TEXT MINING WITH R

Research Data Management Support

ABOUT TEXT MINING

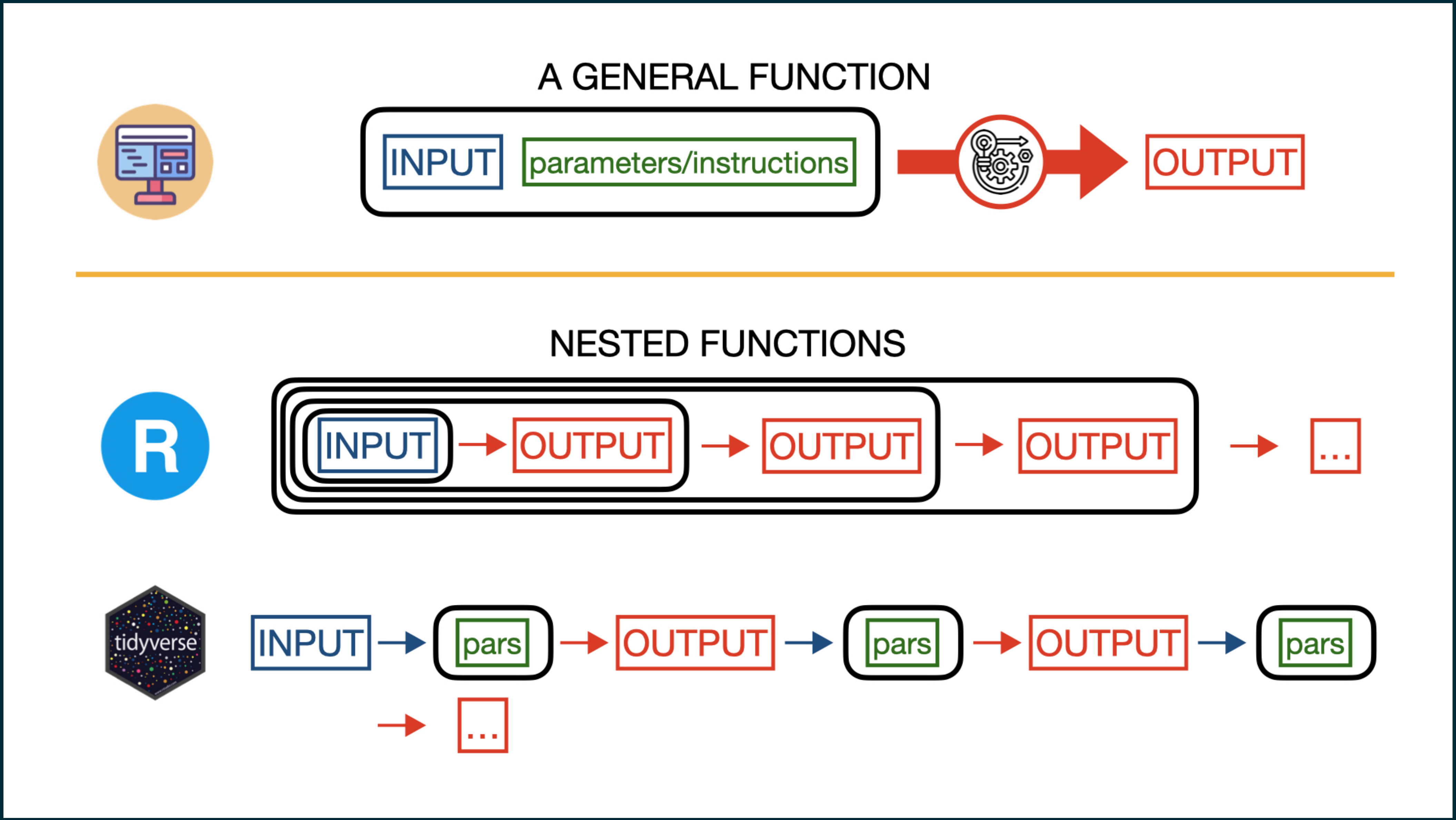
- Text mining refers to the process of extracting (*mining*) information and insights from text;
- Text mining can be extremely useful when looking for any sort of pattern, trend, or relationships in large volumes of text data (articles, documents, emails, social media posts, etc);
- The main challenge of text mining is obtaining meaningful information from unstructured and ambiguous material.

