R in literary analysis in English

Diego Giménez

R codes used for distant reading analysis in English (Giménez, 2024). This document updates and expands a previous version developed by Diego Giménez and Andresa Gomide in 2022, which focused on the analysis of “The Book of Disquiet”. In this new version, works by James Joyce are analyzed, including “Ulysses”, “Dubliners” and “A Portrait of the Artist as a Young Man”.

1 Tools and Data Preparation

1.1 Installation

Quanteda (Quantitative Analysis of Textual Data) is an R package for the manipulation and analysis of textual data.

The installation of R varies according to the operating system (e.g., Windows, Mac, Linux), as well as its different versions. There are several sources where you can get updated instructions on how to install R (e.g., https://didatica.tech/como-instalar-a-linguagem-r-e-o-rstudio/). The Comprehensive R Archive Network (CRAN), the official distribution network of R, offers reliable instructions for this, although perhaps not as detailed as in other sources.

Another suggestion is to install a Graphical User Interface (GUI). GUIs considerably facilitate user interaction with the computer. RStudio is the most used GUI for R and, like R, it is free and open-source.

1.2 Configuration: preparing the environment

When reusing codes, it is good practice to be aware of the installed version of both R and the libraries used. It is not necessary for the versions to be the same as those used during the creation of the codes; however, in some cases, there may be incompatibility between different versions, and some functions or packages may have been discontinued. This article was written using version 4.3.3 of R.

```
# Check the R version

R.version.string
```

```
## [1] "R version 4.3.3 (2024-02-29 ucrt)"
```

For our analysis, we will use some existing packages. These packages are nothing more than extensions for R that usually contain data or codes. To use them, we need to install them on the computer, if it has not already been done, and load them into R. One advantage of loading only the necessary packages (rather than all installed packages) is to avoid unnecessary computational processing. The code below creates a list of the packages used in the present analysis and loads them, installing those that were not present.

```
# We list the packages we need

packages = c("quanteda", # Quantitative analysis of textual data.
             "quanteda.textmodels", # Complements Quanteda, providing specific functionalities
for text modeling.
             "quanteda.textstats", # This package contains functions to calculate descriptive s
tatistics and measures of text complexity, such as lexical diversity and lexical density.
             "quanteda.textplots", # This package offers tools for visualizing textual data, in
cluding word scatter plots, word clouds, and heatmaps.
             "newsmap", # For document classification, based on "seed words," i.e., predefined
keywords indicating topics or categories.
             "readtext", # For reading different text formats.
             "spacyr", # For grammatical class annotation, entity recognition, and syntactic an
```

```
notation (Python must be installed).
             "ggplot2", # For simple frequency graphs.
             "seededlda", # For topic modeling.
             "stringr", # For regular expressions.
             "dplyr",  # This package is part of the tidyverse and offers a set of functions fo
r tabular data manipulation in R, allowing operations like filtering, selection, aggregation, a
nd data merging easily and efficiently.
             "tidytext", # This package complements tidyverse, providing tools for text analysi
s along with tidyverse's data organization principles, allowing easy integration of text analys
is into data analysis pipelines.
             "knitr", # This package is used for producing dynamic documents in R, allowing int
egration of R code and analysis results into Markdown, HTML, PDF, and other formats.
             "stringr", #This package provides functions for text string manipulation in R, fac
ilitating tasks such as pattern matching, substring extraction, and text manipulation.
             "igraph", # This package is used for network analysis and visualization in R, offe
ring functions for creating, manipulating, and representing graphs and complex networks.
             "topicmodels" # This package is used for topic modeling in texts, offering impleme
ntations of algorithms like LDA (Latent Dirichlet Allocation) and LSA (Latent Semantic Analysi
s) for topic inference in collections of documents.
             )

# We install (if necessary) and load the packages.
package.check <- lapply(
  packages,
  FUN = function(x) {
    if (!require(x, character.only = TRUE)) {
      install.packages(x, dependencies = TRUE)
      require(x, character.only = TRUE)
    }
  }
)
```

The codes below were implemented in version 4.0.2 of Quanteda. Using a different version may result in errors or undesired results. To check the version of the packages, we use the `packageVersion` function. To check the version of R, we use `R.version.string` .

```
# Check the version of Quanteda.

packageVersion("quanteda")
```

```
## [1] '4.0.2'
```

Finally, we need to establish what our working directory will be. This will be the location where the results will be saved. To identify what the current working directory is, we use the getwd() function, which returns the absolute path, i.e., the complete address of the directory. To set a new working location, we use the setwd() function. Files saved in this directory can be read by simply indicating the file name because we can use the relative path, i.e., the file address from the directory we are working in.

1.3 Data

Once the necessary packages are installed, we can proceed with the analysis of the corpus. For this, we need to load the corpus into R. If we are working with data stored locally, that is, available on the computer where the analyzes will be performed, simply use the readtext() function, indicating the location (relative or absolute) of the desired file.

The book ‘Ulysses’ can be read as a single file,

```
# To read a single file with all the content of the book

ulysses <- readtext("~/corpora/Ulysses.txt", encoding = "utf-8")

# Returns the structure of the created object

str(ulysses)
```

```
## Classes 'readtext' and 'data.frame': 1 obs. of  2 variables:
##  $ doc_id: chr "Ulysses.txt"
##  $ text  : chr "Ulysses\n\n\nby James Joyce\n\n\nContents\n\n — I —\n\n [ 1 ]\n [ 2 ]\n [ 3
]\n\n — II —\n\n [ 4 ]\n [ 5 ]\n [ "| __truncated__
```

Or considering the book as a unit within a corpus formed by several documents:

```
# Read all files in the ldod folder of the corpora directory

joyce_files <- readtext("~/corpora/joyce", encoding = "utf-8")

# Returns the structure of the created object

str(joyce_files)
```

```
## Classes 'readtext' and 'data.frame': 4 obs. of  2 variables:
##  $ doc_id: chr  "A Portrait of the Artist as a Young Man.txt" "Dubliners.txt" "PPn25_01.pdf"
"Ulysses.txt"
##  $ text  : chr  "A Portrait of the Artist as a Young Man\n\nby James Joyce\n\n\nContents\n\n
Chapter I\n Chapter II\n Chapter II"| __truncated__ "Dubliners\n\nby James Joyce\n\n\nContents
\n\n The Sisters\n An Encounter\n Araby\n Eveline\n After the Race\n T"| __truncated__ "
n. 25\n                                                    "| __truncated__ "Ulysses\n\n\nby James
Joyce\n\n\nContents\n\n — I —\n\n [ 1 ]\n [ 2 ]\n [ 3 ]\n\n — II —\n\n [ 4 ]\n [ 5 ]\n [ "| __t
runcated__
```

The texts above derive from the work by James Joyce, available in Project Gutenberg.

