

HUMBOLDT–UNIVERSITÄT ZU BERLIN



LADISLAUS VON BORTKIEWICZ CHAIR OF STATISTICS  
C.A.S.E. – CENTER FOR APPLIED STATISTICS AND ECONOMICS

---

## Sentiment Analyses of European Countries

---

ANNA BOTHE, IVAN PEREZ, ROBERT SITNER

August 12, 2018

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Theory and Design</b>	<b>3</b>
2.1	Dictionaries . . . . .	3
2.1.1	Minqing-Bing Dictionary . . . . .	3
2.1.2	NRC Word-Emotion Association Lexicon . . . . .	3
2.1.3	Syuzhet Default Lexicon . . . . .	4
2.2	Mapping . . . . .	4
2.2.1	ggplot2 . . . . .	4
2.2.2	Shiny . . . . .	4
2.2.3	Shiny Dashboard . . . . .	4
2.3	Wordcloud . . . . .	5
<b>3</b>	<b>Input</b>	<b>5</b>
3.1	Data Loading . . . . .	5
3.2	Data Cleaning . . . . .	5
<b>4</b>	<b>Implementation</b>	<b>8</b>
4.1	Sentiment Analysis . . . . .	8
4.1.1	Minqing-Bing Dictionary . . . . .	8
4.1.2	NRC Word-Emotion Association Lexicon . . . . .	10
4.1.3	Syuzhet Default Lexicon . . . . .	11
4.2	Mapping . . . . .	12
4.2.1	Data Preparation for the Shiny Dashboard . . . . .	12
4.2.2	Shiny and Shiny Dashboard . . . . .	13
<b>5</b>	<b>Output</b>	<b>16</b>
5.1	Actual output . . . . .	16
5.2	Comments . . . . .	16
<b>6</b>	<b>Conclusion</b>	<b>17</b>
<b>A</b>	<b>Appendix</b>	<b>18</b>
A.1	Source Code . . . . .	18
A.2	Footnotes . . . . .	18
A.3	Declaration of Authorship . . . . .	19

# 1 Introduction

In general, sentiment analysis aims to identify the emotional tone of a text by using a variety of machine learning techniques. Due to the ambiguity of natural language, computers encounter many problems when processing text produced by humans. Natural Language Processing can serve as an interface, in order to enable computers to understand the correct content and sentiment of a text. With the use of particular dictionaries, a word appearing in a text can be mapped to a sentiment. The complexity of the mapping can vary. While some dictionaries just distinguish between positive and negative sentiments, others can categorize words in more complex sentiments like, for example, trust, joy or fear.

The applications are very broad. Many companies benefit of its use by extracting the sentiment of social media content about their products. Politicians and political candidates use sentiment analysis to gauge public opinion, in order to determine their campaign slogan or even to adjust their political contents to a broader mass of voters.

In our project, we analyze the country reports of all European countries, published by the European Commission<sup>1</sup>. Our goal is to identify the sentiment of the European Commission about the economic and political state of each European country for the years of 2012<sup>2</sup>, 2015<sup>3</sup> and 2018<sup>4</sup>. In our case, we focus on the Executive Summary, a short passage of text in the beginning of each report.

We use three different dictionaries with different mapping complexities, to gain a broader look on the possibilities of sentiment analysis.

We display the results on an interactive map, allowing the user to switch between the analyses of the Country Reports of the different publishing years and the dictionary used, resulting in a total of nine different views. In addition, we provide a word cloud for each country and for all three analyzed years, displaying the most frequent words used in the executive summaries.

All analyses are conducted using the latest version of R<sup>5</sup> and RStudio<sup>6</sup>.

## 2 Theory and Design

In order to conduct a sentiment analysis of a given text, first, a separate list of words or word combinations has to be created for every sentiment, which wants to be detected in the text. A combination of such lists is called a dictionary. The creation of a dictionary and therefore the categorization of a word into a sentiment is a task performed by human beings. Afterwards, an algorithm can determine whether the words of a text also appear in the dictionary. In order to receive the general sentiment of a text, all matches of a list have to be counted. Finally, the counts have either to be compared to each other or computed with each other. The sentiment with the most frequent occurrence or the highest score, represents the general sentiment of the text.

