university of groningen

Quantitative Text Analysis – Essex Summer School

Principles of QTA. Cleaning data. Preprocessing data

dr. Martijn Schoonvelde

University of Groningen

Today's class

- Principles of automated text analysis
- Clean text in R: string operations and regular expressions (using the **stringr** library)
- Preprocessing data: going from text to numbers (using the **quanteda** library)

Principles of Automated Text Analysis (Grimmer, Roberts & Stewart 2022)

These principles represents an agnostic approach to text analysis as opposed to a structural approach

1. Social science theories and domain knowledge are essential for research design
   - E.g., when examining parliamentary speech, we need to know how a bill becomes a law in a country.
2. QTA does not replace humans – it augments them
   - For example, QTA may help us categorise our documents or find key documents
3. We have to separate the discovery phase from the testing phase
4. The best method depends on the task
5. Validations are essential and depend on the theory and the task

Data preparation

Once we have selected our texts it's time to prepare them for analysis

1. Import the texts into R
2. Get rid of irrelevant parts in a text
   - This can be anything: headers, html tags, page numbers, etc.
3. Preprocess the texts – reduce their complexity while retaining the most relevant features

Table 1. An overview of text analysis operations, with the R packages used in this Teacher's Corner.

	R packages	
Operation	example	alternatives
Data preparation		
importing text	*readtext*	*jsonlite, XML, antiword, readxl, pdftools*
string operations	*stringi*	*stringr*
preprocessing	*quanteda*	*stringi, tokenizers, snowballC, tm, etc.*
document-term matrix (DTM)	*quanteda*	*tm, tidytext, Matrix*
filtering and weighting	*quanteda*	*tm, tidytext, Matrix*
Analysis		
dictionary	*quanteda*	*tm, tidytext, koRpus, corpustools*
supervised machine learning	*quanteda*	*RTextTools, kerasR, austin*
unsupervised machine learning	*topicmodels*	*quanteda, stm, austin, text2vec*
text statistics	*quanteda*	*koRpus, corpustools, textreuse*
Advanced topics		
advanced NLP	*spacyr*	*coreNLP, cleanNLP, koRpus*
word positions and syntax	*corpustools*	*quanteda, tidytext, koRpus*

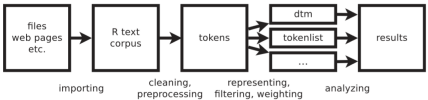


Figure 1. Order of text analysis operations for data preparation and analysis.

Welbers *et al.*(2017)

Importing text

- Exact steps depend on how texts are stored
  - When stored as txt or csv files you can work with base R functions such **read.csv(), read.txt()** or **read_csv**, or **read_tsv** from the Tidyverse
  - Often text is stored in other formats (JSON, XML, HTML, PDF) – needs to be transformed into R objects
    - Functions in the **readtext** library can help with this
- In many cases there exist multiple libraries and functions for doing the same thing
  - R open source – different groups of developers tackling similar issues
  - Use that to your advantage: Google / Stackexchange / other resources

Importing text

When importing text – especially non-English text – you can run into issues of character encoding

- Bit and bytes of information are transformed through a coding scheme (e..g, UTF-8, UTF-16, ASCII) into readable text on your screen
- Many such coding schemes exists, and sometimes R assumes an incorrect scheme, leading to all sorts of weird characters being displayed a text.
- No one size fits all solution

Some background: `http://kunststube.net/encoding/`

Text snippet

```
<p>Ladies and gentlemen,</p><p>It is an honour to be here today to introduce the
theme of 'recession and recovery'. If you will permit,   I would like to suggest that
this afternoon we focus more on recovery than on recession. I think we know enough
about the recession side   of the story.</p><p>It started with the fall of Lehman
Brothers on 15 September 2008.. I happened to be here, at the Blouin Creative
Leadership Summit, only ten days later. Everyone was talking about the collapse of
Lehman. They were shocked and alarmed.   But even then we could hardly imagine that
its impact would be so dramatic, so historic.</p><p>As we now know, this event
triggered a global financial and economic crisis. Governments were forced to give
cash injections running into billions to prevent an economic and financial meltdown.
When credit dried up and demand fell, businesses struggled  to keep their heads above
water, and many went under. Ordinary people's jobs, homes and pensions were at
risk.</p><p>
```

Clean text, string operations

- This can be done in base R
  - That is, no packages needed
- But also in specific libraries
  - E.g, **stringr**, **stringi**
- Text analysis libraries can do a lot, but often necessary to dive into regular expressions, a concise language for describing patterns of text.
  - Expect lots of trial and error
  - Check the **stringr** cheat sheet:
    `https://github.com/rstudio/cheatsheets/blob/master/strings.pdf`

