LAB 6 - Dictionary Methods

Iulia Cioroianu

July 2021

This tutorial is partially based upon the sentiment analysis using dictionaries Quanteda Tutorial available here: https://tutorials.quanteda.io/advanced-operations/targeted-dictionary-analysis/

```
library(quanteda)
```

```
## Warning in stringi::stri_info(): Your current locale is not in the list
## of available locales. Some functions may not work properly. Refer to
## stri_locale_list() for more details on known locale specifiers.

## Warning in stringi::stri_info(): Your current locale is not in the list
## of available locales. Some functions may not work properly. Refer to
## stri_locale_list() for more details on known locale specifiers.

## Package version: 3.0.0
## Unicode version: 13.0
## ICU version: 66.1

## Parallel computing: 16 of 16 threads used.

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

Example 1: Applying a simple user-defined dictionary to a simple text

You can define your own dictionary by passing a named list of characters to dictionary().

Here is a passage of John Donne's poem "No Man is an Island":

```
mytext="No man is an island entire of itself; every man
is a piece of the continent, a part of the main;
if a clod be washed away by the sea, Europe
is the less, as well as if a promontory were, as
well as any manner of thy friends or of thine
own were; any man's death diminishes me,
because I am involved in mankind."
```

Defining our dictionary:

```
mydict <- dictionary(list(Human = c('man*', 'friend*'),
                          Nature = c('island*', 'sea*', 'clod*', 'promontor*')))
print(mydict)
```

```
## Dictionary object with 2 key entries.
## - [Human]:
##   - man*, friend*
## - [Nature]:
```

```
##   - island*, sea*, clod*, promontor*
```

First step: read in the corpus

```
mycorpus <- corpus(mytext)
```

Second step: tokenize and process

```
mytext_tok <- tokens_tolower(tokens(mycorpus,
                     remove_punct=TRUE))
```

Using the lookup method - find dictionary keywords within the tokens:

```
dict_toks <- tokens_lookup(mytext_tok, dictionary = mydict)
dict_toks
```

```
## Tokens consisting of 1 document.
## text1 :
##  [1] "Human"  "Nature" "Human"  "Nature" "Nature" "Nature" "Human"  "Human"
##  [9] "Human"  "Human"
```

Put them in a DFM:

```
dfmat_dict <- dfm(dict_toks)
dfmat_dict
```

```
## Document-feature matrix of: 1 document, 2 features (0.00% sparse) and 0 docvars.
##        features
## docs    human nature
##   text1     6      4
```

This is the same as specifying the dictionary and keeping only dictionary terms in a DFM.