### 2.1 Dictionaries

With the use of multiple dictionaries, we gain a broader understanding of the possibilities of sentiment analysis. In the following passages, we introduce the dictionaries we use.

#### 2.1.1 Minqing-Bing Dictionary

Minqing Hu and Bing Liu created the Minqing-Bing Dictionary<sup>7</sup>, which consists of two lists, containing positive opinion words and negative opinion words respectively. The list of positive words contains about 2,000 words, while the list of negative words contains about 4,700 words. Both lists also contain misspelled versions of words, since it was initially designed to analyze social media content.

#### 2.1.2 NRC Word-Emotion Association Lexicon

The Word-Emotion Association Lexicon<sup>8</sup>, also known as EmoLex, was created by Saif Mohammad and Peter Turney of the National Research Council Canada (NRC). It contains about 14,000 words, roughly 3,300 being negative sentiment words and about 2,300 being positive sentiment words. The rest of the words can be categorized in eight emotions: anger, anticipation, disgust, fear, joy, sadness, surprise and trust. The following graphic visualizes the sentiment and emotion distribution. Please visit Mohammad's website<sup>9</sup> for an interactive version.



Figure 1: NRC Dictionary, distribution of sentiments and emotions

### 2.1.3 Syuzhet Default Lexicon

The Syuzhet Default Lexicon<sup>10</sup> was developed by Matthew L. Jockers with the support of the Nebraska Literary Lab. The dictionary consist of 165,000 human coded sentences taken from contemporary novels, which makes it stronger in describing the sentiment of whole sentences. Further, it categorizes the words or sentences in five sentiments: positive, very positive, negative, very negative and neutral. Unfortunately, it isn't possible to inspect the dictionary's contents. However, there are multiple validation experiments conducted by Jockers and his team, which are available on his blog<sup>11</sup>.

## 2.2 Mapping

We display the results of the sentiment analyses on a map, in order to achieve a visual comparison of the sentiments of the different European countries. In the following sections, we describe the tools we use for the implementation.

### 2.2.1 ggplot2

ggplot2<sup>12</sup> is a broadly used R-package, allowing its user to create graphs, plots and other different forms of visual data presentation. In our case, we choose to display our results by providing a map of Europe. The different sentiments and emotions are characterized by colors. The countries are dyed in the color of the sentiment or emotion, which associates the strongest with the country's executive summary. With the use of a legend, we describe which color belongs to which sentiment or emotion.

### 2.2.2 Shiny

The R-package shiny<sup>13</sup> allows to build interactive web applications. Via renderggigraph we create a reactive version of the shiny object. This allows us adding certain functionality to the map:

- mousing over a country displays the country's name
- clicking on a country opens the country's Country Report

Examining the actual Country Report the sentiment analysis derived from, allows a convenient way of human validation of our results.

### 2.2.3 Shiny Dashboard

In order to allow the user to choose between the different publishing years of the Country Reports and the dictionaries we use, we implement an R-package called shinydashboard<sup>14</sup>. Thus, a user can select the required settings on the left hand side of the dashboard.

## 2.3 Wordcloud

In addition to the possibility of examining the whole text of the executive summary of the Country Reports, we provide a wordcloud, showing the most frequently occurring words in an executive summary. This is realized by using the R-package `wordcloud2`<sup>15</sup>.

## 3 Input

As mentioned before, our input data consists of the country reports of the European Commission for the years 2012<sup>2</sup>, 2015<sup>3</sup> and 2018<sup>4</sup>. Specifically the executive summary, which is a small text passage at the beginning of each report. The following examples and code demonstrations are for the reports of the year 2012, but do not differ for the reports of 2015 or 2018 in any way.

### 3.1 Data Loading

In order to receive the data, we download the PDF files containing the Country Reports from the European Commission website<sup>1</sup> and save those as `.txt`-files. We perform this task manually and without the help of an algorithm. In the following step, we create three lists for the three different years, each containing the names of all countries, for which a report was prepared for in the respective year. Next, we create a new list and iterate over the country name list, inserting the name of the countries and their executive summaries in the new list, using a `for`-loop and the function `readLines`, which is included in the *base* R-package<sup>16</sup>.