Clean text, string operations: the stringr library

All functions in **stringr** start with **str_** and take a vector of strings as the first argument. Most functions work with regular expressions

```
> corpus <- c("a", "ab", "abba", "bbbb")
> str_length(corpus)
[1] 1 2 4 4
> str_count(corpus, "a")
[1] 1 1 2 0
> str_detect(corpus, "a")
[1]  TRUE  TRUE  TRUE FALSE
```

Stringr example

```
corpus <- c(" This Is text_one <font size = '6'> ",
            "And here is teXt Number 2!?",
            "And text %%$ number 3")

#remove html tag
> corpus <- str_remove(corpus, "<.*>")
> print(corpus)
[1] " This Is text_one  "
[2]"And here is teXt Number 2!?"
[3] "And text %%$ number 3"
```

Stringr example

Transform to lower case

```
corpus <- str_to_lower(corpus)
print(corpus)
```

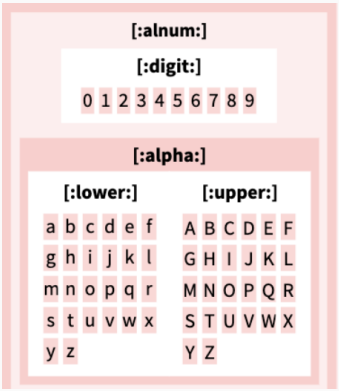
```
[1] " this is text_one  "
[2] "and here is text number 2!?"
[3] "and text %%$ number 3"
```

Stringr example

Remove everything but letters or numbers

```
corpus <- str_replace_all(corpus, "[^[:alnum:]]
print(corpus)
```

```
[1] " this is text one  "
[2] "and here is text number 2    "
[3] "and text number 3  "
```



Source: RStudio Stringr cheat sheet

Stringr example

Remove multiple white spaces, as well as surrounding white spaces

```
corpus <- str_squish(corpus)
print(corpus)
```

```
[1] "this is text one"
[2] "and here is text number 2"
[3] "and text number 3"
```

Biden inaugural address

"So now, on this hallowed ground where just days ago violence sought to shake this Capitol's very foundation, we come together as one nation, under God, indivisible, to carry out the peaceful transfer of power as we have for more than two centuries." (Biden, 2021)

Bag of words

- Also known as document-term-matrix or document-feature-matrix
- Matrix with each row a document and each column a word. Each cell denotes the number of times a particular word appears in a particular document
- This is its most common form but specific features may vary
  - 1-gram, 2-gram, 3-gram
  - Word weights (tf-idf)
  - Yes / No

Biden inaugural address

"So now, on this hallowed ground where just days ago violence sought to shake this Capitol's very foundation, we come together as one nation, under God, indivisible, to carry out the peaceful transfer of power as we have for more than two centuries." (Biden, 2021)

	features									
docs	so	now	,	on	this	hallowed	ground	where	just	days
2021-Biden.12	1	1	5	1	2	1	1	1	1	1

Implications of bag of words

- Pros
  - Reduce complexity while retaining much information
- Cons
  - Negations are discarded ("not good") appears as c("not", "good")
    - But taken into account when using multi-word expressions
  - Does not deal with polysemous words
  - More generally: semantic context gets lost - this is different from representations based on more recent word embeddings models or transformer models

Bag of Words Model

1. Choose the unit of analysis
2. Tokenize
3. Reduce complexity
    - Lowercase
    - Remove punctuation
    - Remove stopwords
    - Create equivalent classes (lemmatize / stem)
        - Treat tokens with a similar root as one type
    - Filter by frequency
4. Create the document-feature matrix

Tokenization

Tokenization is the process through we divide up a text into discrete words. Each individual words is called a token, and each token is of a particular type

- The set of types in our corpus is called the vocabulary

Tokenization

```
biden_sentence <- corpus(biden_sentence}
biden_tokens <- tokens(biden_sentence)
as.character(biden_tokens)
```

```
 [1] "So"  "now"  ","   "on"   "this" "hallowed"  "ground"
 [6] "where"  "just"  "days"  "ago"  "violence" "sought"  "to"
[15] "shake"  "this" "Capitol's"  "very"  "foundation"  "," "we"
[22] "come"  "together" "as" "one"  "nation"  ","  "under"
[29] "God"  ","  "indivisible" ","  "to" "carry" "out"
[36] "the"  "peaceful"  "transfer"   "of" "power"  "as"  "we"
[43] "have"  "for"  "more" "than" "two"  "centuries"   "."
```

NB everything that is not a white space is tokenized

Tokenization

The simplest way to tokenize is to divide into unigrams, but sometimes we lose important information (e.g., European Union, United States, International Monetary Fund). This is where multiword expressions come in.

```
> biden_tokens <- tokens_compound(biden_tokens, phrase(c("one nation", "under God")))
> as.character(biden_tokens)

 [1] "So"  "now"  ","  "on"  "this"  "hallowed" "ground"  "where"  "just" "days"
[11] "ago" "violence" "sought"  "to"  "shake"  "this"  "Capitol's" "very"
[19] "foundation"  ","  "we" "come" "together" "as"  "one_nation"  ","
[31] "under_God"  "," "indivisible" "," "to" "carry" "out" "the" "peaceful"
"transfer" "of"  "power"  "as" "we"
[41] "have" "for"  "more"  "than"  "two"  "centuries"   "."
```