```
dfmat_dict <- dfm(mytext_tok, dictionary = mydict, verbose=TRUE)
```

```
## Creating a dfm from a tokens input...
##  ...lowercasing
##  ...found 1 document, 45 features
## Warning: 'dictionary' and 'thesaurus' are deprecated; use dfm_lookup() instead
## ...
## applying a dictionary consisting of 2 keys
##  ...complete, elapsed time: 0.005 seconds.
## Finished constructing a 1 x 2 sparse dfm.
```

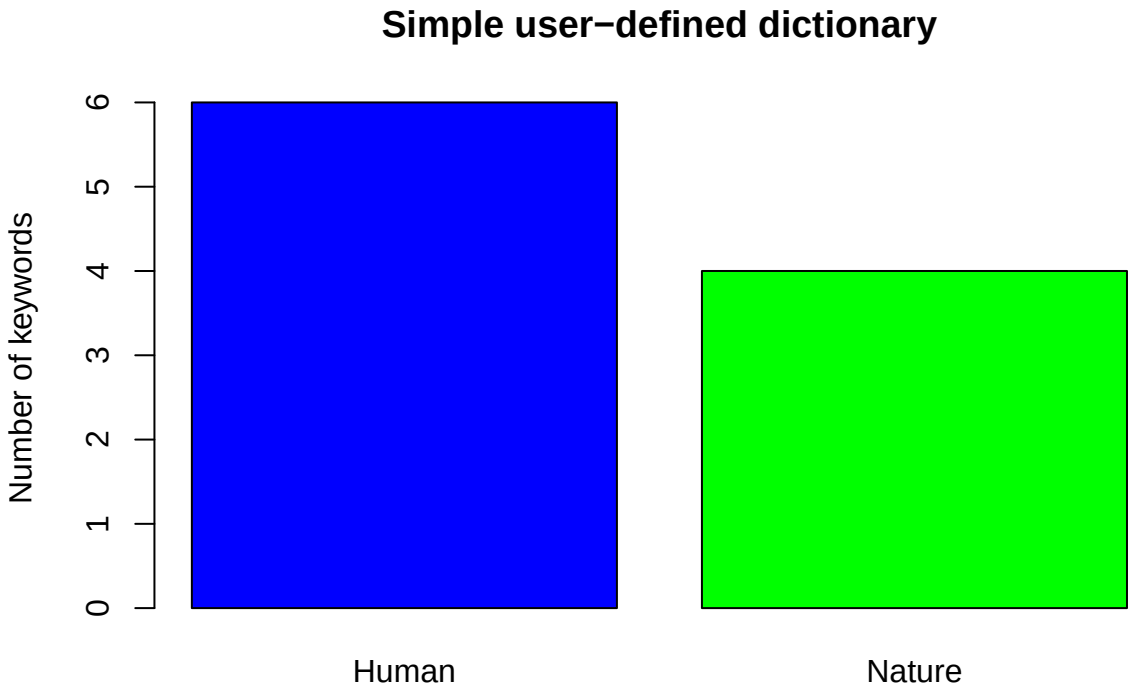
```
dfmat_dict
```

```
## Document-feature matrix of: 1 document, 2 features (0.00% sparse) and 0 docvars.
##        features
## docs    Human Nature
##   text1     6      4
```

Or you can use dfm_lookup() if you want to apply the dictionary after you have created the corpus.

Let's do a simple bar plot of the results.

```
barplot(as.vector(dfmat_dict), main="Simple user-defined dictionary",
        names=colnames(dfmat_dict), ylab="Number of keywords", col = c("blue", "green"))
```



Do the results look good to you? What issue do you spot? How can we fix it?

Example 2: Applying a user-defined dictionary to a corpus

We need to read in a subsample of the quanteda-stored Inaugural Corpus.

Let's learn more about this dataset:

```
?data_corpus_inaugural
head(docvars(data_corpus_inaugural), 10)
```

```
##    Year  President   FirstName                Party
## 1  1789 Washington      George                 none
## 2  1793 Washington      George                 none
## 3  1797      Adams        John           Federalist
## 4  1801  Jefferson      Thomas Democratic-Republican
## 5  1805  Jefferson      Thomas Democratic-Republican
## 6  1809    Madison       James Democratic-Republican
## 7  1813    Madison       James Democratic-Republican
## 8  1817     Monroe       James Democratic-Republican
## 9  1821     Monroe       James Democratic-Republican
## 10 1825      Adams John Quincy Democratic-Republican
```

```
# Keep only most recent speeches
inaug_corp <- corpus_subset(data_corpus_inaugural, Year>1992)
```

Tokenize it:

```
tok_inaug <- tokens(inaug_corp,
                    remove_numbers=TRUE,
                    remove_punct=TRUE,
                    remove_symbols=TRUE,
```

```
                    remove_separators=TRUE,
                    verbose=TRUE)
```

```
## Creating a tokens object from a corpus input...
##  ...starting tokenization
##  ...1993-Clinton to 2021-Biden.txt
##  ...preserving hyphens
##  ...preserving social media tags (#, @)
##  ...segmenting into words
##  ...2,883 unique types
##  ...removing separators, punctuation, symbols, numbers
##  ...complete, elapsed time: 0.037 seconds.
## Finished constructing tokens from 8 documents.
```