A `for`-loop repeats a certain passage of code any given amount of times, in this case for every country in the list. `readLines` extracts the content of a `.txt`-file by providing the function with the name of the file. The names of the `.txt`-files follow the structure of: "nameOfCountry\_" + year + ".txt". With the usage of the `paste0`-function, which is also included in the *base* R-package<sup>16</sup>, we concatenate multiple strings into one and provide the `readLines`-function with the file-name.

---

```
38 for (i in 1:length(countries_2012)){
39   print(i)
40   list_2012[[countries_2012[i]]] <- paste(readLines(
      paste0(countries_2012[i], "_", "2012", ".txt")), collapse = " ")
41 }
```

---

### 3.2 Data Cleaning

In order to conduct a sentiment analysis, we need to pro-process the executive summaries. This step is also known as data cleaning. In detail, we

- lowercase every word, so that for example "Market" equals "market",
- remove so-called stop words: words which are adding little to no semantic value, for example "the" or "at"

- remove single character words, numbers, punctuation and the country's name of it's own executive summary.

In order to achieve this, we use a variety of functions:

- *gsub*: Substitutes existing strings in a text.
  - included in the *base* R-package<sup>16</sup>
  - input "pattern": a regular expression syntax (VERWEIS)
  - input "replacement": replacement for the matched pattern
  - input "x": the text
- *tolower*: Lowercases a given text.
  - included in the *base* R-package<sup>16</sup>
  - input "x": the text
- *removeWords*: Removes words from a given text.
  - included in the *tm* R-package<sup>17</sup>
  - input "x": the text
  - input "words": words to remove
- *stripWhitespace*: Collapses multiple whitespaces to a single blank.
  - included in the *tm* R-package<sup>17</sup>
  - input "x": the text

Afterwards, we subdivide the executive summaries into separate words, creating so-called bag of words. In order to achieve this, we use the function *strsplit* included in the *base* R-package<sup>16</sup>. It takes a text and the pattern by which the text should be split as inputs and gives the text separated as single words as the output.

While some sentiment analyses require a bag of words, others are able to analyze the non-separated text, retaining additional semantic value. Thus, we save both options of every pre-processed executive summary: the bag of words and the non-separated text. In the following list, we show how we implement the data cleaning.

- We remove Not Words (e.g. '.' and '\$').

---

```
75 list_2012_cleaned[[i]]
   <- gsub(pattern = "\\W", replace=" ", list_2012[[i]])
```

---

- We remove digits (e.g. '89' and '2017').

---

```
77 list_2012_cleaned[[i]]
   <- gsub(pattern = "\\d", replace=" ", list_2012_cleaned[[i]])
```

---

- We lowercase all letters. (e.g. 'Market'→'market').

---

```
79 list_2012_cleaned[[i]]  
  <- tolower(list_2012_cleaned[[i]])
```

---

- We remove stop words (e.g. 'the' and 'at').

---

```
81 list_2012_cleaned[[i]]  
  <- removeWords(list_2012_cleaned[[i]], stopwords())
```

---

- We remove the country's name of it's own executive summary.

---

```
83 list_2012_cleaned[[i]]  
  <- removeWords(list_2012_cleaned[[i]], names(list_2012)[i])
```

---

- We remove the word "also" as an additional stop word.

---

```
85 list_2012_cleaned[[i]]  
  <- removeWords(list_2012_cleaned[[i]], "also")
```

---

- We remove single letter words (e.g. 'a' and 's').

---

```
87 list_2012_cleaned[[i]]  
  <- gsub(pattern = "\\b[a-z]\\b{1}", replace=" ",  
    list_2012_cleaned[[i]])
```

---

- We replace multiple white spaces between words with a single white space.

---

```
89 list_2018_cleaned[[i]]  
  <- stripWhitespace(list_2018_cleaned[[i]])
```

---

- We transform the text into a bag of words.

