "time"  
## [7] "in_reply_to_screen_name" "in_reply_to_status_id"  
## [9] "in_reply_to_user_id" "is_quote_status"  
## [11] "lang" "retweet_count"  
## [13] "favorite_count" "longitude"  
## [15] "latitude" "place_id"  
## [17] "place_full_name" "place_name"  
## [19] "place_type" "place_country_code"  
## [21] "place_country" "place_contained_within"  
## [23] "place_attributes" "place_bounding_box"  
## [25] "source_url" "truncated"  
## [27] "entities" "extended_entities"
```

```
head(tweets$is_retweet)
```

```
## [1] FALSE TRUE TRUE FALSE FALSE FALSE
```

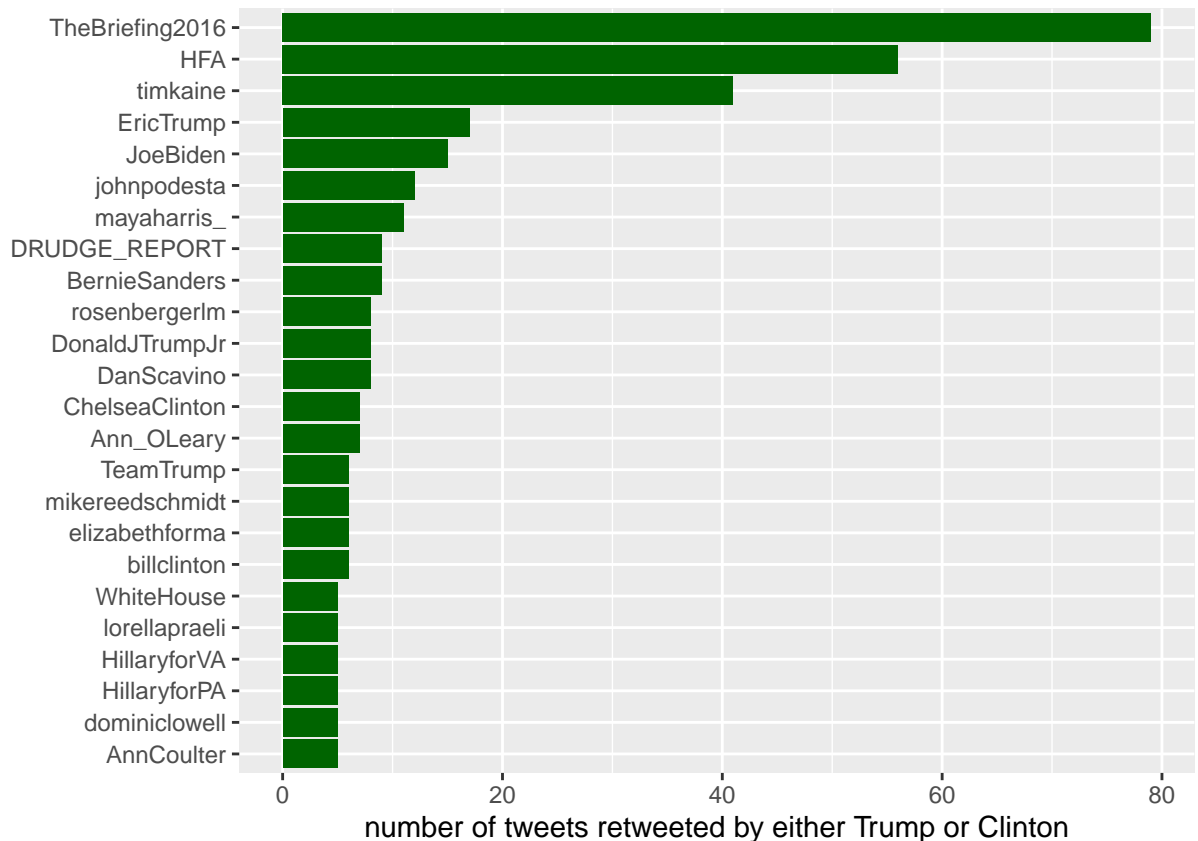
```
tweets$handle <- sub("realDonaldTrump", "Trump", tweets$handle)  
tweets$handle <- sub("HillaryClinton", "Clinton", tweets$handle)  
tweets$is_retweet <- as.logical(tweets$is_retweet)  
kable(tweets %>% filter(is_retweet==FALSE) %>% group_by(handle) %>% count())
```

handle	n
Clinton	2557
Trump	3081

Now, let's see if which people were interesting enough for Donald or Hillary to retweet. I have only displayed people that were retweeted at least 5 times, as in total 274 people were retweeted.