Remove punctuation

```
> biden_tokens <- tokens(biden_sentence,
+                        remove_punct = TRUE)
> as.character(biden_tokens)

 [1] "So"  "now"  "on"  "this"  "hallowed" "ground"  "where"
 [8] "just"  "days"  "ago"   "violence"  "sought"  "to" "shake"
[15] "this"  "Capitol's"  "very"  "foundation"  "we" "come"  "together"
[22] "as"  "one"  "nation" "under"  "God"  "indivisible" "to"
[29] "carry"  "out"  "the"  "peaceful"  "transfer" "of"  "power"
[36] "as"  "we"  "have"  "for"  "more" "than" "two"
[43] "centuries"
```

NB everything that is not a white space is tokenized

Remove stopwords

```
> biden_tokens <- tokens_select(biden_tokens,
+                               pattern = stopwords("en"),
+                               selection = "remove")
> as.character(biden_tokens)

 [1] "now"  "hallowed"  "ground"  "just"  "days"  "ago"  "violence"
 [8] "sought"  "shake"  "Capitol's"  "foundation"  "come"  "together" "one"
[15] "nation" "God"  "indivisible" "carry"  "peaceful"  "transfer"  "power"
[22] "two"  "centuries"
```

Stemming

- Stemming: algorithmic conversion of inflected forms of words into their root forms
- Fast but not perfect:
  - Unrelated words may be grouped together; related words may not be grouped together
  - Stems may not be words themselves – problematic if further analysis is based on dictionaries

```
> biden_tokens_stemmed <- tokens_wordstem(biden_tokens,
language = quanteda_options("language_stemmer"))
> as.character(biden_tokens_stemmed)

 [1] "now"  "hallow"  "ground"  "just"  "day" "ago" "violenc" "sought" "shake"
[10] "Capitol" "foundat" "come"  "togeth" "one" "nation" "God" "indivis" "carri"
[19] "peac"  "transfer" "power"  "two"  "centuri"
```

Lemmatization

- Use of a dictionary to replace words with their root form
- Results are always words; neither does it group together unrelated words, nor does it miss to group together related words

```
words <- c("hallowed", "violence", "foundation")
lemma <- rep("XX", length(words))
biden_tokens_lemmatized <- tokens_replace(biden_tokens, words,
                                          lemma,
                                          valuetype = "fixed")
```

```
 [1] "now"   "XX"    "ground"  "just"    "days"   "ago"   "XX"
 [8] "sought"  "shake"   "Capitol's" "XX"   "come" "together" "one"
[15] "nation" "God"   "indivisible" "carry" "peaceful" "transfer"  "power"
[22] "two" "centuries"
```

Preprocessing data in languages other than English

Global linguistic diversity is enormous...

- Some 6,000–7,000 languages are spoken globally today
- Among other dimensions they vary in their script (e.g, Latin, Cyrillic, and Greek), their morphology (the system through which words get formed) and their syntax (how words are brought together to make sentence)

...but diversity in computational tools and methods much less so

- Keep in mind that preprocessing steps are language-specific!

dfm in quanteda

When we are happy with our features, we can create the document feature matrix using the dfm function

```
> dfm(biden_tokens)
Document-feature matrix of: 1 document, 23 features (0.00% sparse) and 4 docvars.
               features
docs             now hallowed ground just days ago violence sought shake capitol’s
  2021-Biden.12    1        1      1    1    1   1        1      1     1
   1
```

dfm in quanteda

```
> dfmat_inagural <- data_corpus_inaugural %>%
+     tokens(remove_punct = TRUE) %>%
+     tokens_remove(stopwords("en")) %>%
+     dfm()
dfmat_inagural[,1:5]
Document-feature matrix of: 59 documents, 5 features (68.47% sparse) and 4 docvars
                 features
docs              fellow-citizens senate house representatives among
  1789-Washington               1      1     2               2     1
  1793-Washington               0      0     0               0     0
  1797-Adams                    3      1     0               2     4
  1801-Jefferson                2      0     0               0     1
  1805-Jefferson                0      0     0               0     7
  1809-Madison                  1      0     0               0     0
[ reached max_ndoc ... 53 more documents ]
```

Filtering and weighting

- Not all terms are equally informative – feature selection
  - Very common words and very rare words
- A simple but effective method is to filter on document frequencies (the number of documents in which a term occurs), using a threshold for minimum and maximum number (or proportion) of documents
- Other possibility, assign weights, using, for example, tf-idf