---

```
91 list_2018_cleaned[[i]]  
  <- strsplit(list_2018_cleaned[[i]], "\\s+")[1]
```

---



In the following figure, we provide an example of (a) a list of non-separated texts and (b) a list of bag of words.

list_2012_cleaned_full	list [27]	List of length 27
austria	character [1]	' executive summary , austrian economy surpassed pre-crisis output employment le ...
belgium	character [1]	' executive summary , \' gdp expected remain broadly flat first half year, follo ...
bulgaria	character [1]	'executive summary real gdp growth expected remain rather low first half , accel ...
cyprus	character [1]	'executive summary , \' economic activity expected contract . % , regaining mome ...
czech republic	character [1]	' executive summary real gdp growth expected stagnate , pick . % . unemployment ...
denmark	character [1]	'executive summary , danish economy expected pick slightly compared , foreseen g ...
estonia	character [1]	'executive summary strong rebound , \' gdp growth expected slow . % . rate unempl ...
finland	character [1]	'executive summary , \' economic activity expected grow . % . % . unemployment f ...
france	character [1]	'executive summary , \' economic activity expected grow . % , regaining momentum ...
germany	character [1]	'executive summary , \' economic activity expected significantly slow compared . ...

(a) Cleaned full text list

list_2012_cleaned	list [27]	List of length 27
austria	character [190]	" 'executive' 'summary' 'austrian' 'economy' 'surpassed' ...
belgium	character [152]	" 'executive' 'summary' 'gdp' 'expected' 'remain' ...
bulgaria	character [103]	'executive' 'summary' 'real' 'gdp' 'growth' 'expected' ...
cyprus	character [151]	'executive' 'summary' 'economic' 'activity' 'expected' 'contract' ...
czech republic	character [134]	" 'executive' 'summary' 'real' 'gdp' 'growth' ...
denmark	character [107]	'executive' 'summary' 'danish' 'economy' 'expected' 'pick' ...
estonia	character [162]	'executive' 'summary' 'strong' 'rebound' 'gdp' 'growth' ...
finland	character [117]	'executive' 'summary' 'economic' 'activity' 'expected' 'grow' ...
france	character [124]	'executive' 'summary' 'economic' 'activity' 'expected' 'grow' ...
germany	character [131]	'executive' 'summary' 'economic' 'activity' 'expected' 'significantly' ...

(b) Cleaned *bag of words* list

Figure 2: Example of data sets after data cleaning

## 4 Implementation

In this section we focus on the implementation of our sentiment analyses, the shiny dashboard, including the maps of European countries and the wordclouds.

### 4.1 Sentiment Analysis

In the following passages, we describe how we performed the separate sentiment analyses, given the different dictionaries.

#### 4.1.1 Minqing-Bing Dictionary

Minqing-Bing Dictionary<sup>7</sup> consists of two list, one containing positive opinion words and one containing negative opinion words. Firstly, we import both text files and separate both texts into a list of single words respectively, creating a bag of words.

---

```

264 # read positive words
265 poswords <- paste(readLines("poswords.txt"), collapse = " ")
266 # Divide poswords text into separate words:
267 poswords <- strsplit(poswords, "\\s+")[[1]]

269 # read negative words
270 negwords <- paste(readLines("negwords.txt"), collapse = " ")
271 # Divide negwords text into separate words:
272 negwords <- strsplit(negwords, "\\s+")[[1]]

```

---

Afterwards, we scan through the bag of words of every executive summary and check whether a word appears in the dictionary. Therefore, we create two vectors with the length of the size of the bag of words of every summary. One vector for positive matches and one vector for negative matches. If a word of the summary is listed in the list of positive words, the value of the position of the word in the vector of positive matches is set to *true*. We repeat the same procedure for the list of negative words. Finally, we count every *true* value of the vector of the positives matches and subtract it by the count of every *true* value of the vector of the negative matches.

---

```

275 for (i in 1:length(list_2012_cleaned)){
280   score <- sum(!is.na(match(list_2012_cleaned[[i]], poswords)))
281   -sum(!is.na(match(list_2012_cleaned[[i]], negwords)))
288 }

```

---

This value represents the sentiment score:

$$sentiment\ score = \sum pos.\ words - \sum neg.\ words$$

A sentiment score  $> 0$  indicates an overall positive attitude towards the country's economy. The higher the score, the more positive is the attitude towards a country. The same applies for the opposite case. A sentiment score  $= 0$  represents a neutral sentiment. We save the sentiment score of every country in the list of the respective year. The final list has the following structure (score = column "List of length 27"):

minqing_2018	list [27]	List of length 27
austria	integer [1]	113
belgium	integer [1]	84
bulgaria	integer [1]	98
croatia	integer [1]	69
cyprus	integer [1]	92
czech republic	integer [1]	45
denmark	integer [1]	71
estonia	integer [1]	65
finland	integer [1]	78
france	integer [1]	117
germany	integer [1]	74

Figure 3: Example of Minging-Bing Dictionary Analysis Results

#### 4.1.2 NRC Word-Emotion Association Lexicon

As mentioned before, in addition to a list of positive and a list of negative words, this dictionary<sup>8</sup> also includes lists for the following emotions: anger, anticipation, disgust, fear, joy, sadness, surprise and trust. Because we already examined the executive summaries for positive and a negative sentiments with the use of the Minging-Bing Dictionary, we exclude these sentiments from this analysis.