The files were saved with utf-8 encoding, and pre-textual and editorial information (such as editors' notes) that could interfere with the software's automatic search were eliminated.

The analyzes below will be demonstrated using the two corpora, at different times.

1.3.1 Cleaning

The cleaning below was applied only to the texts saved separately (´joyce_files). The file with the book in a single text (Ulysses`) had already been cleaned previously.

```
# We create a copy to recover the original in case there are errors in the regex

joyce_clean <- joyce_files

## Removal of unwanted elements


# Remove numbers at the beginning of lines (indices)

joyce_clean$text <- str_replace_all(joyce_clean$text, "\\n\\d", "\n")
```

```
# Remove dates

joyce_clean$text <- str_replace_all(joyce_clean$text, "\\d{1,2}-(\\d{1,2}|[IVX]{1,4})-19\\d
{2}", "")
```

1.4 Research with Quanteda

After the files are loaded into the system, we need to create a “corpus” object, that is, the format necessary for Quanteda to process and generate information about the text(s). To do this, just apply the ‘corpus’ function. Automatically, the text is segmented into tokens and sentences. Tokens correspond to all occurrences (including repetitions) of words, as well as other items such as punctuation, numbers, and symbols. When we investigate the corpus with the ‘summary’ function, we obtain the count of sentences, tokens, and types (the number of distinct tokens in a corpus).

```
# Create the corpus from multiple files

corpus_clean <- corpus(joyce_clean)

# Present a summary of the corpus

summary(corpus_clean)
```

	Text <chr>	Typ… <int>	Toke… <int>	Sentences <int>
1	A Portrait of the Artist as a Young Man.txt	9944	96592	4601
2	Dubliners.txt	8165	79650	4736
3	PPn25_01.pdf	589	3367	41
4	Ulysses.txt	35108	318968	22729

4 rows

```
# Create a corpus from the single file

corpus_unico <- corpus(ulysses)

# Present a summary of the corpus

summary(corpus_unico)
```

	Text <chr>	Types <int>	Tokens <int>	Sentences <int>
1	Ulysses.txt	35104	318975	22730

1 row

If necessary, we can change the structure of our corpus. In ‘corpus_unico’, we have a corpus made with only one text. With ‘corpus_reshape’, we can create a new corpus where each sentence is considered a text, that is, a unit.

```
# Reveal the number of texts in the corpus

ndoc(corpus_unico)
```

```
## [1] 1
```

```
# Restructure the corpus, converting each sentence into a unit

corpus_sents <- corpus_reshape(corpus_unico, to = "sentences")

# Present a summary of the corpus

summary(corpus_sents)
```

	Text <chr>	Types <int>	Tokens <int>	Sentences <int>
1	Ulysses.txt.1	51	99	1
2	Ulysses.txt.2	16	17	1
3	Ulysses.txt.3	14	14	1
4	Ulysses.txt.4	19	20	1
5	Ulysses.txt.5	7	7	1
6	Ulysses.txt.6	10	10	1
7	Ulysses.txt.7	15	18	1
8	Ulysses.txt.8	24	29	1
9	Ulysses.txt.9	36	47	1
10	Ulysses.txt.10	19	20	1

1-10 of 100 rows Previous **1** 2 3 4 5 6 … 10 Next

```
# Total number of units in the new corpus structure

ndoc(corpus_sents)
```

```
## [1] 22730
```

The examples above show us that a corpus is a set of texts with information about each text (metadata), from which we can easily extract the count of tokens, types, and sentences for each text. However, to perform quantitative analyzes on the corpus, we need to break the texts into tokens (tokenization). It is also possible to filter them, removing elements such as punctuation, symbols, numbers, URLs, and separators.

```
# Tokenize our three corpora

toks_unico <- tokens(corpus_unico)
toks_sents <- tokens(corpus_sents)
toks_files <- tokens(corpus_clean)
```

```
## Next, we filter the three corpora in various ways, for demonstration

# Remove punctuation (clean corpus with regex)

toks_nopunct_files <- tokens(corpus_clean, remove_punct = TRUE)
toks_nopunct_unico <- tokens(corpus_unico, remove_punct = TRUE)

# Remove numbers (corpus with only one file)

toks_nonumbr <- tokens(corpus_unico, remove_numbers = TRUE)

# Remove separators (Unicode categories "Separator" [Z] and "Control" [C]) (corpus made by sent
ences)

toks_nosept <- tokens(corpus_sents, remove_separators = TRUE)

# Remove various elements at the same time (corpus with only one file)

toks_simples <- tokens(corpus_unico, remove_numbers = TRUE, remove_symbols = TRUE, remove_punct
= TRUE)
```

It is also possible to remove unwanted tokens. Quanteda offers a list of 'stopwords' for different languages. Stopwords, or empty words in Portuguese, are words to be removed during text processing for computational analyses. There is no standard list, but generally stopwords are the most frequently used words in a language, such as prepositions and articles. The block below eliminates the words included in the list of stopwords for Portuguese and also includes other words that are repeated in the corpus in question.

```
# Remove stopwords from the corpus made with a single file

toks_nostop <- tokens_select(toks_unico, pattern = stopwords("en"), selection = "remove")

# Remove specific tokens from the corpus made with multiple files and cleaned with regex, after
removing punctuation

toks_selected_files <- tokens_select(toks_nopunct_files, pattern = c("o", "said", "say", "say
s", "like", "come", "get", "us", "go", "must", "put", "can", "j", "let", "came", "ask", "aske
d", "don't", "went", "got", "give", "much", "it's", "that's", "I'm", "he's", "she's", "give",
"gave", "told", "tell", "mr", "one", "two", "three", "don't", "sir", "mrs", "just", "might", "i
t's", "don't", "that's", "I'm", "he's", "she's", stopwords("en")), selection = "remove")

# Remove specific tokens from the corpus made with one file, after removing punctuation

toks_selected_unico <- tokens_select(toks_nopunct_unico, pattern = c("o", "said", "say", "say
s", "like", "come", "get", "us", "go", "must", "put", "can", "j", "let", "came", "ask", "aske
d", "don't", "went", "got", "give", "much", "it's", "that's", "I'm", "he's", "she's", "give",
"gave", "told", "tell", "mr", "one", "two", "three", "don't", "sir", "mrs", "just", "might", "i
t's", "don't", "that's", "I'm", "he's", "she's", stopwords("en")), selection = "remove")
```