```
p1 <- tweets %>% filter(original_author != "") %>% group_by(original_author) %>%  
count() %>% filter(n>=5) %>% arrange(desc(n)) %>% ungroup()  
ggplot(p1, aes(x=reorder(original_author, n), y=n)) +  
geom_bar(stat="identity", fill="darkgreen") + coord_flip() +  
labs(x="", y="number of tweets retweeted by either Trump or Clinton") +  
theme(legend.position = "none")
```





## Text Mining

### Tweets text

**Question 6** What does function *ifelse* do? Check the code below to understand it.

```
set.seed(42)
x <- rnorm(10)
y <- ifelse(x>=0, "positive", "negative")
x
```

```
## [1] 1.37095845 -0.56469817 0.36312841 0.63286260 0.40426832 -0.10612452
```

```
## [7] 1.51152200 -0.09465904 2.01842371 -0.06271410
```

```
y
```

```
## [1] "positive" "negative" "positive" "positive" "positive" "negative"
```

```
## [7] "positive" "negative" "positive" "negative"
```

We realized that *text* variable contains some regular expressions (Regex) and text that we are not interested in. **Question 7** What do we mean with *Regex*?

```
tweets$text[c(2,4)]
```

```
## [1] "Last night, Donald Trump said not paying taxes was \"smart.\" You know what I call it? Unpa
```

```
## [2] "If we stand together, there's nothing we can't do. \n\nMake sure you're ready to vote: https://"
```

**Question 8** How is he cleaning the data to avoid the *Regex*? What is he doing in the code below?



```
library(stringr)
tweets$text <- str_replace_all(tweets$text, "[\n]" , "") #remove new lines
tweets$text <- str_replace_all(tweets$text, "&", "") # rm ampersand
#URLs are always at the end and did not counts towards the 140 characters limit
tweets$text <- str_replace_all(tweets$text, "http.*" , "")
tweets$text <- iconv(tweets$text, "latin1", "ASCII", sub="")
```

## Creating a VCorpus object

In general, corpus is a collection of texts that often are complete and self-contained. In natural language processing (NLP), a corpus is a text document that is analysed for example in sentiment analysis. So the first step in sentiment analysis is to create a corpus. In R, this can be done with:

```
Corpus name = Vcorpus(vectorsource(data frame$column name))
```

The chunk below, creates a VCorpus with all tweets of Trump and another for all tweets of Clinton that have not been retweeted.

```
library(tm)
ClintonTweets <- tweets %>% filter(is_retweet=="FALSE" & handle=="Clinton")
TrumpTweets <- tweets %>% filter(is_retweet=="FALSE" & handle=="Trump")
names(TrumpTweets)[1]="doc_id"
TrumpCorpus <- DataframeSource(TrumpTweets)
TrumpCorpus <- VCorpus(TrumpCorpus)
names(ClintonTweets)[1]="doc_id"
ClintonCorpus <- DataframeSource(ClintonTweets)
ClintonCorpus <- VCorpus(ClintonCorpus)
TrumpCorpus
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 26
## Content:  documents: 3081
```

```
inspect(TrumpCorpus[1:2])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 26
## Content:  documents: 2
##
## [[1]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 95
##
## [[2]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 90
```

See the content of the tweet you want using function *content()* as in

```
content(TrumpCorpus[[1]])
```

```
## [1] "Join me for a 3pm rally - tomorrow at the Mid-America Center in Council Bluffs, Iowa! Tickets: "
```



```
content(ClintonCorpus[[5]])
```

```
## [1] "When Donald Trump goes low...register to vote: "
```

## Data reduction

We create a function called *CleanCorpus* that when called will do the following to the corpus:

- Convert the text to all lower case
- Remove numbers
- Remove stopwords
- Remove punctuation
- Remove white spaces

```
CleanCorpus <- function(x){  
x <- tm_map(x, content_transformer(tolower))  
#remove numbers before removing words. Otherwise "trump2016" leaves "trump"  
x <- tm_map(x, removeNumbers)  
#this does not work in Rstudio Cloud, we need more memory. It should work in your computer  
x <- tm_map(x, removeWords, tidytext::stop_words$word)  
x <- tm_map(x, removePunctuation)  
x <- tm_map(x, stripWhitespace)  
return(x)  
}
```

The function *CreateTermsMatrix()* which first creates a term-document matrix. Remember that a term-document matrix has as many columns as documents in a corpus, in our case, as many documents as tweets and as many rows as different words. Then it counts how many times each word appears, counting the number of times for each document. This is done instead of using the *row\_binds*. Finally, it sorts the words by frequency.