For the sentiment analysis using the NRC Word-Emotion Association Lexicon<sup>8</sup>, we use the *get\_nrc\_sentiment* function included in the *syuzhet* R-package<sup>18</sup>.

The function takes a bag of words as the input creates a data frame, which displays whether a word of the executive summary (rows) can be matched to a any list of the eight emotions (columns). If so, this value is set to *1*, otherwise the value is set to *0*. We provide an example of a data frame in the following graphic:

	anger	anticipation	disgust	fear	joy	sadness	surprise	trust	negative	positive
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	1
9	0	1	0	0	1	0	0	1	0	1
10	0	0	0	0	0	0	0	0	0	0

Figure 4: Example of a *get\_nrc\_sentiment* data frame showing the first ten words

Afterwards, we find the most frequently occurring emotion for a executive summary by calculating the column sum of each data frame. Columns 9 and 10 are excluded, because they contain the matches for positive and the negative sentiments.

---

```

213 NRC_2012 = list()
220 NRC_2012[[i]] <- colnames(NRC[,c(-9,-10)])
    [which.max(colSums(NRC[,c(-9,-10)]))]

```

---

These sums are saved in a lists, one list for each year. We provide an example for a result list with the following graphic.

🔻 NRC_2012	list [27]	List of length 27
austria	character [1]	'anticipation'
belgium	character [1]	'trust'
bulgaria	character [1]	'trust'
cyprus	character [1]	'trust'
czech republic	character [1]	'anticipation'
denmark	character [1]	'trust'
estonia	character [1]	'anticipation'
finland	character [1]	'anticipation'
france	character [1]	'anticipation'
germany	character [1]	'trust'

Figure 5: Example of a NRC Word-Emotion Association Lexicon Analysis result

### 4.1.3 Syuzhet Default Lexicon

In contrast to the both previous dictionaries, this dictionary<sup>10</sup> consists of text passages, instead of words. Therefore, we use the *sentiment*-function, included in the R-package *sentimentr*<sup>19</sup>, since it is able to determine the sentiment of a text at the sentence level. This yields the advantage of taking valance shifters into account. Valance shifters are words which change or modify the sentiment of a text, i.e. negators and amplifiers. A negator inverts the sentiment of a word. For example: The word "not" in the sentence "I do not like it", changes the sentiment of the sentence to negative, although it would be positive without it. An amplifier increases the impact of a word. For example: The word "really" in the sentence "I really like it.", amplifies the positive sentiment of the text. Creating a bag of words prevents such analysis, since the words are taken out of context and words like "not" are removed as stop words. So, we provide the *sentiment*-function with a cleaned full text list, instead of a bag of words. Then, we determine the sentiment of every sentence and obtain the general sentiment of each text by obtaining the average sentiment of the sentences for each text respectively.