After tokenization, the next step is to create a table with the frequency of each token for each text, or, in Quanteda's terms, a `document-feature-matrix` (DFM). The DFM is a prerequisite for several other functions in Quanteda, such as `topfeatures`, which returns the most frequent tokens in a corpus.

```
# Here we can see the 20 most frequent words when removing numbers, symbols, and punctuation
```

```
dfm_simples <- dfm(toks_simples)
print("With the removal of numbers, symbols, and punctuation")
```

```
## [1] "With the removal of numbers, symbols, and punctuation"
```

```
topfeatures(dfm_simples, 20)
```

```
##   the    of   and     a    to    in    he   his     i  that  with    it   was
## 14882  8138  7206  6488  4953  4909  4028  3328  2681  2603  2514  2349  2131
##    on   for   you   her   him    is   all
##  2108  1932  1889  1783  1522  1432  1317
```

```
dfm_nostop <- dfm(toks_nostop)
print("Removal of stopwords")
```

```
## [1] "Removal of stopwords"
```

```
topfeatures(dfm_nostop, 20)
```

```
##     .     ,     _     :     —     ?     )     (     !  said
## 22185 16361  2720  2568  2318  2233  1788  1776  1574  1208
## bloom  like    mr   one stephen   old  says   now   see   man
##   933   731   717   703   503   487   473   438   432   410
```

```
dfm_selected_unico <- dfm(toks_selected_unico)
print("Removal of selected tokens in the corpus previously cleaned with regex and without stopw
ords")
```

```
## [1] "Removal of selected tokens in the corpus previously cleaned with regex and without stop
words"
```

```
topfeatures(dfm_selected_unico, 20)
```

```
##   bloom stephen     old     now     see     man    time    back     yes    eyes
##     933     503     487     438     432     410     376     361     358     329
##    know    good    hand  street  little   first  father     way    well   never
##     327     321     302     293     290     278     277     276     273     251
```

```
dfm_selected_files <- dfm(toks_selected_files)
print("Removal of selected tokens in the single file corpus and without stopwords")
```

```
## [1] "Removal of selected tokens in the single file corpus and without stopwords"
```

```
topfeatures(dfm_selected_files, 20)
```

```
##   bloom stephen     old     man     now  little    time    eyes     see    back
##     934     874     740     730     716     631     608     598     597     570
##    know    good     yes  father    face   first    well    hand     god     day
##     541     513     489     479     472     463     451     448     442     412
```

After generating the token list, we can explore the corpus. One of the simplest and most used techniques for corpus investigation is through concordance lines, also known as concordance lines or keywords in context (`kwic`). Concordance lines show fragments of the corpus where the searched terms occur. The number of words in the context can be stipulated by the user, with 5 tokens to the left and 5 to the right being the standard. The first column indicates the name of the file where the searched word occurs. There are several options for searches. They can be made by words or by fragments, sequences, or combinations thereof.

```
# Occurrences of words that start with "feli*"
kwic(toks_unico, pattern =  "happ*")
```

docname <chr>	from <int>	to <int>	pre <chr>	keyword <chr>
Ulysses.txt	2413	2413	sensations . Why ? What	happened
Ulysses.txt	6940	6940	, his wellshaped mouth open	happily
Ulysses.txt	6984	6984	to chant in a quiet	happy
Ulysses.txt	13608	13608	free . — I am	happier
Ulysses.txt	22523	22523	often as he walked in	happy
Ulysses.txt	27004	27004	? No , nothing has	happened
Ulysses.txt	27222	27222	backbone , increasing . Will	happen
Ulysses.txt	27239	27239	sweet light lips . Will	happen
Ulysses.txt	30458	30458	and the peri . Always	happening
Ulysses.txt	31535	31535	their haunches . Might be	happy

1-10 of 84 rows | 1-5 of 7 columns Previous **1** 2 3 4 5 6 … 9 Next

```
# We can also search for more than one word at the same time
kwic(toks_unico, pattern = c("happ*", "joy*"))
```

docname <chr>	from <int>	to <int>	pre <chr>	keyword <chr>
Ulysses.txt	4	4	Ulysses by James	Joyce
Ulysses.txt	2413	2413	sensations . Why ? What	happened
Ulysses.txt	6940	6940	, his wellshaped mouth open	happily
Ulysses.txt	6984	6984	to chant in a quiet	happy
Ulysses.txt	11981	11981	you couldn’t , he said	joyously

docname <chr>	from <int>	to <int>	pre <chr>	keyword <chr>
Ulysses.txt	13608	13608	free . — I am	happier
Ulysses.txt	22523	22523	often as he walked in	happy
Ulysses.txt	27004	27004	? No , nothing has	happened
Ulysses.txt	27222	27222	backbone , increasing . Will	happen
Ulysses.txt	27239	27239	sweet light lips . Will	happen

1-10 of 123 rows | 1-5 of 7 columns Previous 1 2 3 4 5 6 … 13 Next

```
# By sequence of more than one token
kwic(toks_unico, pattern = phrase("I dream*"))
```

docname <chr>	from <int>	to <int>	pre <chr>	keyw… <chr>	post <chr>
Ulysses.txt	161066	161067	, all is prepared .	I dreamt	. What ? Worst is
Ulysses.txt	232076	232077	STEPHEN : Mark me .	I dreamt	of a watermelon . ZOE
Ulysses.txt	314577	314578	look at that and didnt	I dream	something too yes there was

3 rows

1.4.1 N-grams

Word frequency lists can be useful for identifying common elements in a text. However, in many cases, it is equally important to know in what context these words are. Identifying which words frequently co-occur in a corpus can provide us with even more information about the text. For example, knowing that the sequence 'I am sad' frequently occurs in the corpus gives us richer insights than just the frequency of the word 'sad' alone. The sequence 'I am sad' is an example of what we call n-grams, or, in this specific case, bigrams. N-grams are sequences of two or more words that occur in a text. To generate lists of n-grams, we start from a list of tokens and specify the minimum and maximum number of tokens in each n-gram.

```
# Create a list of bigrams, trigrams, and tetragrams
toks_ngram <- tokens_ngrams(toks_simples, n = 2:4)

# Visualize only the 30 most frequent
head(toks_ngram[[1]], 30)
```

```
##  [1] "Ulysses_by"        "by_James"          "James_Joyce"
##  [4] "Joyce_Contents"    "Contents_I"        "I_II"
##  [7] "II_III"            "III_I"             "I_Stately"
## [10] "Stately_plump"     "plump_Buck"        "Buck_Mulligan"
## [13] "Mulligan_came"     "came_from"         "from_the"
## [16] "the_stairhead"     "stairhead_bearing" "bearing_a"
## [19] "a_bowl"            "bowl_of"           "of_lather"
## [22] "lather_on"         "on_which"          "which_a"
```

```
## [25] "a_mirror"          "mirror_and"        "and_a"
## [28] "a_razor"           "razor_lay"         "lay_crossed"
```