R PACKAGES FOR TEXT MINING

```r
library(tidyverse)
library(tidytext)
library(wordcloud)
```

- **tidyverse**: this is an "opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures";
- **tidytext**: an R package for text mining based on the tidy data principles;
- **wordcloud**: a package to generate word cloud plots.

TIDYVERSE PIPELINE



PLAIN R SYNTAX

```r
output1 <- func1(data, pars1)
output2 <- func2(output1, pars2)
output3 <- func3(output2, pars3)
output4 <- func4(output3, pars4)
```

or

```r
output4 <-
  func4(func3(func2(func1(data,pars1),pars2),pars3),pars4)
```

TIDYVERSE SYNTAX

```r
output4 <- data %>%
  func1(pars1) %>%
  func2(pars2) %>%
  func3(pars3) %>%
  func4(pars4)
```

READING DATA

```r
data_file_name <- '../../data/ianalyzer_query.csv'

data_df <- read_delim(data_file_name,
    delim = ";",
    escape_double = FALSE,
    col_types = cols(`date-pub` = col_date(format = "%B %d, %Y"),
        issue = col_integer()), trim_ws = TRUE)

print(nrow(data_df))
```

```
[1] 1532
```

```r
print(colnames(data_df))
```

```
[1] "author"   "category" "content"  "date-pub" "edition"  "issue"    "query"
[8] "title"    "volume"
```

if you are in truble with ``read_delim()``, get help from R studio

TOKENIZATION

- **Tokenization** is process of dividing a string of text into meaningful units called **tokens**;
- A token can be a word, a phrase, a paragraph, or a single character depending on the nature of our analysis;
- In R tokenization is performed using the tidytext function `unnest_tokens()`.

TOKENIZATION

```r
tidy_content <- data_df %>% unnest_tokens(word, content, token="words")

tidy_content
```

```
# A tibble: 1,549,578 × 9
   author                category `date-pub` edition issue query title volume
word
   <chr>                 <chr>    <date>     <lgl>   <int> <chr> <chr> <lgl>
<chr>
 1 ['FROM A SPECIAL … ['News'] 1962-11-05 NA      55540 time… Euro… NA
from
 2 ['FROM A SPECIAL … ['News'] 1962-11-05 NA      55540 time… Euro… NA     a
 3 ['FROM A SPECIAL … ['News'] 1962-11-05 NA      55540 time… Euro… NA
spec…
 4 ['FROM A SPECIAL … ['News'] 1962-11-05 NA      55540 time… Euro… NA
corr…
 5 ['FROM A SPECIAL … ['News'] 1962-11-05 NA      55540 time… Euro… NA     at
 6 ['FROM A SPECIAL … ['News'] 1962-11-05 NA      55540 time… Euro… NA
this
```

CLEANING UP DATA

Checking if the column *issue* has any Na

```r
are_there_na <- any(is.na(tidy_content$issue))
are_there_na
```

```
[1] TRUE
```

let’s clean up

```r
tidy_content <- tidy_content[!is.na(tidy_content$issue), ]
```

and let’s check again

```r
are_there_na <- any(is.na(tidy_content$issue))
are_there_na
```

```
[1] FALSE
```

REMOVING STOP WORDS

Unstructured data can contain a lot of irrelevant information. The most common words in a text are words that have very little meaning, such as “the”, “and”, “a”, etc. These words are referred to as **stop words** and removing stop words from text (in a way or another) is a fundamental step of text mining.

```
data(stop_words)

tidy_clean_content <- tidy_content %>% anti_join(stop_words)

tidy_clean_content
```