```
head(tok_inaug, n=3)
```

```
## Tokens consisting of 3 documents and 4 docvars.
## 1993-Clinton :
##  [1] "My"        "fellow"    "citizens"  "today"     "we"        "celebrate"
##  [7] "the"       "mystery"   "of"        "American"  "renewal"   "This"
## [ ... and 1,586 more ]
##
## 1997-Clinton :
##  [1] "My"           "fellow"       "citizens"     "At"           "this"
##  [6] "last"         "presidential" "inauguration" "of"           "the"
## [11] "20th"         "century"
## [ ... and 2,145 more ]
##
## 2001-Bush :
##  [1] "President"     "Clinton"       "distinguished" "guests"
##  [5] "and"           "my"            "fellow"        "citizens"
##  [9] "the"           "peaceful"      "transfer"      "of"
## [ ... and 1,571 more ]
```

Extra processing:

```
# Turn tokens to lower case
tok_inaug <- tokens_tolower(tok_inaug)

# Remove stopwords
tok_inaug <- tokens_remove(tok_inaug, pattern = stopwords('en'))
```

Create our dictionary:

```
mydict <- dictionary(list(Conflict = c("war*", "conflict*", "peace", "terror*"),
                          Environment = c("environment*", "warming")))
```

Create the document feature matrix.

```
dfm_inaug <- dfm(tok_inaug)
```

Apply the dictionary:

```
dfm_dict <- dfm_lookup(dfm_inaug, dictionary=mydict)
#?dfm_lookup # have a look at what this function does
```

If you want to get percentages instead of raw counts you can divide by the total number of tokens in each document but not now, since we're grouping at the next step

```
#dfm_dict <- dfm_dict/ntoken(dfm_inaug)
```

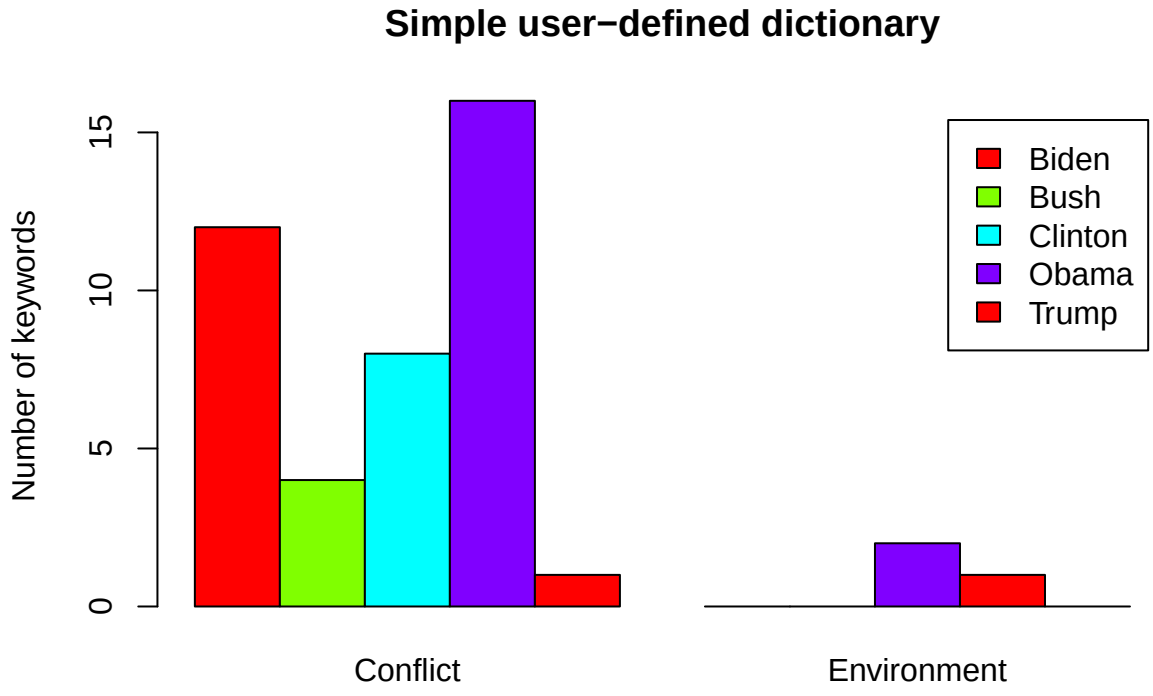
Group the dataset by president:

```
dfm_by_pres <- dfm_group(dfm_dict, groups=President)
#?dfm_group
```

Group by president

Let's plot it:

```
barplot(as.matrix(dfm_by_pres), beside=TRUE,
        main="Simple user-defined dictionary",
        ylab="Number of keywords",
        legend=rownames(dfm_by_pres),
        col=rainbow(4))
```



Example 3: Importing and using a pre-defined dictionary

Let's say that you want to use a specific dictionary that is not included in Quanteda. We have such dictionary in the files section, the Moral Foundations Dictionary.

Import the Moral Foundations dictionary, specifing the format, this case .dic is LIWC format.

```
mfd_dict <- dictionary(file = "mfd2.0.dic",
                       format = "LIWC")
mfd_dict
```

```
## Dictionary object with 10 key entries.
## - [care.virtue]:
##   - alleviate, alleviated, alleviates, alleviating, alleviation, altruism, altruist, beneficence, ben
## - [care.vice]:
##   - abused, abuser, abusers, abuses, abusing, ache, ached, aches, aching, achingly, afflict, afflicte
## - [fairness.virtue]:
##   - avenge, avenged, avenger, avengers, avenges, avenging, been objective, being objective, civil rig
## - [fairness.vice]:
##   - am partial, bamboozle, bamboozled, bamboozles, bamboozling, be partial, been partial, behind thei
## - [loyalty.virtue]:
##   - all for one, allegiance, allegiances, allegiant, allied, allies, ally, belong, belonged, belongin
## - [loyalty.vice]:
##   - against us, apostate, apostates, backstab, backstabbed, backstabber, backstabbers, backstabbing,
## [ reached max_nkey ... 4 more keys ]
```

Create the DFM:

```
mfd_dfm <- dfm(tok_inaug) %>% dfm_lookup(dictionary = mfd_dict) %>%
  dfm_group(groups=President)
#mfd_dfm
```

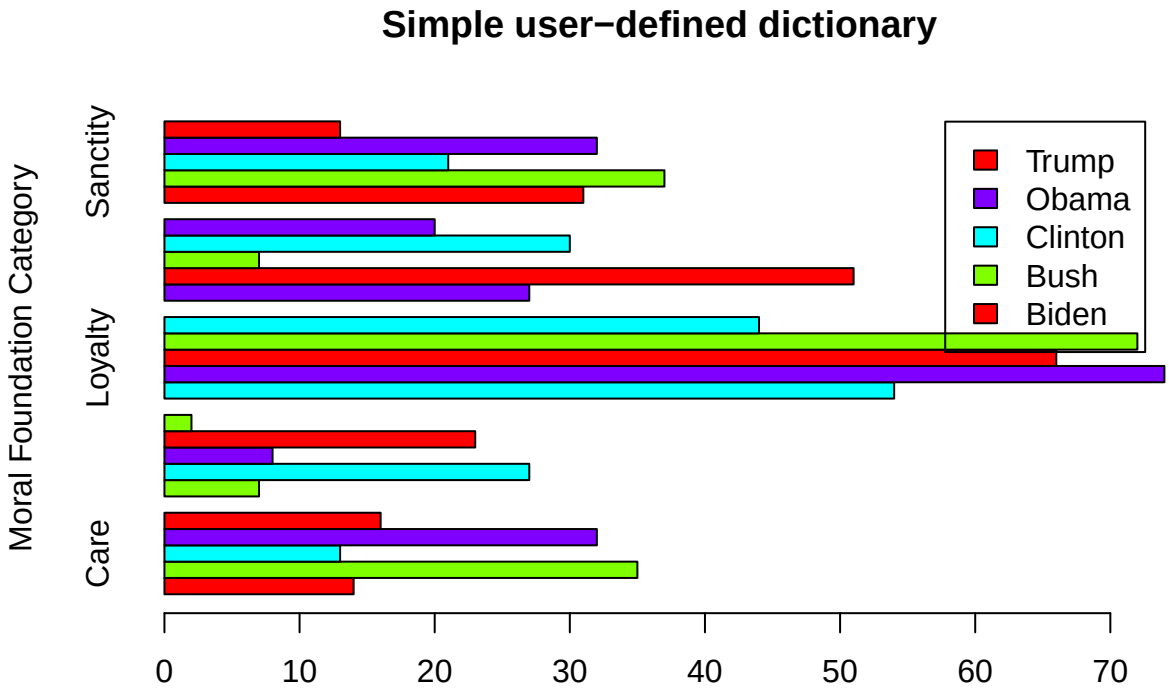
Let's keep only variables that refer to virtues:

```
mfd_dfm <- mfd_dfm[,grep("virtue", colnames(mfd_dfm))]
mfd_dfm
```

```
## Document-feature matrix of: 5 documents, 5 features (0.00% sparse) and 3 docvars.
##          features
## docs      care.virtue fairness.virtue loyalty.virtue authority.virtue
##   Biden            14               7             54               27
##   Bush             35              27             74               51
##   Clinton          13               8             66                7
##   Obama            32              23             72               30
##   Trump            16               2             44               20
##          features
## docs      sanctity.virtue
##   Biden                31
##   Bush                 37
##   Clinton              21
##   Obama                32
##   Trump                13
```

Display the info in a basic barplot:

```
barplot(as.matrix(mfd_dfm), beside=TRUE,
        main="Simple user-defined dictionary",
        ylab="Moral Foundation Category",
        names=c("Care", "Fairness", "Loyalty", "Authority", "Sanctity"),
        legend=rownames(dfm_by_pres),
        horiz=TRUE, col=rainbow(4))
```



Exercise

Work with your group to create your own LIWC-alike dictionary for two categories: * 1. the economy * 2. education

Save is as a .dic dictionary. Use the sample_mfd.dic as an example of the format. Note that you can edit that file directly.

Apply your dictionary to the subset of inaugural speeches data from 1992 onwards, grouping by president.

Plot the results. ##

Example 4: Using a pre-defined dictionary for sentiment analysis

```
library(quanteda)
library(quanteda.corpora)
library(lubridate)
```

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

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

This corpus contains 6,000 Guardian news articles from 2012 to 2016.

```
corp_news <- download('data_corpus_guardian')
```

Analyse date variable

```
corp_news$date[1]
```

```
## [1] "2016-02-26"
```

Seprate it into year, month, week using the lubridate package

```
corp_news$year <- year(corp_news$date)
corp_news$month <- month(corp_news$date)
corp_news$week <- week(corp_news$date)
```

What values do we have for the year variable?

```
summary(corp_news$year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2012    2014    2015    2015    2016    2016
```

Select only year 2016:

```
corp_news <- corpus_subset(corp_news, 'year' >= 2016)
```

Tokenize corpus:

```
toks_news <- tokens(corp_news, remove_punct = TRUE)
```

Remember, you can use tokens_lookup() or dfm_lookup() to count dictionary values. Quanteda contains the Lexicoder Sentiment Dictionary created by Young and Soroka, so you can perfrom sentiment analysis of English texts right away.

```
lengths(data_dictionary_LSD2015)
```

```
##     negative     positive neg_positive neg_negative
##         2858         1709         1721         2860
```

Lookup in positive and negative terms in the news tokens.