1.4.2 Dictionary

Another way to extract information from a text is through the creation of “dictionaries”. The `dictionary` function in Quanteda allows grouping tokens by categories. This categorization can then be used for searches in the corpus. For example, we can create the categories “joy” and “sadness” containing words related to these feelings, respectively. With the dictionary created, we can identify the distribution of these terms in a corpus.

```
# Create a dictionary from the corpus formed by a single document

dict <- dictionary(list(happiness = c("happ*", "joy*", "smil*", "content*", "cheer*", "delig
*"),
                        sadness = c("sad*", "unhapp*", "soorw*", "depress*", "dejec*", "misera
*")))

dict_toks <- tokens_lookup(toks_unico, dictionary = dict)
print(dict_toks)
```

```
## Tokens consisting of 1 document.
## Ulysses.txt :
##  [1] "happiness" "happiness" "happiness" "happiness" "happiness" "happiness"
##  [7] "sadness"   "happiness" "happiness" "happiness" "happiness" "happiness"
## [ ... and 405 more ]
```

```
dfm(dict_toks)
```

```
## Document-feature matrix of: 1 document, 2 features (0.00% sparse) and 0 docvars.
##              features
## docs          happiness sadness
##   Ulysses.txt       341      76
```

```
# Create a dictionary from the corpus formed by multiple documents

dict <- dictionary(list(happiness = c("happ*", "joy*", "smil*", "content*", "cheer*", "delig
*"),
                        sadness = c("sad*", "unhapp*", "soorw*", "depress*", "dejec*", "misera
*")))

dict_toks <- tokens_lookup(toks_files, dictionary = dict)
print(dict_toks)
```

```
## Tokens consisting of 4 documents.
## A Portrait of the Artist as a Young Man.txt :
##  [1] "happiness" "happiness" "happiness" "happiness" "happiness" "happiness"
##  [7] "happiness" "happiness" "happiness" "sadness"   "sadness"   "sadness"
## [ ... and 168 more ]
##
## Dubliners.txt :
##  [1] "happiness" "happiness" "happiness" "happiness" "happiness" "happiness"
##  [7] "happiness" "happiness" "happiness" "happiness" "happiness" "happiness"
```

```
## [ ... and 117 more ]
##
## PPn25_01.pdf :
## [1] "happiness"
##
## Ulysses.txt :
##  [1] "happiness" "happiness" "happiness" "happiness" "happiness" "happiness"
##  [7] "sadness"   "happiness" "happiness" "happiness" "happiness" "happiness"
## [ ... and 405 more ]
```

```
dfm(dict_toks)
```

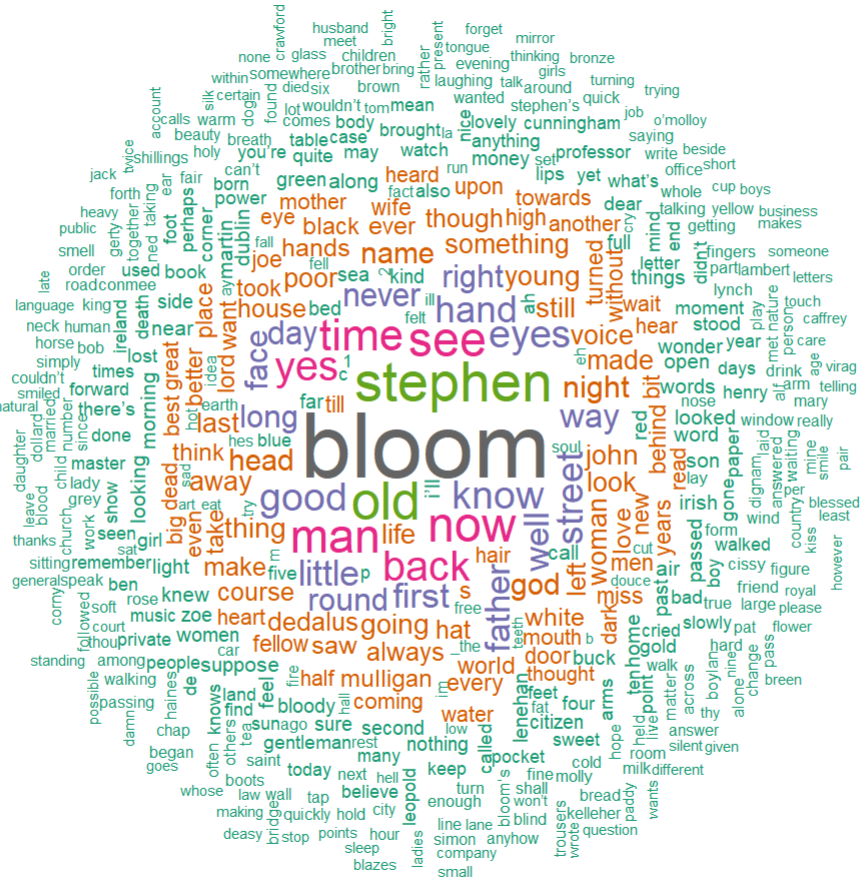
```
## Document-feature matrix of: 4 documents, 2 features (12.50% sparse) and 0 docvars.
##                                                 features
## docs                                             happiness sadness
##   A Portrait of the Artist as a Young Man.txt          148      32
##   Dubliners.txt                                        117      12
##   PPn25_01.pdf                                           1       0
##   Ulysses.txt                                          341      76
```

2 Data Visualization and Analysis

2.1 Word Cloud and Frequency Graph

In 1.4, we created a DFM with the frequency of tokens. To absorb these frequencies more quickly, we can generate visualizations. One option is the word cloud, a graph that allows for quick visualization of the most frequent terms.

```
# Demonstration of how word frequencies change depending on the corpus preparation

set.seed(100) # For the reproduction of the results

textplot_wordcloud(dfm_selected_unico, min_count = 6, random_order = FALSE, rotation = .25, col
or = RColorBrewer::brewer.pal(8, "Dark2"))
```




```
set.seed(100)
textplot_wordcloud(dfm_selected_files, min_count = 6, random_order = FALSE, rotation = .25, col
or = RColorBrewer::brewer.pal(8, "Dark2"))
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## followed could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## friends could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## beautiful could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## beauty could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## drink could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## person could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
```

```
## breath could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## won't could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## smiling could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## suddenly could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## taking could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## simon could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## large could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## certain could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## henry could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## private could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## different could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## passing could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## really could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## opened could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## getting could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## smiled could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## school could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## daughter could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## college could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## yellow could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, : chap
## could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## turning could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## heavy could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## around could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## language could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## business could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, : fall
## could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## smile could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## number could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## making could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## whether could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, : neck
## could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## present could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## couldn't could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## since could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, : pass
## could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## given could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## closed could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## married could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, : ago
## could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## boots could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## bring could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## blessed could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## conmee could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## bridge could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## leopold could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## raised could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## laughter could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## taken could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## simply could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## temple could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, : jack
## could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## thinking could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## sleep could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## understand could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## dignam could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## sight could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## square could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## twice could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## faces could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## heaven could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## chapel could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, : meet
## could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## stopped could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## wrote could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## moved could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## shadow could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## tongue could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## gentlemen could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## however could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## m’coy could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## sound could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## bread could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## laugh could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## written could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, : help
## could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## quick could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## shook could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## broke could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## molly could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## remembered could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## bright could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## grace could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## speech could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## bloom’s could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## telling could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## quietly could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## trying could not be fit on page. It will not be plotted.
```