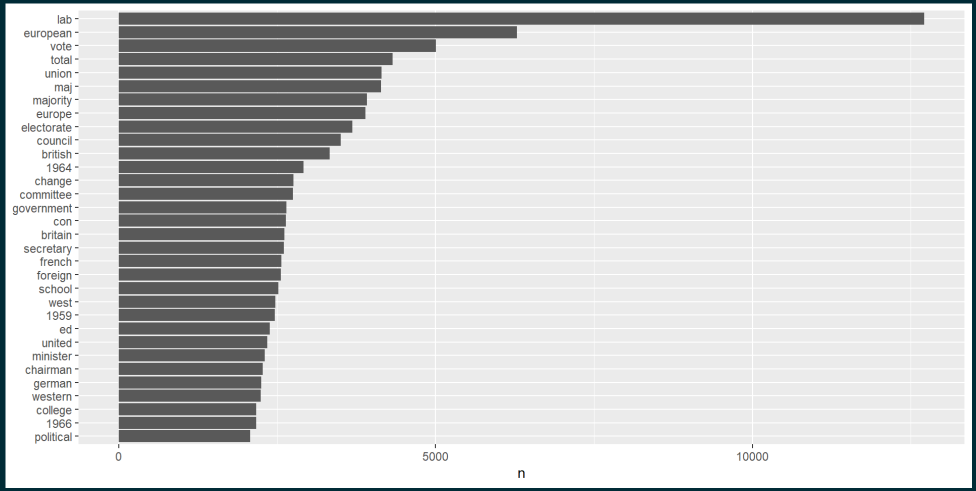
```
# A tibble: 801,754 × 9
   author              category `date-pub` edition issue query title volume
word
   <chr>               <chr>    <date>     <lgl>   <int> <chr> <chr> <lgl>
<chr>
 1 ['FROM A SPECIAL … ['News'] 1962-11-05 NA      55540 time… Euro… NA
spec…
 2 ['FROM A SPECIAL … ['News'] 1962-11-05 NA      55540 time… Euro… NA
corr…
 3 ['FROM A SPECIAL … ['News'] 1962-11-05 NA      55540 time… Euro… NA
junc…
 4 ['FROM A SPECIAL … ['News'] 1962-11-05 NA      55540 time… Euro… NA
focus
 5 ['FROM A SPECIAL … ['News'] 1962-11-05 NA      55540 time… Euro… NA
```

COUNTING WORDS

```r
word_count <- tidy_clean_content %>%
  count(word) %>%
  filter(n > 2000) %>%
  mutate(word = reorder(word, n))

word_count_plot <-
  word_count %>%
  ggplot(aes(n, word)) +
  geom_col() +
  labs(y = NULL)

word_count_plot
```