```
CreateTermsMatrix <- function(x) {  
x <- TermDocumentMatrix(x)  
x <- as.matrix(x)  
y <- rowSums(x)  
y <- sort(y, decreasing=TRUE)  
return(y)  
}
```

Let us apply those functions to the *TrumpCorpus*. First reduce the data and later count how many times each word appear for all the tweets in the *TrumpCorpus*. The result is in variable *TermFreqTrump*.

```
TrumpCorpus <- CleanCorpus(TrumpCorpus)  
#See the difference with the first tweet before the reduction  
content(TrumpCorpus[[1]])
```

```
## [1] "join pm rally tomorrow midamerica center council bluffs iowa tickets "
```

```
# It counts the left words
```

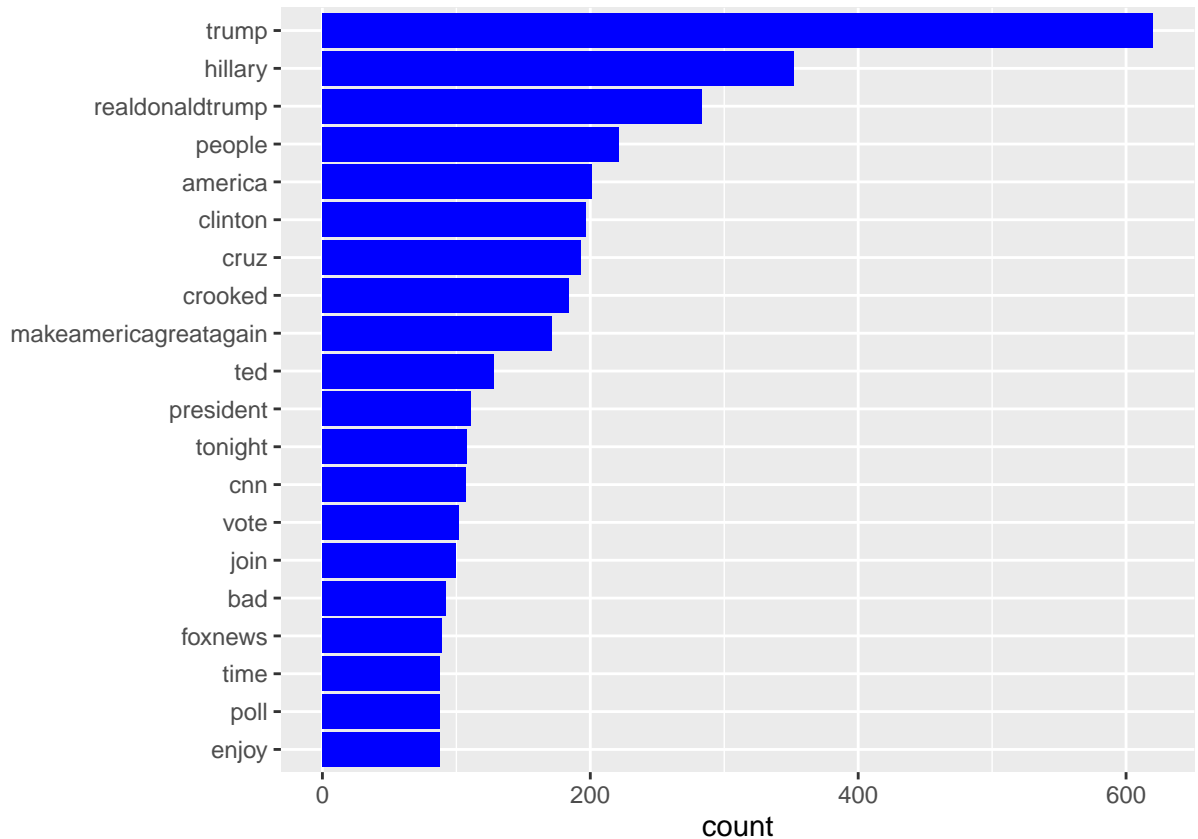
```
TermFreqTrump <- CreateTermsMatrix(TrumpCorpus)