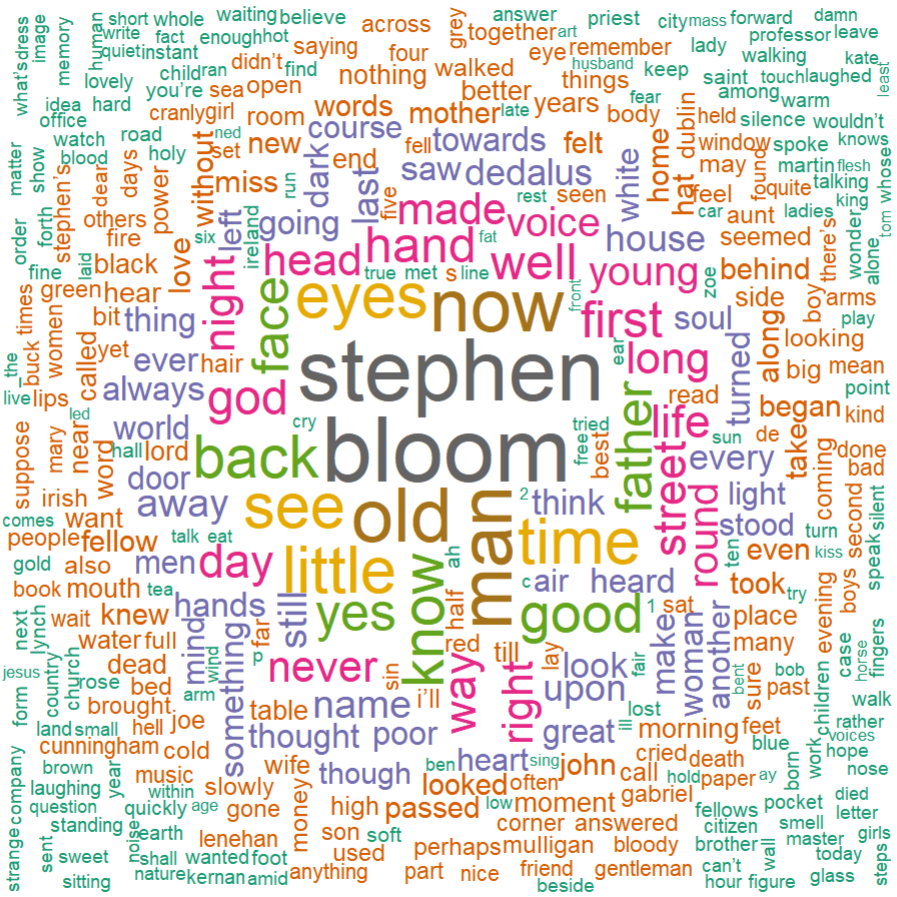
```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## haines could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## cissy could not be fit on page. It will not be plotted.
```

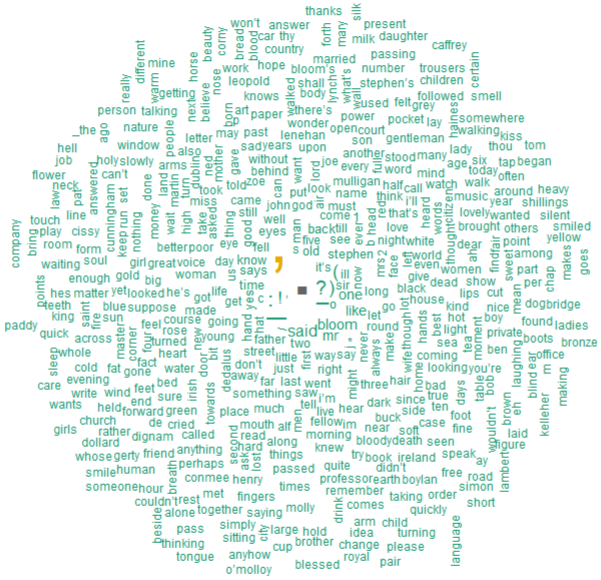
```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## somewhere could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## please could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(x, min_size, max_size, min_count, max_words, color, :
## seeing could not be fit on page. It will not be plotted.
```