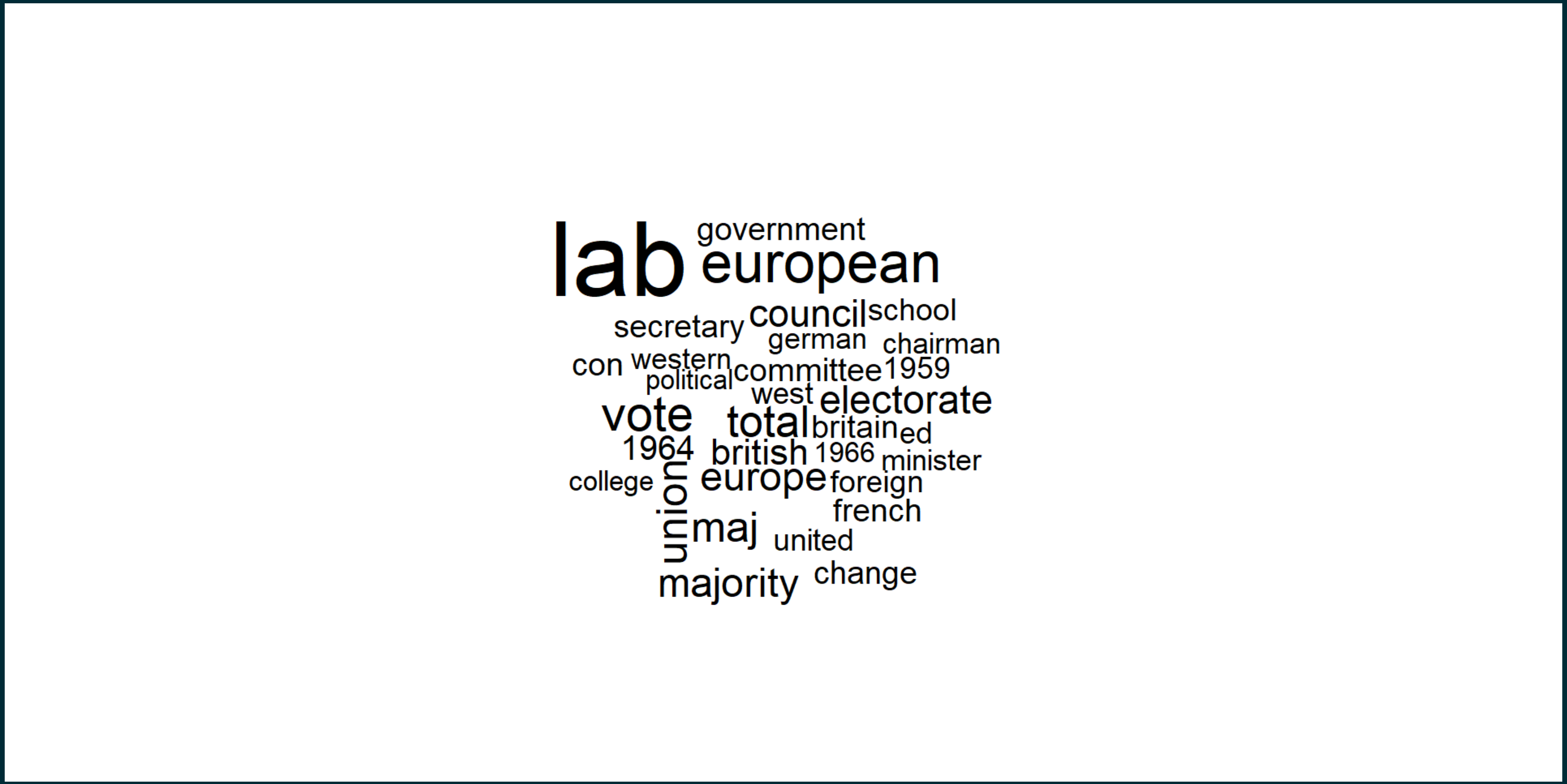


WORD CLOUD VISUALIZATION

```r
word_cloud_plot <-
  word_count %>%
  with(wordcloud(word, n))
```

```r
word_cloud_plot
```

```
NULL
```



SENTIMENT ANALYSIS

- **sentiment analysis** has the goal of systematically identify, extract, quantify, and study affective states and subjective information from text;
- Sentiment analysis is based on the assumption that we can view a text as a combination of individual words (the text sentiment will be the sum of the sentiment of its individual words);
- To perform sentiment analysis, we need a reference database of words called **lexicon** assigning a sentiment to each word.

LEXICON AND JOY WORDS

```r
nrc_lexicon_df <- read.table("../../lexicons/NRC_lexicon.txt", header = FAL

joy_words <- nrc_lexicon_df  %>%
  filter(emotion == "joy", score == 1)

joy_words
```

```
              word emotion score
1       absolution     joy     1
2        abundance     joy     1
3         abundant     joy     1
4         accolade     joy     1
5    accompaniment     joy     1
6       accomplish     joy     1
7     accomplished     joy     1
8          achieve     joy     1
9      achievement     joy     1
10         acrobat     joy     1
11       admirable     joy     1
12      admiration     joy     1
13        adorable     joy     1
14       adoration     joy     1
```

COMPUTING JOY WORDS FRACTION

$$Frac_{joy}(issue) = \frac{\text{Number of joy words per issue}}{\text{Number of words per issue}} * 100$$