```

## Words that Trump uses the most

We have done the cleaning above because usually the words we used most are a, 'the', etc and now *TrumpCorpus* has only informative words that we counted. Let us plot the top most frequent words in a bar plot.



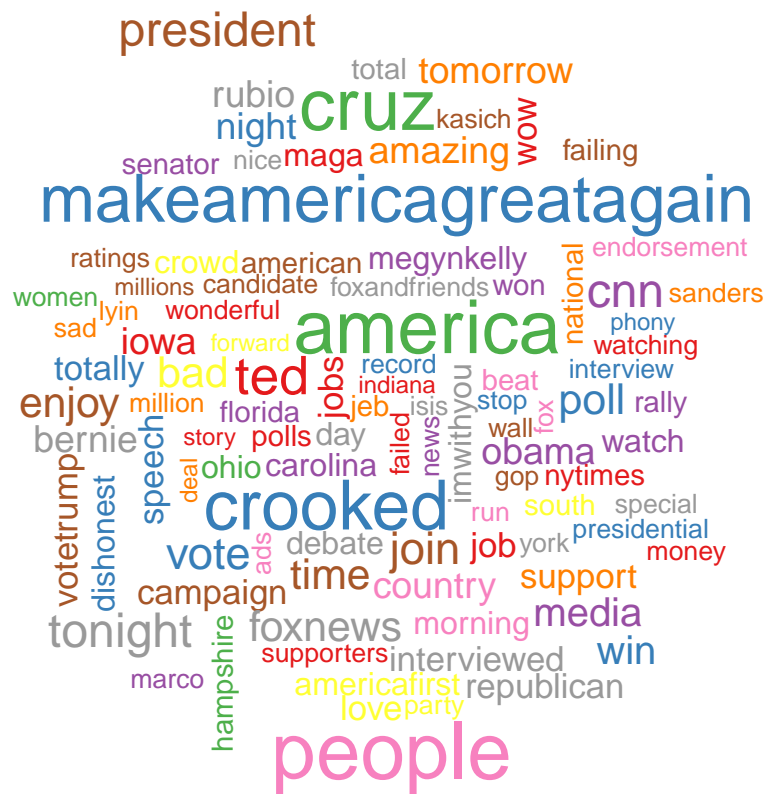
```
library(ggplot2)
TrumpDF <- data.frame(word=names(TermFreqTrump), count=TermFreqTrump)
TrumpDF[1:20,] %>%
  ggplot(aes(x=reorder(word, count), y=count)) +
  geom_bar(stat='identity', fill="blue") + coord_flip() + theme(legend.position = "none") +
  labs(x="")
```



See these results in a Cloudword format but removing the names: hillary, trump, clinton, realdonaldtrump and hillaryclinton.