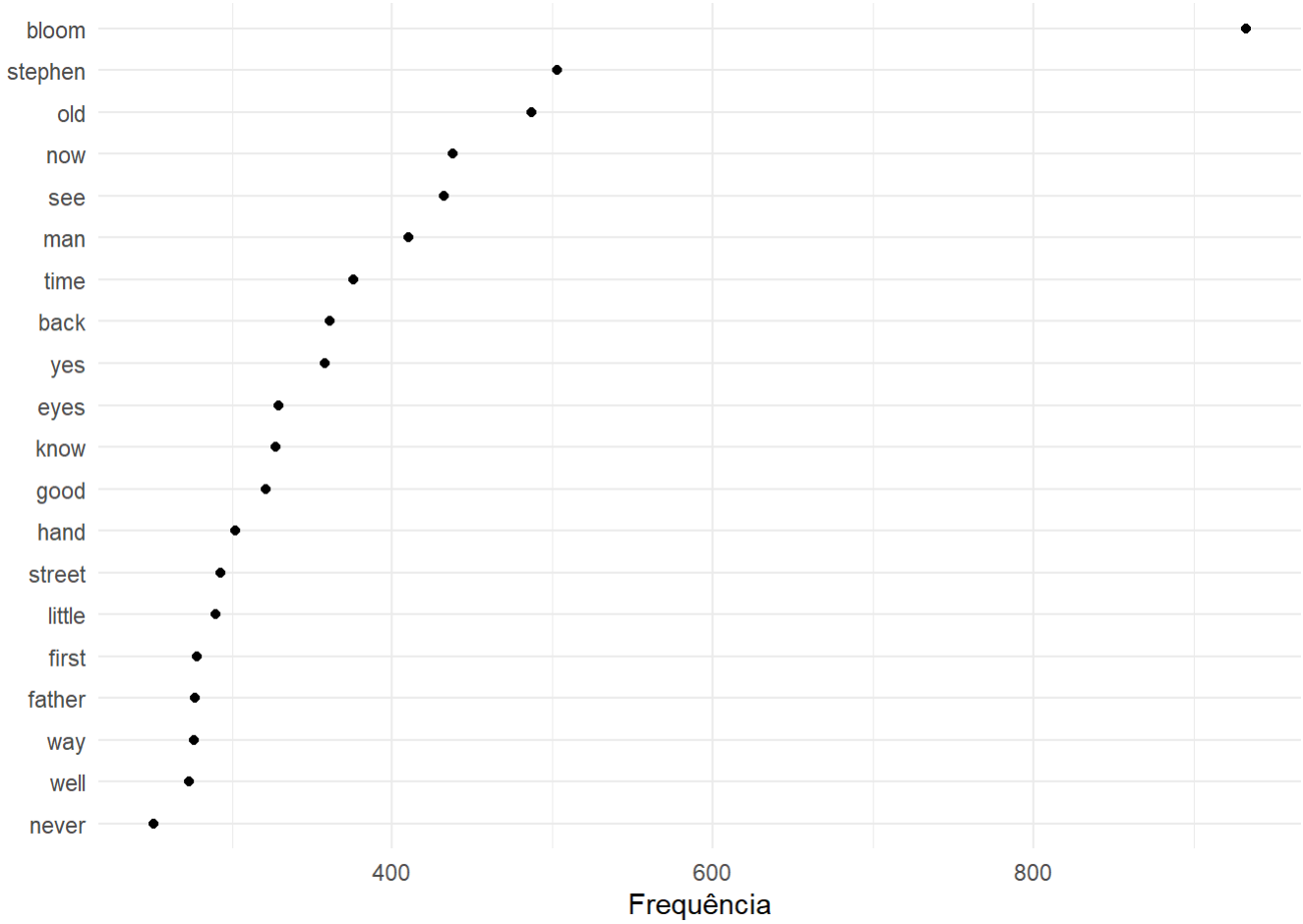



```
set.seed(100)
textplot_wordcloud(dfm_nostop, min_count = 6, random_order = FALSE, rotation = .25, color = RCo
lorBrewer::brewer.pal(8, "Dark2"))
```

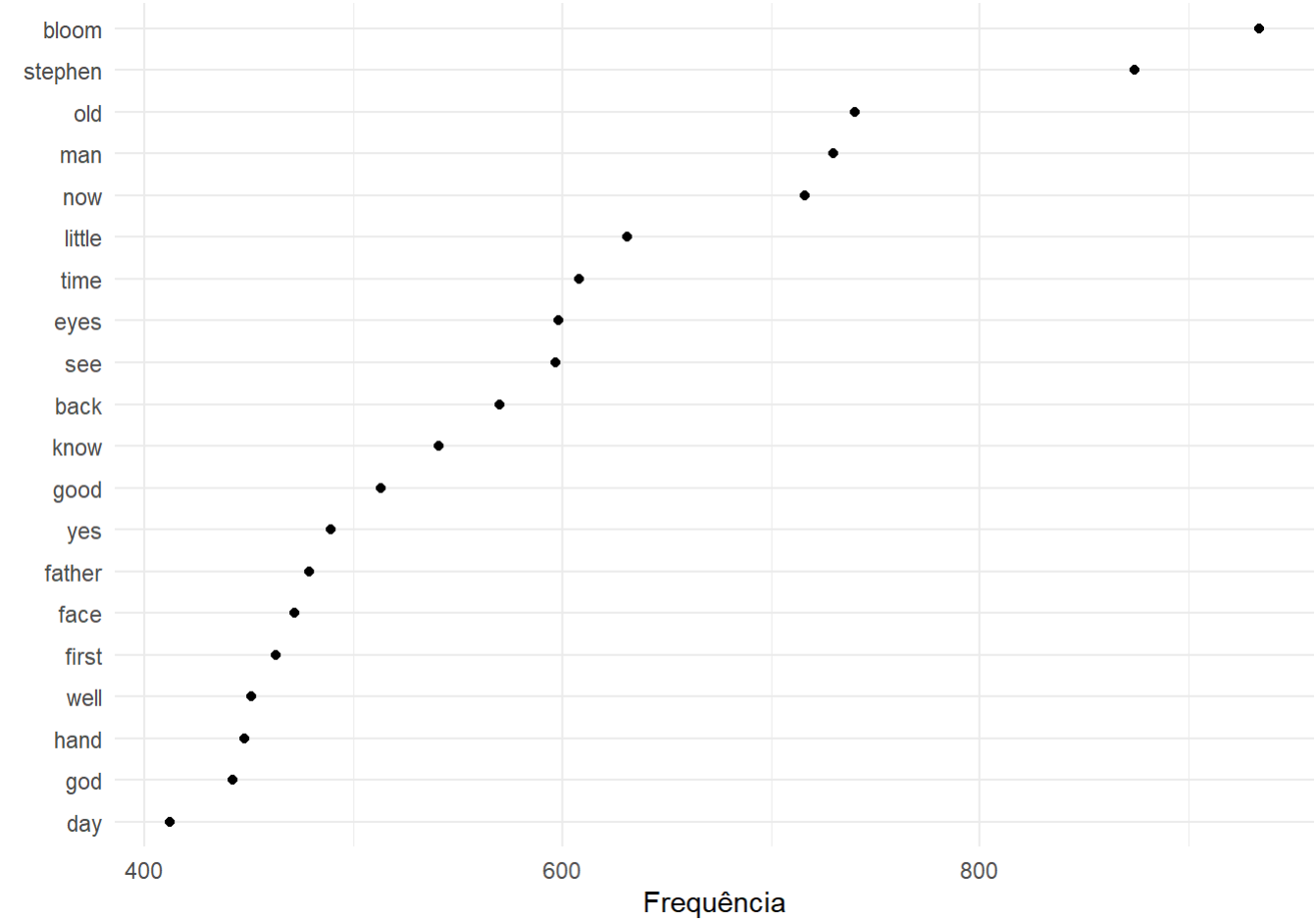


Another solution is to use the ´ggplot` library and represent in a graph the number of occurrences of the most frequent words.

```
# From the corpus formed by a single document

dfm_selected_unico %>%
  textstat_frequency(n = 20) %>%
  ggplot(aes(x = reorder(feature, frequency), y = frequency)) +
  geom_point() +
  coord_flip() +
  labs(x = NULL, y = "Frequência") +
  theme_minimal()
```




```
# From a corpus formed by multiple documents

dfm_selected_files %>%
  textstat_frequency(n = 20) %>%
  ggplot(aes(x = reorder(feature, frequency), y = frequency)) +
  geom_point() +
  coord_flip() +
  labs(x = NULL, y = "Frequência") +
  theme_minimal()
```