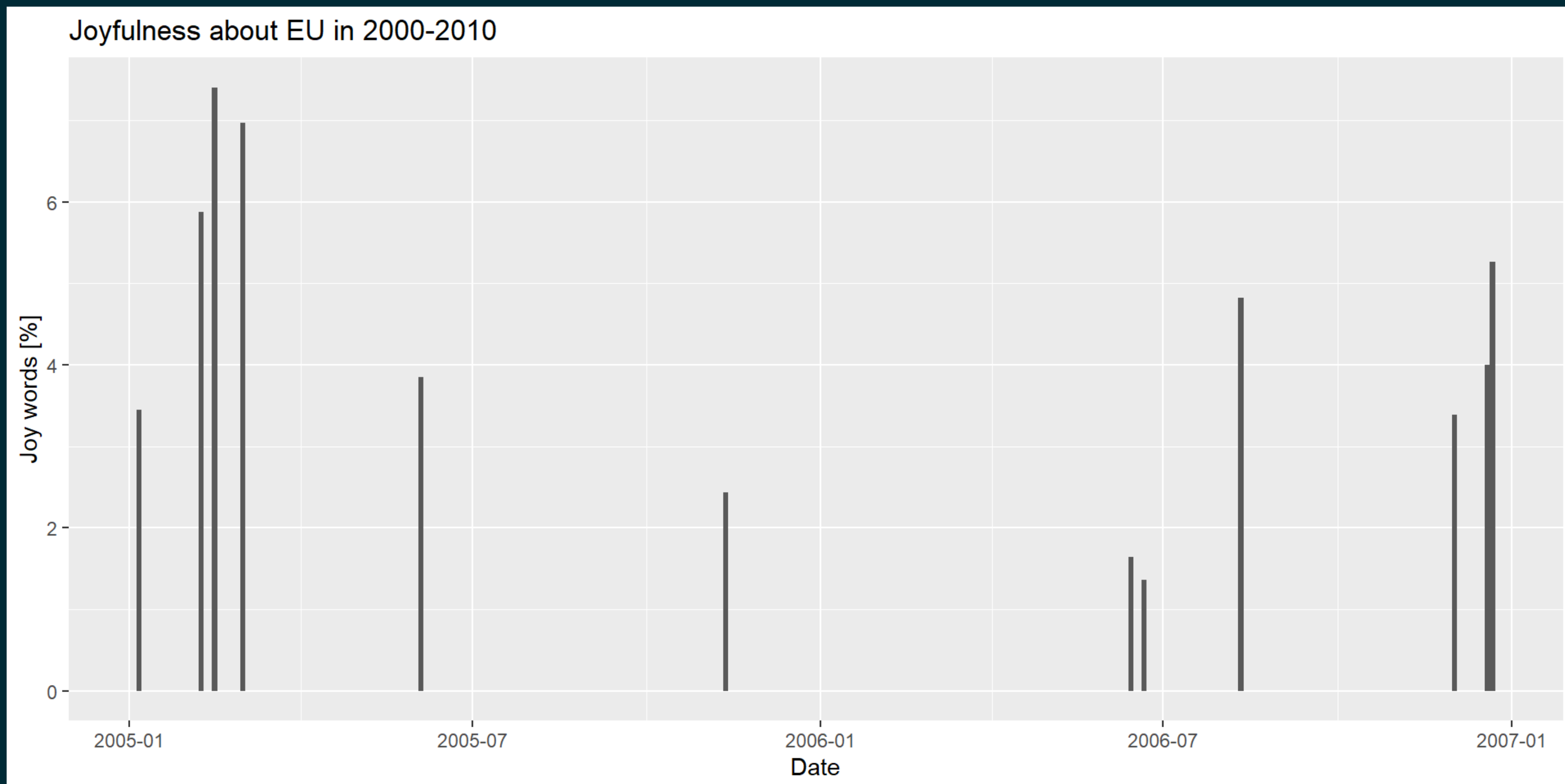
```r
issue_df <- tidy_clean_content %>%
  filter(`date-pub`>='2000-01-01' & `date-pub` < '2010-01-01') %>%
  group_by(issue) %>%
  reframe(words_per_issue = n(), date= `date-pub`) %>%
  unique()

issue_joy_df <- tidy_clean_content %>%
  filter(`date-pub`>='2000-01-01' & `date-pub` < '2010-01-01') %>%
  inner_join(joy_words) %>%
  group_by(issue) %>%
  reframe(joy_words_per_issue = n())

issue_tot_df <- merge(issue_df, issue_joy_df, by='issue')
```

```r
percent_of_joy_plot <-
  issue_tot_df %>%
  mutate(per_cent_joy=joy_words_per_issue/words_per_issue*100) %>%
  ggplot(aes(x = date, y = per_cent_joy) )+
  geom_col() +
  labs(x = "Date", y = "Joy words [%]",
       title = "Joyfulness about EU in 2000-2010")

percent_of_joy_plot
```



COMPUTING "TOTAL JOY" FRACTION

$$Frac_{joy} = \frac{\text{Number of joy words}}{\text{Number of words}} * 100[\%]$$

```
distinct_words <- tidy_clean_content %>%
  distinct(word)

total_dis_words <- distinct_words %>%
  nrow()
total_dis_joy_words <- distinct_words %>%
  inner_join(joy_words,by='word') %>%
  nrow()

total_joy <- (total_dis_joy_words/total_dis_words)*100
print(paste(total_joy,' [%]'))
```

```
[1] "0.534795712569965  [%]"
```

ANALYZING WORD AND DOCUMENT FREQUENCY: TF-IDF

$$\text{tf-idf} = \text{term frequency} * idf$$

$$idf(term) = log\left(\frac{n_{documents}}{n_{documents\ containing\ term}}\right)$$

COMPUTING TERM FREQUENCY

- Let's compute and store in two DataFrames the frequency of occurrence of each word and the total number of words per issue.

```
issue_words <- data_df %>%
  unnest_tokens(word, content) %>%
  count(issue, word)

issue_words <- na.omit(issue_words)

total_words <- issue_words %>%
  group_by(issue) %>%
  summarize(total = sum(n))

issue_total_words <- left_join(issue_words, total_words) %>%
  arrange(desc(issue))
```

