quanteda initiative

An Introduction to quantitative text analysis using R and **quanteda**

Kenneth Benoit and Kohei Watanabe

ANU/RSSS Data Sprint 2019 (February 20)

Contents

Purpose of the workshop

This workshop is designed to

- introduce the tools for quantitative text analysis using R
- focus specifically on the **quanteda** package
- demonstrate several major categories of analysis
- answer any questions about text analysis that you might have

Whom this short course is for

- Those familiar with text analysis methods but not in R/**quanteda**
- Those who have tried R/**quanteda** but want to learn more proficiency
- The merely "text-curious"
- No real pre-requisites: We're all here to learn

About us

- Ken Benoit (http://kenbenoit.net)
  - Professor of Quantitative Social Research Methods, Department of Methodology, London School of Economics and Political Science; Part-time Professor, School of Politics and International Relatons, ANU
  - Creator and maintainer of the **quanteda** R package and related tools
  - Founder of the Quanteda Initiative, a non-profit company aimed at advancing development and dissemination of open-source tools for text analytics
  - My research: Political science (party competition), measurement, text analysis
- Kohei Watanabe (https://koheiw.net)
  - Studied at the Department of Methodology of the London School of Economics and Political Science for a PhD
  - Assistant Professor at Waseda Institute for Advanced Study (Japan); (From July 1, 2019) Assistant Professor at Digital Science Center of the University of Innsbruck (Austria)
  - **quanteda** co-author, developer of core functions in R and C++
  - Reseach interests: Political communication: media bias, Russia's propaganda, international news,

Assumptions, Concepts, and Examples

Workflow, demystified

Raw texts → **Matrix representation** → **Analytics**

Fellow-Citizens of the Senate and of the House of Representatives: Among the vicissitudes incident to life..

Fellow citizens, I am again called upon by the voice of my country to execute the functions of its Chief Magistrate. When the occasion

When it was first perceived, in early times, that no middle course for America remained between unlimited submission to a foreign legislature

- tokenization
- feature selection

column 0: rownames	fellow-citizens	of	the	senate	and
1789-Washington	1	71	116	1	48
1793-Washington	0	11	13	0	2
1797-Adams	3	140	163	1	130
1801-Jefferson	2	104	130	0	81
1805-Jefferson	0	101	143	0	93
1809-Madison	1	69	104	0	43
1813-Madison	1	65	100	0	44
1817-Monroe	5	164	275	0	122
1821-Monroe	1	197	360	0	141
1825-Adams	0	245	304	0	116
1829-Jackson	0	71	92	0	49
1833-Jackson	0	76	101	0	53
1837-VanBuren	0	198	252	0	150
1841-Harrison	11	604	829	5	231
1845-Polk	1	298	397	0	189

Statistics:

- Term frequencies
- Keyness
- Readability
- Lexical diversity
- Similarity, distance

Models

- Supervised ML
- Unsupervised ML
- Scaling
- "Word embeddings"
- Topic models

Plots

Keyness, networks, scaling, word clouds, "x-ray"

Basic concepts and terminology

- Texts: Organized into *documents*
- Corpus: Collection of texts, often with associated document-level metadata, what I will call "document variables" or *docvars*
- Stems: words with suffixes removed (using a set of rules)
- Lemmas: canonical word form

Word	**win**	**winning**	**wins**	**won**
Stem	win	win	win	*won*
Lemma	win	win	win	*win*

Basic concepts and terminology (cont.)

- Stop words: words that are designed for exclusion from any analysis of text

Basic concepts and terminology (cont.)

- Stop words: words that are designed for exclusion from any analysis of text
- Parts of speech: linguistic markers indicating the general category of a word's inguistic property, e.g. *noun*, *verb*, *adjective*, etc.

Basic concepts and terminology (cont.)

- Stop words: words that are designed for exclusion from any analysis of text
- Parts of speech: linguistic markers indicating the general category of a word's inguistic property, e.g. *noun*, *verb*, *adjective*, etc.
- Named entities: a real-world object, such as persons, locations, organizations, products, etc., that can be denoted with a proper name, often a phrase, e.g. "Australian Society for Quantitative Political Science"

Basic concepts and terminology (cont.)

- Stop words: words that are designed for exclusion from any analysis of text
- Parts of speech: linguistic markers indicating the general category of a word's inguistic property, e.g. *noun*, *verb*, *adjective*, etc.
- Named entities: a real-world object, such as persons, locations, organizations, products, etc., that can be denoted with a proper name, often a phrase, e.g. "Australian Society for Quantitative Political Science"
- Multi-word expressions: sequences of words denoting a single concept (and would be in German), e.g. *value added tax* (in German: *Mehrwertsteuer*)

Types and tokens

- tokens: a sequence of characters that are grouped together as a useful semantic unit
  - words
  - could also include puncutation characters or symbols
  - stems or lemmas
  - multi-word expressions
  - named entities
  - usually, but not always, delimited by spaces
- type: a unique token

```
toks <- tokens("Of all tax, income taxes are worst.", remove_punct = TRUE)
toks
```

```
## tokens from 1 document.
## text1 :
## [1] "Of"     "all"    "tax"    "income" "taxes"  "are"    "worst"
```

```
tokens_wordstem(toks)
```

```
## tokens from 1 document.
## text1 :
## [1] "Of"    "all"   "tax"   "incom" "tax"   "are"   "worst"
```

```
tokens_wordstem(toks) %>%
  tokens_remove(stopwords("english"))
```

```
## tokens from 1 document.
## text1 :
## [1] "tax"   "incom" "tax"   "worst"
```

```
tokens_wordstem(toks) %>%
  tokens_remove(stopwords("english")) %>%
  types()
```

```
## [1] "tax"   "incom" "worst"
```