---

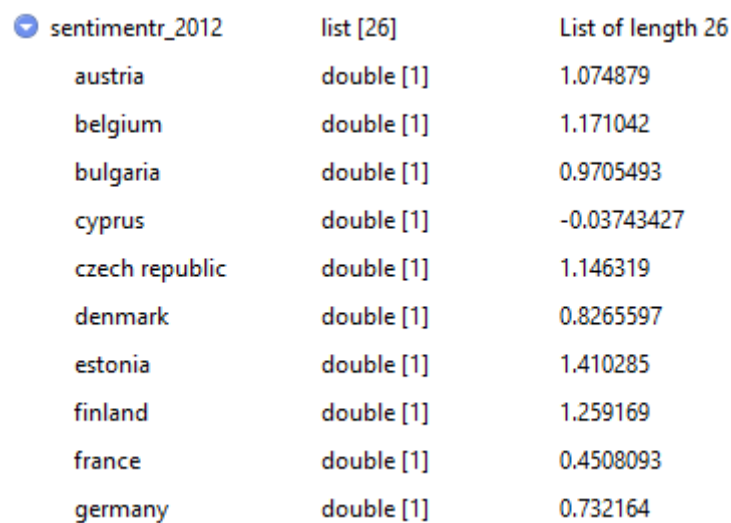
```

360 for (i in 1:length(list_2012_cleaned_full)){
361   # Apply sentimentr to every sentence of the executive summary:
362   sent <- sentimentr::sentiment(get_sentences
      (list_2012_cleaned_full[[i]]))$sentiment
363   # Compute the mean of the polarity and assing it to the new list:
364   sentimentr_2012[[i]] <- mean(sent)
365   # Use the corresponding country name:
366   names(sentimentr_2012)[i] <- names(list_2012)[i]

```

---

These averages are saved in a lists, one list for each year. We provide an example for a result list with the following graphic.



sentimentr_2012	list [26]	List of length 26
austria	double [1]	1.074879
belgium	double [1]	1.171042
bulgaria	double [1]	0.9705493
cyprus	double [1]	-0.03743427
czech republic	double [1]	1.146319
denmark	double [1]	0.8265597
estonia	double [1]	1.410285
finland	double [1]	1.259169
france	double [1]	0.4508093
germany	double [1]	0.732164

Figure 6: Example of a Syuzhet Default Lexicon Analysis result

## 4.2 Mapping

In the following passages we describe how we produce the different maps and word clouds of the European countries, in order to display them in a shiny dashboard.

### 4.2.1 Data Preparation for the Shiny Dashboard

In order to produce the different maps of the European countries, we create a data frame containing the mapping between the countries and the results of the different sentiment analyses. Therefore, we use the function *map\_data*, which is included in the R-package *ggplot2*<sup>12</sup>, and pass *world* as the input argument. The output is a data frame containing country names in combination with their latitude and longitude coordinates. Additionally, we remove all countries from the data frame *global* which are not part of Europe, with the use of the function *which*, which is included in the R-package *base*<sup>16</sup>.

---

```

470 ind_eur <- which(global$region %in% european_countries)
471 global <- global[ind_eur,]

```

---

In the next step, we create separate data frames for the results of the sentiment analyses for every dictionary and year. We use the *unlist*-function included in the R-package *base*<sup>16</sup>, in order to create vectors out of the lists containing the sentiment analyses results. Afterwards, we create data frames out of the vectors.

---

```
413 NRC_2012_df <- data.frame(unlist(NRC_2012))
428 minqing_2012_df <- data.frame(unlist(minqing_2012))
443 sentimentr_2012_df <- data.frame(unlist(sentimentr_2012))
```

---

Then, we use the function *left\_join* included in the R-package *dplyr*<sup>20</sup>, which allows using join functions known from SQL. We join the *global* data frame and the result data frames by the column *region*, in order to receive a data frame containing all European countries and their results produced by the sentiment analyses.

---

```
479 global <- left_join(global, NRC_2012_df, by = "region")
482 global <- left_join(global, minqing_2012_df, by = "region")
485 global <- left_join(global, sentimentr_2012_df, by = "region")
```

---

#### 4.2.2 Shiny and Shiny Dashboard

Shiny applications consist of two components, a user interface object and a server function. When the function *runApp* of the R-package *shiny*<sup>13</sup> is executed, it searches for the *ui.R*-file, containing the user interface, and the *server.R*-file, containing the server function, in the current working directory.

**server.R-file** The *server.R*-file contains the user defined server function *function*, which generates the different maps, dependent on the year and dictionary, and the different word clouds, dependent on the year and country. The dependencies year, dictionary and country are selected in the user interface, which we describe in more detail later. The server function takes a combination of dependencies (e.g. year = '2012', dictionary = 'NRC', country = 'austria') as inputs and gives the corresponding map and word cloud as the output, after producing them. We assign the map output and the word cloud output as so-called slots to the *output* object.