COMPUTING TERM FREQUENCY

```
unique_issues <- issue_total_words %>%
  filter(total>10000) %>%
  distinct(issue)

first_6_unique_issues <- unique_issues %>% slice(1:6)

issue_total_words6 <- issue_total_words %>%
  semi_join(first_6_unique_issues, by="issue") %>%
  mutate(issue=as.character(issue))

freq_per_issue_plot <-
  issue_total_words6 %>%
  ggplot(aes(n/total, fill = issue)) +
  geom_histogram(show.legend = FALSE) +
  xlim(NA, 0.0005) +
  facet_wrap(~issue, ncol = 2, scales = "free_y")

freq_per_issue_plot
```

COMPUTING TERM FREQUENCY

COMPUTING AND DISPLAYING TF-IDF

```r
issue_tf_idf <- issue_words %>%
  bind_tf_idf(word, issue, n)

issue_tf_idf %>%
  arrange(desc(tf))
```

```
# A tibble: 574,385 × 6
   issue word      n    tf   idf tf_idf
   <int> <chr> <int> <dbl> <dbl>  <dbl>
 1 68302 the       8 0.170     0      0
 2 52204 the      31 0.150     0      0
 3 53256 the      28 0.146     0      0
 4 53191 the      58 0.146     0      0
 5 53078 the      27 0.141     0      0
 6 57761 the      18 0.132     0      0
 7 53284 the      58 0.130     0      0
 8 53077 the      26 0.13      0      0
 9 61094 the      49 0.130     0      0
10 53175 the      76 0.130     0      0
# ℹ 574,375 more rows
```

COMPUTING AND DISPLAYING TF-IDF

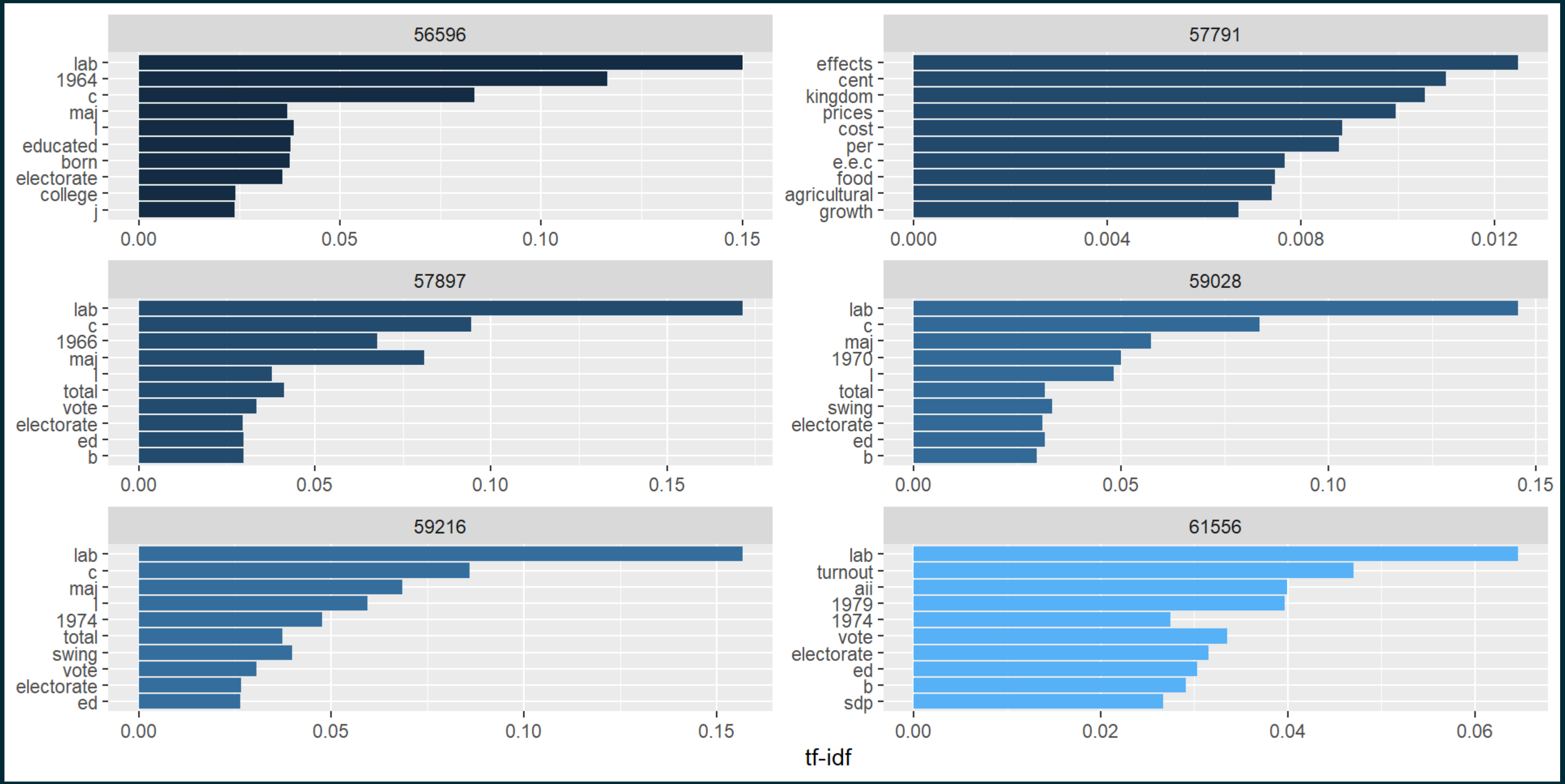
```r
issue_tf_idf %>%
  arrange(desc(tf_idf))
```

```
# A tibble: 574,385 × 6
   issue word           n     tf   idf tf_idf
   <int> <chr>      <int>  <dbl> <dbl>  <dbl>
 1 68732 cod           21 0.0463  7.17  0.332
 2 68277 bosnian        2 0.0435  7.17  0.312
 3 68405 code           2 0.0426  5.38  0.229
 4 68873 croatia        4 0.0317  7.17  0.228
 5 68873 rehn           4 0.0317  7.17  0.228
 6 55541 merlot         3 0.0283  7.17  0.203
 7 68578 flag           2 0.0455  4.40  0.200
 8 68890 ceausescu      2 0.0278  7.17  0.199
 9 68277 agents         2 0.0435  4.53  0.197
10 68302 wording        2 0.0426  4.61  0.196
# ℹ 574,375 more rows
```