```
toks_news_lsd <- tokens_lookup(toks_news, dictionary =  data_dictionary_LSD2015[1:2])
head(toks_news_lsd, 2)
```

```
## Tokens consisting of 2 documents and 12 docvars.
## text136751 :
## [1] "positive" "positive"
##
## text118588 :
##  [1] "positive" "positive" "negative" "negative" "negative" "positive"
##  [7] "negative" "positive" "positive" "negative" "positive" "positive"
## [ ... and 24 more ]
```

Put them into a document frequency matrix

```
dfmat_news_lsd <- dfm(toks_news_lsd)
head(dfmat_news_lsd, 2)
```

```
## Document-feature matrix of: 2 documents, 2 features (25.00% sparse) and 12 docvars.
##             features
## docs         negative positive
##   text136751        0        2
##   text118588       11       25
```

Get summary statistics:

```
summary(as.matrix(dfmat_news_lsd))
```

```
##     negative         positive
##  Min.   :  0.00   Min.   :  0.00
##  1st Qu.: 12.00   1st Qu.: 12.00
```

```
##  Median : 22.00   Median : 21.00
##  Mean   : 29.24   Mean   : 27.04
##  3rd Qu.: 37.00   3rd Qu.: 33.00
##  Max.   :444.00   Max.   :553.00
```

Example 5: Targeted sentiment analysis

You can use tokens_select() with the window argument to perform more targeted sentiment analysis.

Let's evaluate the sentiment around mentions of the European Union.

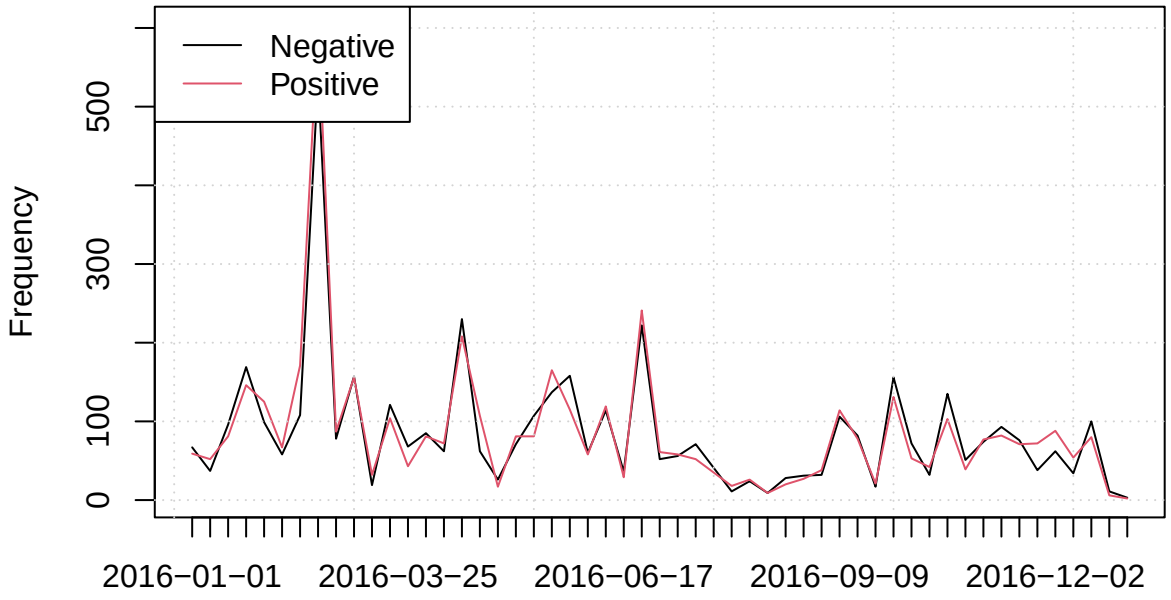
```
# Define keyterms:
eu <- c('EU', 'europ*', 'european union')
# Define relevant tokens - 10 tokens before and after the keyword
toks_eu <- tokens_keep(toks_news, pattern = phrase(eu), window = 10)
#toks_eu
```

Put it into a dataframe matrix:

```
dfmat_eu_lsd <- dfm(toks_eu) %>% dfm_lookup(dictionary = data_dictionary_LSD2015[1:2]) %>%
  dfm_group(week)
#dfmat_eu_lsd
```

Plot the positive and negative sentiment side by side

```
# Draw the line plot:
matplot(dfmat_eu_lsd, type = 'l', xaxt = 'n', lty = 1, ylab = 'Frequency')
# Draw the grid:
grid()
# Draw the time axis:
axis(1, seq_len(ndoc(dfmat_eu_lsd)), ymd("2016-01-01") + weeks(seq_len(ndoc(dfmat_eu_lsd)) - 1))
# Draw the legend:
legend('topleft', col = 1:2, legend = c('Negative', 'Positive'), lty = 1, bg = 'white')
```



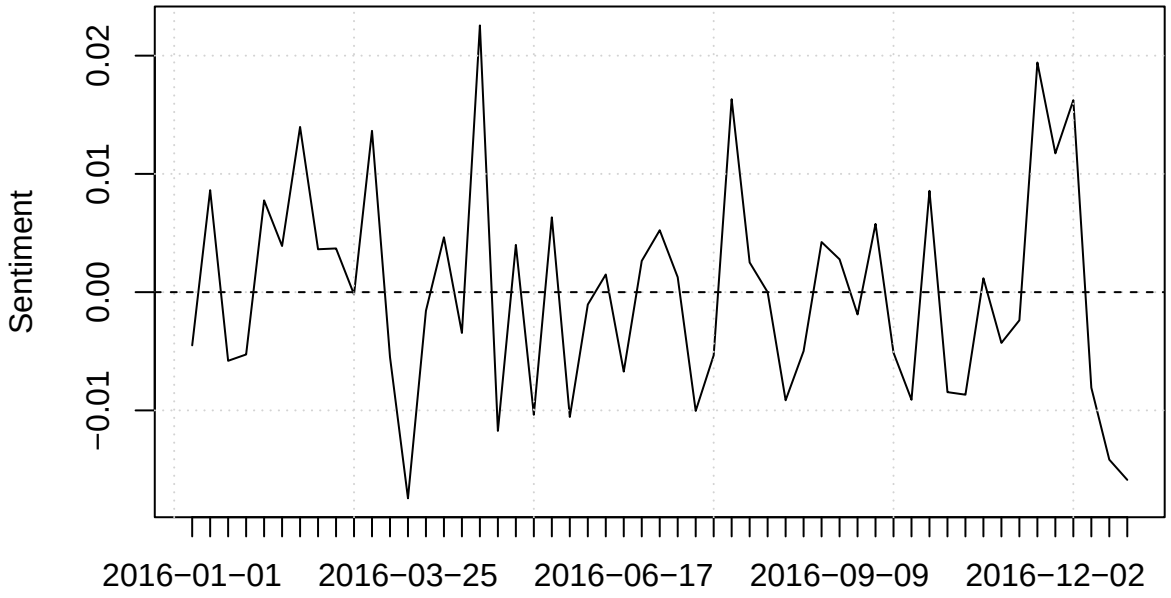
Plot relative sentiment.

Count the number of tokens

```
n_eu <- ntoken(dfm(toks_eu) %>% dfm_group(week))
```

Plot the difference between positive and negative divided by the total number"