```
library(wordcloud)
TrumpCorpus1 <- tm_map(TrumpCorpus, removeWords, c("donald", "hillary", "clinton", "trump",
"realdonaldtrump", "hillaryclinton"))
TermFreqTrump <- CreateTermsMatrix(TrumpCorpus1)
TrumpDF <- data.frame(word=names(TermFreqTrump), count=TermFreqTrump)
wordcloud(TrumpDF$word, TrumpDF$count, max.words = 100, scale=c(2.5,.5), random.color = TRUE,
colors=brewer.pal(9,"Set1"))
```





**Question 9** Plot the Wordcloud of Clinton Top20 most used words apart from names and Regex

## Sentiment analysis

The simplest way to score the sentiment of a text is to count the positively and negatively charged terms in the document. Researchers have proposed numerous collections of terms. For example the dictionary provided by [Hu and Liu \(2004\)](#). It consists of 2 lists of several thousand terms that reveal the sentiment orientation of a text. We can get this list in R from the *sentiments* dataframe in package *tidytext*.

The *bing* lexicon (Hu and Liu, 2004)

```
library(tidytext)
get_sentiments("bing")
```

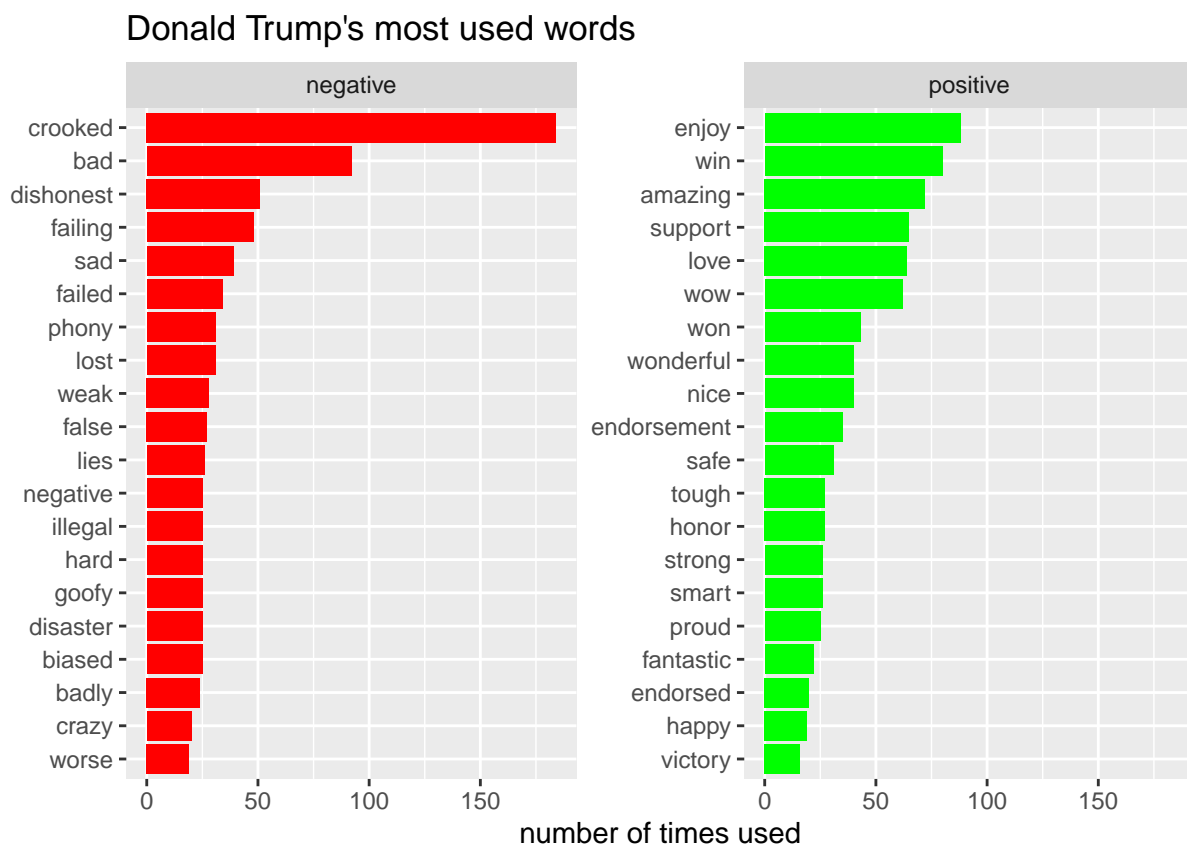
```
## # A tibble: 6,786 x 2
##   word      sentiment
##   <chr>    <chr>
## 1 2-faces  negative
## 2 abnormal negative
## 3 abolish negative
## 4 abominable negative
## 5 abominably negative
## 6 abominate negative
## 7 abomination negative
## 8 abort    negative
## 9 aborted  negative
```



```
## 10 aborts      negative
## # ... with 6,776 more rows
```

We will use this lexicon to do a sentiment analysis over the words that Trump uses the most. First we tidy the data and convert previous tables into tibbles. We do this for technical reasons to be able to use certain functions.