COMPUTING AND DISPLAYING TF-IDF

```
issue_tf_idf %>%
  semi_join(first_6_unique_issues, by="issue") %>%
  group_by(issue) %>%
  slice_max(tf_idf, n = 10) %>%
  ungroup() %>%
  ggplot(aes(tf_idf, fct_reorder(word, tf_idf), fill = issue)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~issue, scales="free",ncol = 2) +
  labs(x = "tf-idf", y = NULL)
```

COMPUTING AND DISPLAYING TF-IDF



RELATIONSHIPS BETWEEN WORDS

- We can tokenize text so to obtain groups of n words or ngrams;
- An **ngram** is just a contiguous sequence of n items;
- In R ngrams are made using the tidytext function `unnest_tokens()`.

RELATIONSHIPS BETWEEN WORDS

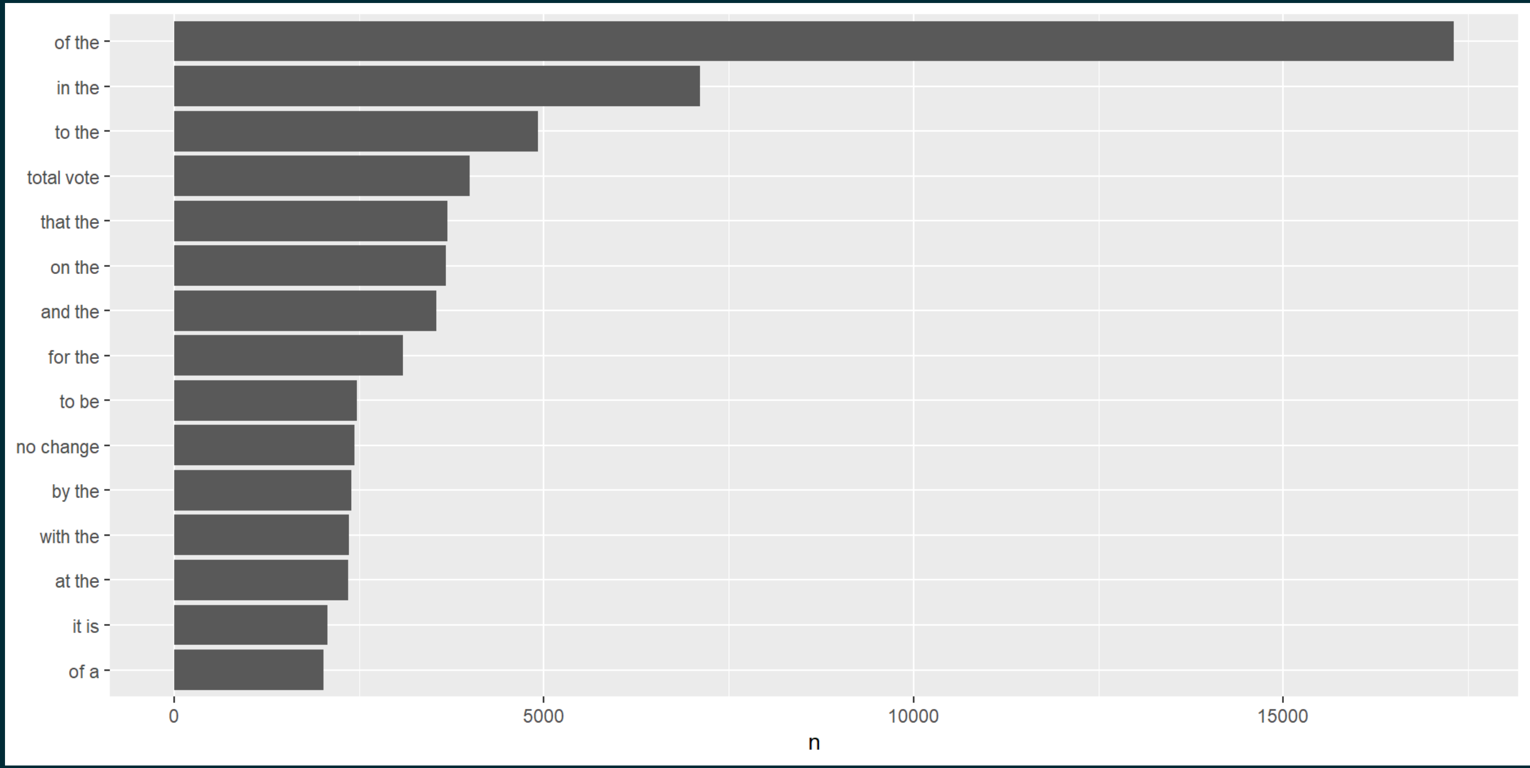
```r
tidy_content_rel <- data_df %>%
  unnest_tokens(bigram, content, token="ngrams", n=2)

tidy_content_rel
```

```
# A tibble: 1,548,046 × 9
   author             category `date-pub` edition issue query title volume
bigram
   <chr>              <chr>    <date>     <lgl>   <int> <chr> <chr> <lgl>
<chr>
 1 ['FROM A SPECIAL… ['News'] 1962-11-05 NA      55540 time… Euro… NA     from