**Map Slot** The map slot is a *ggplot* object, which we produce with the help of the *ggplot*<sup>12</sup> function. Additionally, we use the function *geom\_polygon\_interactive*, which is included in the R-package *ggiraph*<sup>21</sup>, in order to make the *ggplot* object interactive. Now, by clicking on a country on the map, the PDF file of the respective country report opens in the browser. The *ggplot* object is wrapped by the function *renderggiraph*, also included in the R-package *ggiraph*<sup>21</sup>. It adjusts the output, every time the input changes, enabling the interactiveness of the web application.

**Word Cloud Slot** The word cloud slot is a *wordcloud2* object, which we produce with the same-named function, included in the same-named R-package<sup>15</sup>. The function *wordcloud2* takes a data frame with the words and their respective count as input. We generate the data frame by providing the *count*-function, included in the R-package *plyr*<sup>23</sup>, with the bag of words of the respective country of the respective year. Afterwards, we adjust each data frame by including only words with a word count higher than or equal a defined threshold. We set the thresholds as follows:

Threshold for country reports published in ...

- 2012: 1
- 2015: 2
- 2018: 3

Due to the fact, that the amount of words in a executive summary increased with each publishing year of a country report, the word clouds showed an increasing number of words and therefore became confusingly complex as a result. In order to prevent this, we implement the already mentioned thresholds.

## Server Function Code

---

```
6 shinyServer(function(input, output){
  (...)
16 output$map <- renderggiraph({
17
18   if(input$year == "2012" & input$dictionary == "NRC"){
19
20     global$oc2 <- sprintf(
21       "window.open(\"%s%s%s\")",
22       "https://ec.europa.eu/info/sites/info/files/file_import
        /swd2012_",
23       as.character(global$region), "_en_0.pdf"
24     )
25
26     Sentiment <- factor(global$NRC_2012, levels =
        c("trust", "anticipation"))
27
28     gg2 <- ggplot(data = global, aes(x=long, y = lat)) +
29       geom_polygon_interactive(aes(fill = Sentiment,
30                                   group = group,
31                                   tooltip = region, data_id = region,
32                                   onclick = oc2), colour = "black") +
33       coord_fixed(1.3) + xlim(-12,42) + ylim(35,73) +
34       labs(x = "", y = "", title = "Europe Sentiment Analysis",
        subtitle = "Sentiment Analysis on European Commission
        Annual Report") +
```

```

35     theme_bw() +
36     theme(panel.grid = element_blank()) +
37     theme(strip.text = element_blank()) +
38     # theme(legend.title=element_text("Sentiment")) +
39     theme(title = element_text(face = "bold")) +
40     scale_fill_discrete(na.value="grey")
41
42   }
43
44   else if
45   (...)
256   output$wordcloud <- renderWordcloud2({
257
258     # Bigger size wordcloud 2012
259     if (input$year == "2012" & input$country %in% c("czech republic",
260                                                     "france",
261                                                     "germany",
262                                                     "italy",
263                                                     "netherland",
264                                                     "portugal",
265                                                     "slovakia",
266                                                     "uk")) {
267       country_df <- data.frame(unlist
268                               (list_2012_cleaned[[input$country]]))
269       colnames(country_df) <- c("words")
270       words_df <- plyr::count(country_df, "words")
271       words_df <- words_df[order(words_df$freq, decreasing = T),]
272       words_df <- words_df[words_df$freq >= 1,]
273       wordcloud2::wordcloud2(words_df, size = 0.3)
274   }
275   (...)

```

---

**ui.R-file** The *ui.R*-file specifies the general structure of the shiny dashboard. With the use of the *shinyUI*-function, included in the R-package *shiny*<sup>13</sup>, it is possible to define the individual parts of the structure. With the function *dashboardPage*, included in the R-package *shinydashboard*footnotemark[14], we create a page and define its structure by passing further functions as arguments to the *dashboardPage*-function. We provide the choice between the years, dictionaries and countries with the use of the *selectInput*-function. Also, we pass the maps and the word clouds, generated by the *function*, to the *box*-function, in order to display them in the web application.