```
plot((dfmat_eu_lsd[,2] - dfmat_eu_lsd[,1]) / n_eu,
     type = 'l', ylab = 'Sentiment', xlab = '', xaxt = 'n')
# Draw the axis:
axis(1, seq_len(ndoc(dfmat_eu_lsd)), ymd("2016-01-01") + weeks(seq_len(ndoc(dfmat_eu_lsd)) - 1))
# Draw the grid
grid()
# Draw the reference line
abline(h = 0, lty = 2)
```

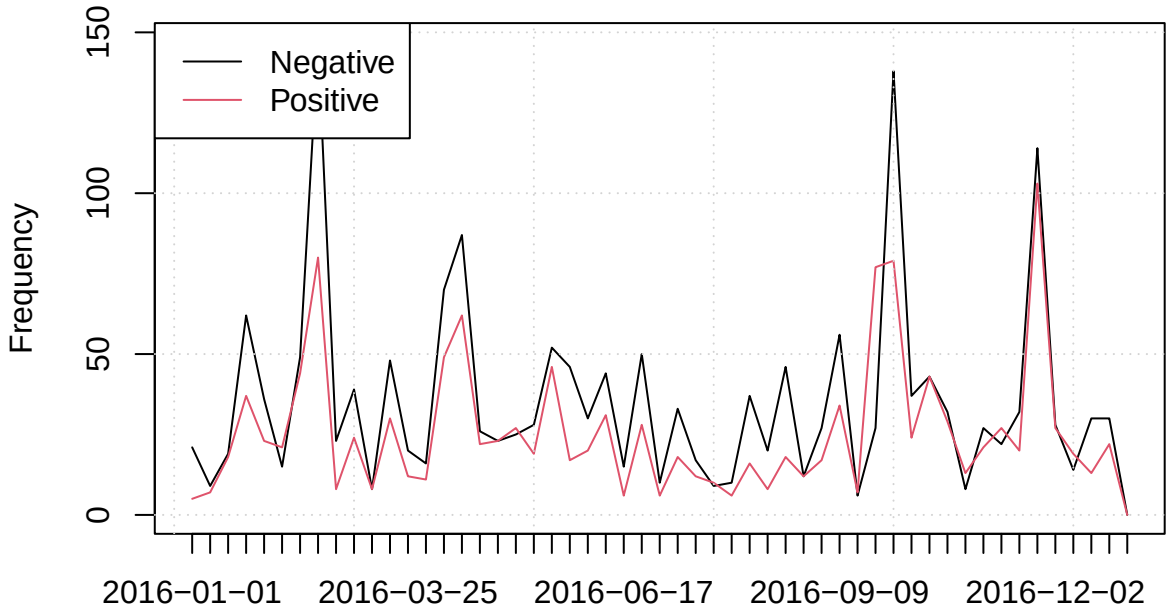


Exercise:

Work with your group to define, measure and plot the use of terms related to immigration in a window of 15 tokens before and after the keyword in this corpus. ##

```
immig <- c('immig*', 'migra*')
toks_immig <- tokens_keep(toks_news, pattern = phrase(immig), window = 10)
dfmat_immig_lsd <- dfm(toks_immig) %>% dfm_lookup(dictionary = data_dictionary_LSD2015[1:2]) %>%
  dfm_group(group = week, fill = TRUE)

matplot(dfmat_immig_lsd, type = 'l', xaxt = 'n', lty = 1, ylab = 'Frequency')
grid()
axis(1, seq_len(ndoc(dfmat_immig_lsd)), ymd("2016-01-01") + weeks(seq_len(ndoc(dfmat_immig_lsd)) - 1))
legend('topleft', col = 1:2, legend = c('Negative', 'Positive'), lty = 1, bg = 'white')
```




```
n_immig <- ntoken(dfm(toks_immig) %>% dfm_group(group = week))

plot((dfmat_immig_lsd[,2] - dfmat_immig_lsd[,1]) / n_immig,
     type = 'l', ylab = 'Sentiment', xlab = '', xaxt = 'n')
axis(1, seq_len(ndoc(dfmat_immig_lsd)), ymd("2016-01-01") + weeks(seq_len(ndoc(dfmat_immig_lsd)) - 1))
grid()
abline(h = 0, lty = 2)
```