a
 2 ['FROM A SPECIAL… ['News'] 1962-11-05 NA      55540 time… Euro… NA     a
spe…
 3 ['FROM A SPECIAL… ['News'] 1962-11-05 NA      55540 time… Euro… NA
speci…
 4 ['FROM A SPECIAL… ['News'] 1962-11-05 NA      55540 time… Euro… NA
corre…
 5 ['FROM A SPECIAL… ['News'] 1962-11-05 NA      55540 time… Euro… NA     at
th…
 6 ['FROM A SPECIAL… ['News'] 1962-11-05 NA      55540 time… Euro… NA     this
```

RELATIONSHIPS BETWEEN WORDS

```r
tidy_content_rel %>%
  count(bigram, sort = TRUE) %>%
  filter(n > 2000) %>%
  mutate(bigram = reorder(bigram, n)) %>%
  ggplot(aes(n, bigram)) +
  geom_col() +
  labs(y = NULL)
```



CLEANING UP BIAGRAMS

```
bigrams_separated <- tidy_content_rel %>%
  separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

tidy_content_rel_clean <- bigrams_filtered %>%
  unite(bigram, word1, word2, sep = " ")
```

PLOTTING BIAGRAMS

```
tidy_content_rel_clean %>%
  count(bigram, sort = TRUE) %>%
  filter(n > 500) %>%
  mutate(bigram = reorder(bigram, n)) %>%
  ggplot(aes(n, bigram)) +
  geom_col() +
  labs(y = NULL)
```