## User Interface Code

```

3 shinyUI(
4   dashboardPage(
5     dashboardHeader(title = "SPL SS2018"),
6     dashboardSidebar(
7       selectInput("year", "Select Year",
8                   choices = c("2012", "2015", "2018")),
9       selectInput("dictionary", "Select dictionary",
10                  choices = c("NRC", "Minqing", "Sentimentr")),
11       selectInput("country", "Select country for wordcloud",
12                  choices = c("austria", "belgium", "bulgaria", "...")),
13     ),
14     dashboardBody(
15       fluidRow(
16         box(ggiraphOutput("map"), align = "center"),
17         box(wordcloud2Output("wordcloud"), align = "center")
18       )
19     )
20 )

```

## 5 Output

### 5.1 Actual output

An example view of the shiny dashboard is shown in the following graphic.

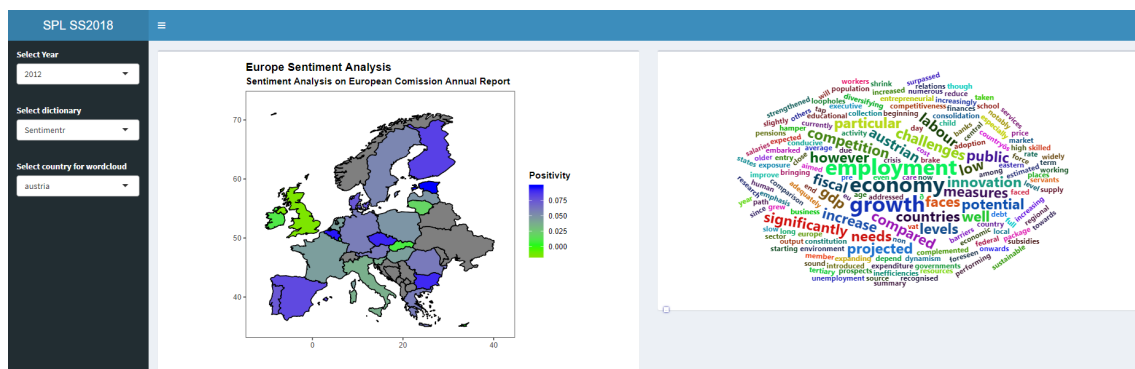


Figure 7: Example view of the shiny dashboard

## 5.2 Comments

The output is less contrary than we initially expected it to be. The European country reports are all relatively positive. This is mainly due to the overall stability of the economic environment of Europe. Also, the time period of our data sources is fairly small. The oldest European country reports are published in 2012 - several

years after the last global economic crisis.

For example, the NRC sentiment analysis shows the sentiment trust for all countries in 2018. But also in 2012 and 2015, there is not much variance. Also the Mincing-Bing dictionary matches only with a small number of words. For example in the case of the country report of Austria published in the year 2018, the matched number of words were only about 14%. Out of the 1,173 given words, 113 were positive and 58 negative; the rest was not evaluated at all. Even the sentiment analysis conducted with the Syzhet Default Lexicon, which analyzes the sentiment at sentence level, didn't yield much improvement.

## 6 Conclusion

As described in the introduction, natural language processing is quite challenging, since the structure of human communication is very complex. Despite the use of different dictionaries, we could not produce significant results. In our chase, another challenge was the too factual tone of the executive summaries. For further analysis, it would be very interesting to have a look at the sentiment changes within a country over the bigger time period, in order to include the economic crisis in the late 2000s.

# A Appendix

## A.1 Source Code

We provide the source code in GitHub:

<https://github.com/ivanchoperez/SPL-SS2018>

## A.2 Footnotes

1. Website of the European Commission
2. European Commission Country Reports of 2012
3. European Commission Country Reports of 2015
4. European Commission Country Reports of 2018
5. The R Project homepage
6. RStudio homepage
7. Minqing-Bing Dictionary download link
8. NRC Dictionary download link
9. NRC Dictionary, interactive table of sentiment and emotion distribution
10. Syuzhet Default Lexicon description
11. Matthew Jockers' blog
12. ggplot2 R-package download link
13. shiny R-package download link
14. shinydashboard R-package download link
15. worldcloud2 R-package download link
16. base R-package documentation link
17. tm R-package download link
18. syuzhet R-package documentation link
19. sentimentr R-package documentation link
20. dplyr R-package download link
21. ggiraph R-package download link
22. plyr R-package download link

### **A.3 Declaration of Authorship**

We hereby confirm that we have authored this Seminar paper independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such.

Berlin, August 12, 2018,  
Anna Bothe, Ivan Perez, Robert Sitner