```
library(lubridate)
TrumpTidy1 <- tidy(TrumpCorpus1)
DocMetaTrump1 <- meta(TrumpCorpus1)
DocMetaTrump1$date <- date(DocMetaTrump1$time)
TrumpTidy1$date <- DocMetaTrump1$date
NoNamesTidy <- bind_rows(trump=TrumpTidy1, .id="candidate")
Words <- NoNamesTidy %>% unnest_tokens(word, text)
Bing <- Words %>% inner_join(get_sentiments("bing"), by="word")
b1 <- Bing %>% filter(candidate=="trump") %>% count(word, sentiment, sort=TRUE) %>%
  group_by(sentiment) %>% arrange(desc(n)) %>% slice(1:20) %>%
  ggplot(aes(x=reorder(word, n), y=n)) +
  geom_col(aes(fill=sentiment), show.legend=FALSE) +
  coord_flip() +
  facet_wrap(~sentiment, scales="free_y") +
  labs(x="", y="number of times used", title="Donald Trump's most used words") +
  scale_fill_manual(values = c("positive"="green", "negative"="red"))
b1
```



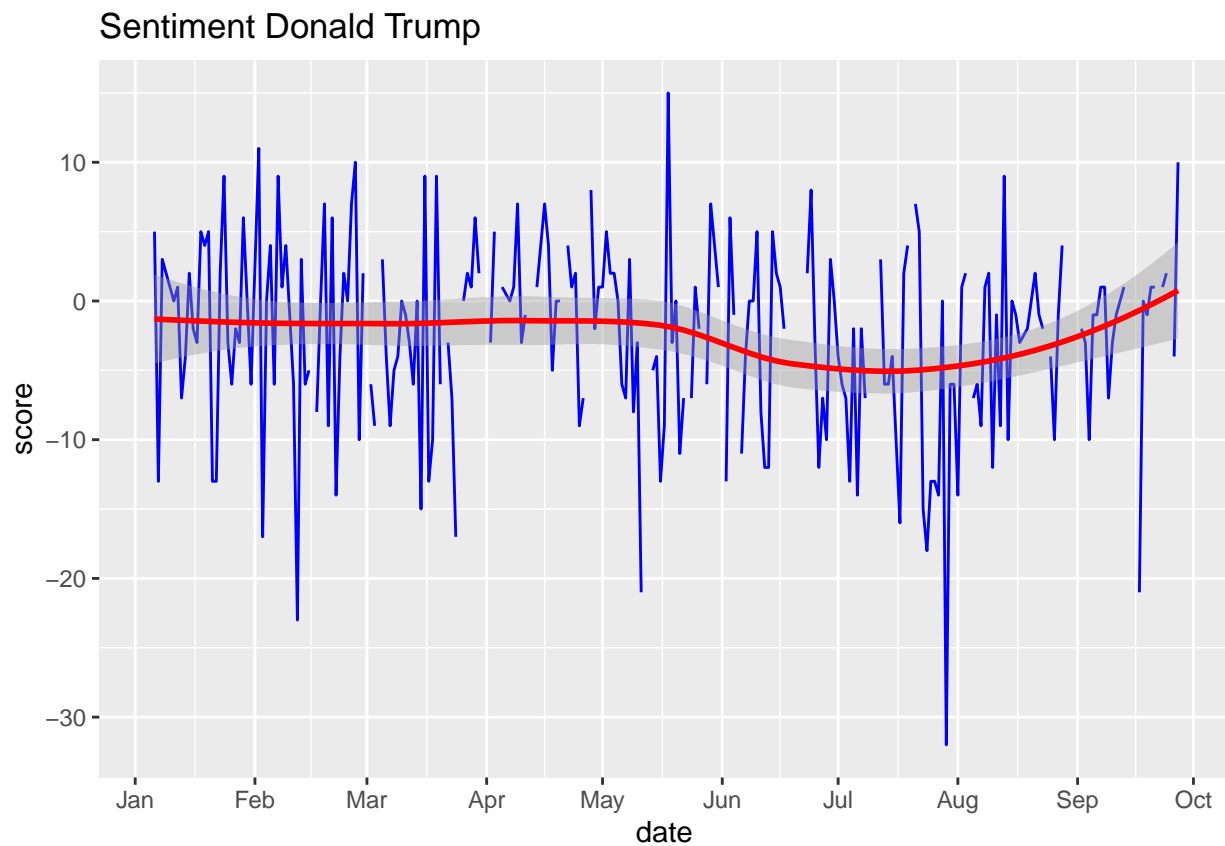
**Question 11** Do the same for Clinton! Which are the top20 positive and negative words from the *bing* lexicon that Clinton has used



## Time series of sentiment

Did the way Trump speak in his campaign changed over time? We can use the date for that.

```
library(tidy)
t1 <- Bing %>% filter(candidate=="trump") %>% group_by(date) %>% count(sentiment) %>%
spread(sentiment, n) %>% mutate(score=positive-negative) %>%
ggplot(aes(x=date, y=score)) +
scale_x_date(limits=c(as.Date("2016-01-05"),
as.Date("2016-09-27")), date_breaks = "1 month", date_labels = "%b") +
geom_line(stat="identity", col="blue") + geom_smooth(col="red") +
labs(title="Sentiment Donald Trump")
t1
```



**Question 12** Do the same for Clinton! How did her sentiment changed over time

**Question 13** Check how to do it for the *AFIN* and *nrc* lexicons

## References

- Bruin, Erik. 2018. [Text Mining the Clinton and Trump Election Tweets](#).
- Hammer, Ben. 2017. [2016 U.S. Presidential Campaign Texts and Polls. Debate and Speech Transcripts & Voter Group Polls](#).