2.2 Topic Modeling (LDA)

Another function frequently used in Natural Language Processing (NLP) is topic modeling (TM). Topic modeling applies a statistical model that seeks to understand the structure of the corpus and identify and group words that are related in some way. TM uses a semi or unsupervised technique to identify these topics. In other words, the program learns to recognize patterns in the data without the need for prior annotations. The code below demonstrates the application of the Latent Dirichlet Allocation (LDA) model.

```
# Topic modeling from the corpus formed by a single document

lda <- LDA(dfm_selected_unico, k = 10)
terms(lda, 10)
```

```
##       Topic 1   Topic 2   Topic 3   Topic 4   Topic 5  Topic 6 Topic 7
##  [1,] "back"    "back"    "bloom"   "bloom"   "bloom"  "bloom" "bloom"
##  [2,] "stephen" "old"     "old"     "stephen" "now"    "old"   "now"
##  [3,] "now"     "bloom"   "stephen" "old"     "way"    "eyes"  "see"
##  [4,] "time"    "now"     "time"    "now"     "man"    "see"   "stephen"
##  [5,] "first"   "see"     "eyes"    "man"     "see"    "man"   "know"
##  [6,] "voice"   "stephen" "going"   "hand"    "well"   "know"  "never"
##  [7,] "father"  "time"    "hand"    "back"    "yes"    "good"  "man"
##  [8,] "old"     "first"   "look"    "yes"     "first"  "day"   "poor"
##  [9,] "street"  "good"    "round"   "little"  "john"   "hand"  "hat"
## [10,] "little"  "little"  "john"    "see"     "street" "long"  "street"
##       Topic 8   Topic 9 Topic 10
##  [1,] "bloom"   "man"   "stephen"
##  [2,] "stephen" "well"  "time"
##  [3,] "yes"     "old"   "father"
##  [4,] "back"    "bloom" "man"
```

```
##  [5,] "see"     "way"   "way"
##  [6,] "good"    "thing" "see"
##  [7,] "old"     "time"  "old"
##  [8,] "know"    "eyes"  "john"
##  [9,] "well"    "yes"   "street"
## [10,] "face"    "now"   "now"
```

```
# Topic modeling from a corpus formed by multiple documents

lda <- LDA(dfm_selected_files, k = 10)
terms(lda, 10)
```

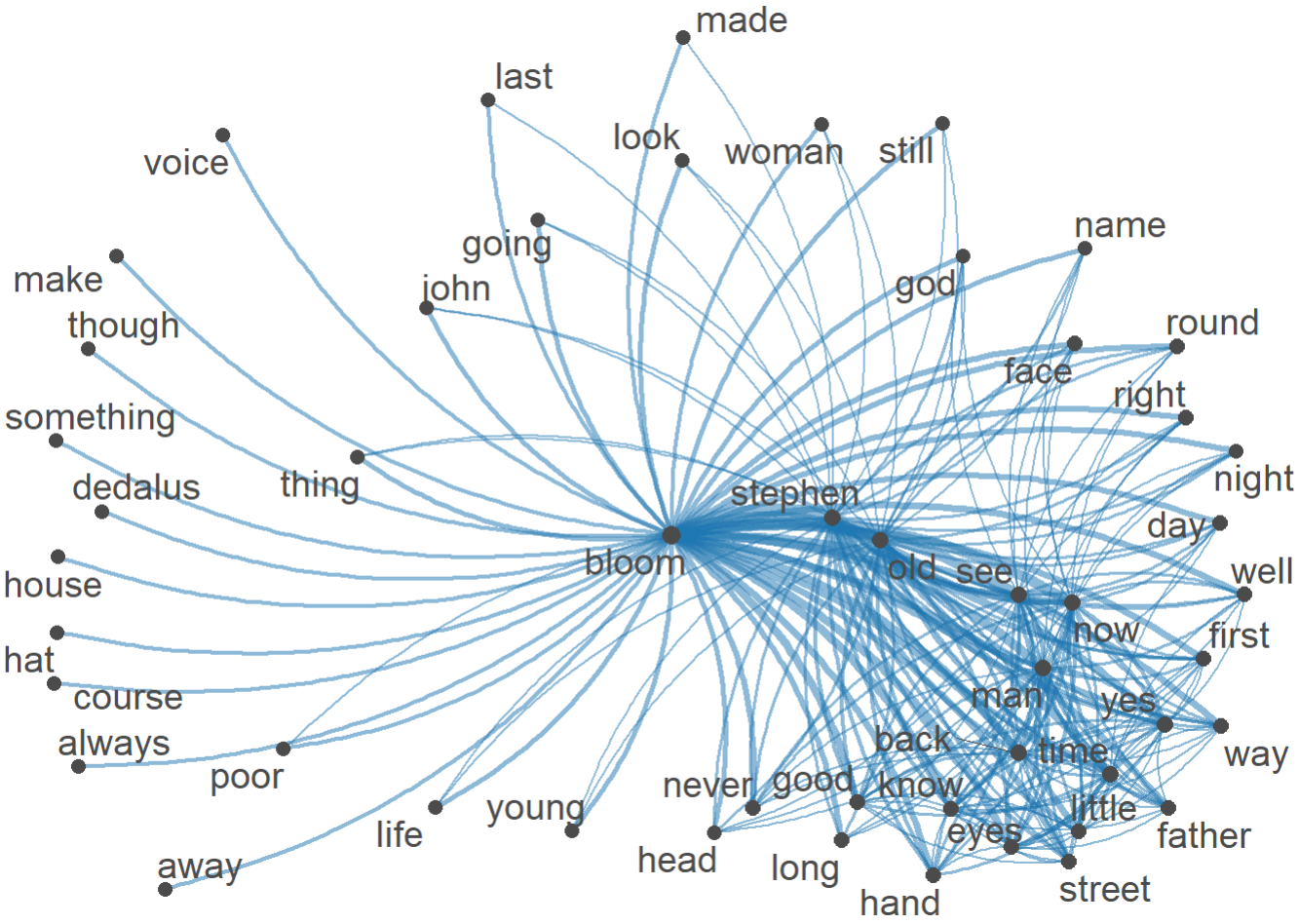
```
##       Topic 1   Topic 2   Topic 3   Topic 4    Topic 5  Topic 6  Topic 7
##  [1,] "eyes"    "stephen" "never"   "de"       "man"    "bloom"  "see"
##  [2,] "stephen" "god"     "bloom"   "pessoa"   "well"   "old"    "bloom"
##  [3,] "now"     "soul"    "now"     "fernando" "time"   "good"   "stephen"
##  [4,] "air"     "first"   "stephen" "recensão" "along"  "see"    "man"
##  [5,] "dedalus" "father"  "time"    "e"        "house"  "now"    "now"
##  [6,] "heart"   "life"    "yes"     "um"       "miss"   "well"   "life"
##  [7,] "upon"    "mind"    "head"    "não"      "head"   "father" "old"
##  [8,] "passed"  "little"  "man"     "filme"    "fellow" "thing"  "day"
##  [9,] "face"    "hell"    "way"     "2023"     "world"  "know"   "eyes"
## [10,] "back"    "old"     "eyes"    "boca"     "day"    "street" "long"
##       Topic 8   Topic 9   Topic 10
##  [1,] "back"    "little"  "made"
##  [2,] "stephen" "gabriel" "man"
##  [3,] "bloom"   "now"     "upon"
##  [4,] "hand"    "old"     "stephen"
##  [5,] "time"    "aunt"    "life"
##  [6,] "eyes"    "know"    "turned"
##  [7,] "call"    "young"   "god"
##  [8,] "yes"     "began"   "face"
##  [9,] "name"    "good"    "now"
## [10,] "god"     "face"    "heard"
```

2.3 Semantic Network

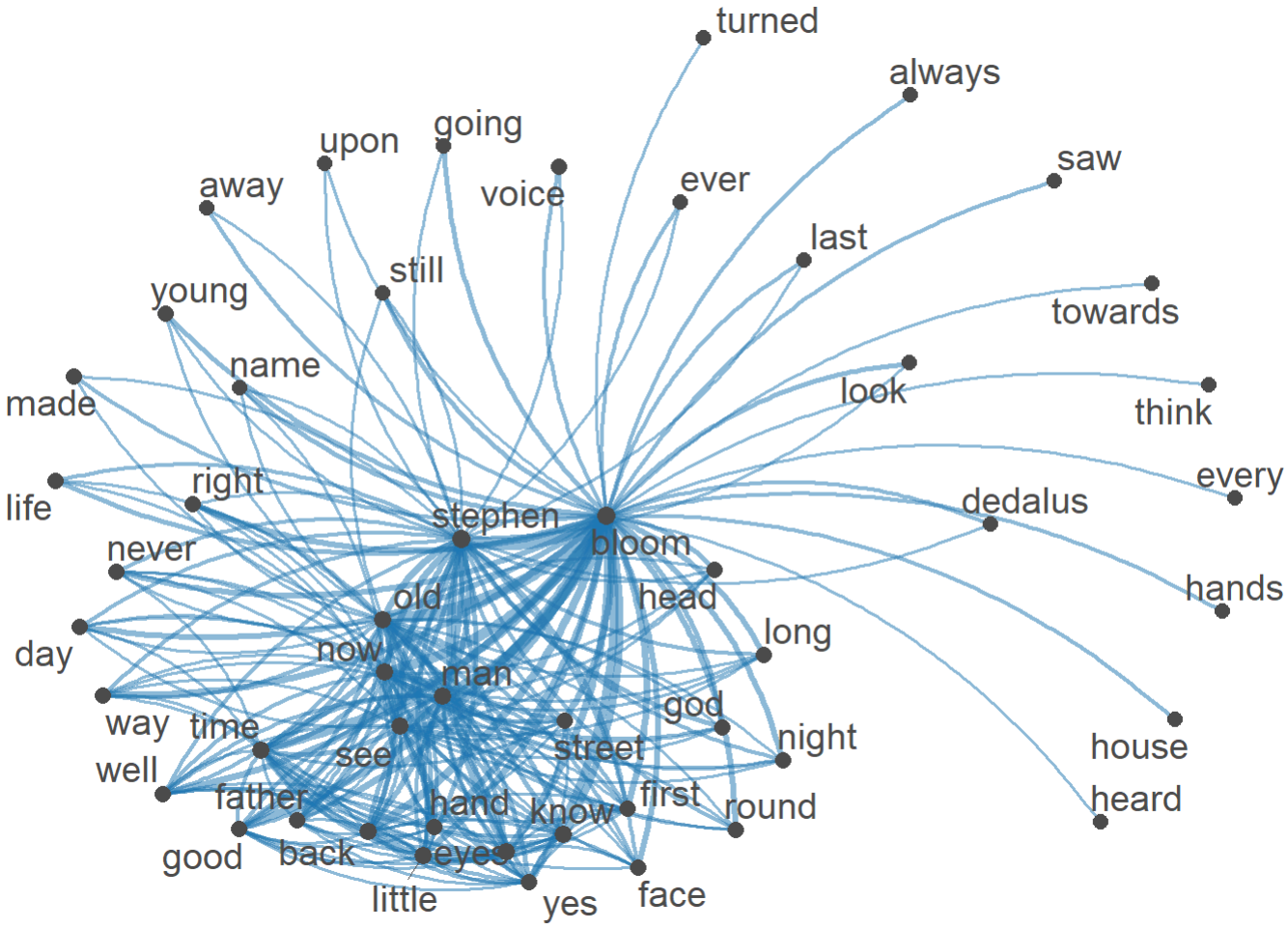
The Feature `Co-occurrence Matrix` (FCM) is similar to the DFM but considers co-occurrences, presenting a graph with semantic networks.

```
# Network from the corpus formed by a single document

# Create FCM from DFM
fcm_nostop <- fcm(dfm_selected_unico)

# List the top features
feat <- names(topfeatures(dfm_selected_unico, 50))

# Select
fcm_select <- fcm_select(fcm_nostop, pattern = feat, selection = "keep")

size <- log(colSums(dfm_select(dfm_selected_unico, feat, selection = "keep")))
```

```
textplot_network(fcm_select, min_freq = 0.8, vertex_size = size / max(size) * 3)
```




```
# Network from a corpus formed by multiple documents

# Create a Feature Co-occurrence Matrix (FCM) from DFM
fcm_nostop <- fcm(dfm_selected_files)

# List the top features
feat <- names(topfeatures(dfm_selected_files, 50))

# Select features for the FCM
fcm_select <- fcm_select(fcm_nostop, pattern = feat, selection = "keep")

# Calculate vertex sizes
size <- log(colSums(dfm_select(dfm_selected_files, feat, selection = "keep")))

textplot_network(fcm_select, min_freq = 0.8, vertex_size = size / max(size) * 3)
```



Data and Repository

The data and codes are available via GitHub https://github.com/DiegoEGimenez/R_literatura_Quanteda

The code can be viewed at https://rpubs.com/DiegoEGimenez/1192115

Acknowledgments

This document (2024) contains a review and expansion of codes originally prepared by Diego Giménez and Andressa Gomide in 2022 for the analysis of “The Book of Disquiet”. Some of the codes described in the 2022 document used the codes kindly provided by Mark Alfano, used in his work “Nietzsche corpus analysis”.