Typical workflow steps

1. Lowercase
    - "**a** corpus is a set of documents."
    - "**t**his is the second document in the corpus."

Typical workflow steps

1. Lowercase
    - "**a** corpus is a set of documents."
    - "**t**his is the second document in the corpus."
2. Remove stopwords and punctuation
    - "~~a~~ corpus ~~is~~ ~~a~~ set ~~of~~ documents."
    - "~~this~~ ~~is~~ ~~the~~ second document ~~in~~ ~~the~~ corpus."

Typical workflow steps

1. Lowercase
    - "**a** corpus is a set of documents."
    - "**t**his is the second document in the corpus."
2. Remove stopwords and punctuation
    - "~~a~~ corpus ~~is~~ ~~a~~ set ~~of~~ documents."
    - "~~this~~ ~~is~~ ~~the~~ second document ~~in~~ ~~the~~ corpus."
3. Stem
    - "corpus set document~~s~~"
    - "second document corpus"
4. Tokenize
    - [corpus, set, document]
    - [second, document, corpus]

Typical workflow steps

1. Lowercase
    - "**a** corpus is a set of documents."
    - "**t**his is the second document in the corpus."
2. Remove stopwords and punctuation
    - "~~a~~ corpus ~~is~~ ~~a~~ set ~~of~~ documents."
    - "~~this~~ ~~is~~ ~~the~~ second document ~~in~~ ~~the~~ corpus."
3. Stem
    - "corpus set document~~s~~"
    - "second document corpus"
4. Tokenize
    - [corpus, set, document]
    - [second, document, corpus]
5. Create a document-feature matrix
    - a "bag-of-words" conversion of documents into a matrix that counts the features (types) by document

Document-feature matrix

Document 1: "*A corpus is a set of documents.*"
Document 2: "*This is the second document in the corpus.*"

	corpus	**set**	**document**	**second**
Document 1	1	1	1	0
Document 2	1	0	1	1
…				
Document *n*	0	1	1	0

Getting started

Installing **quanteda**

Install the **quanteda** package from CRAN:

```
install.packages("quanteda")
```

You should also install the **readtext** package:

```
install.packages("readtext")
```

Afterwards, load both packages:

```
library("quanteda")
library("readtext")
packageVersion("quanteda")
```

```
## [1] '1.4.0'
```

```
packageVersion("readtext")
```

```
## [1] '0.72'
```

Reproduce example in **quanteda**

Create a text corpus

```
library(quanteda)
# Create text corpus
mycorp <- corpus(c("A corpus is a set of documents.",
                   "This is the second document in the corpus."))
```

Exercise: Create this corpus and get a summary using `summary(mycorp)`.

Reproduce example in **quanteda**

Create a text corpus

```
library(quanteda)
# Create text corpus
mycorp <- corpus(c("A corpus is a set of documents.",
                   "This is the second document in the corpus."))
```

Exercise: Create this corpus and get a summary using `summary(mycorp)`.

```
summary(mycorp)
```

```
## Corpus consisting of 2 documents, showing 100 documents:
##
##   Text Types Tokens Sentences
##  text1     8      8         1
##  text2     8      9         1
```

Reproduce example in quanteda

Create a document-feature matrix

```
dfm(mycorp, remove_punct = TRUE,
    remove = stopwords("en"), stem = TRUE)
```

```
## Document-feature matrix of: 2 documents, 4 features (25.0% sparse).
## 2 x 4 sparse Matrix of class "dfm"
##        features
## docs    corpus set document second
##   text1      1   1        1      0
##   text2      1   0        1      1
```

Question: How does the dfm change when we change the preprocessing steps?

The quanteda Infrastructure

quanteda: Quantitative Analysis of Textual Data

- 6 years of development, 25 releases
- 7,700 commits, 190,000 downloads

Design of the package

- consistent grammar
- flexible for power users, simple for beginners
- analytic transparency and reproducibility
- compability with other packages
- pipelined workflow using **magrittr**'s %>%

Quanteda Initiative

- UK non-profit organization devoted to the promotion of open-source text analysis software
- User, technical and development support
- Teaching and workshops

Additional packages

For some exercises, we will need **quanteda.corpora**:

```
install.packages("devtools")
devtools::install_github("quanteda/quanteda.corpora")
```

For POS tagging, entity recognition, and dependency parsing, you should install **spacyr** (not covered extensively).

```
install.packages("spacyr")
```

Installation instructions: http://spacyr.quanteda.io

Course Resources

- Documentation:
  - https://quanteda.io
  - https://readtext.quanteda.io
  - https://spacyr.quanteda.org
- Tutorials: https://tutorials.quanteda.io
- Cheatsheet: https://www.rstudio.com/resources/cheatsheets/
- Kenneth Benoit, Kohei Watanabe, Haiyan Wang, Paul Nulty, Adam Obeng, Stefan Müller, and Akitaka Matsuo. 2018. "quanteda: An R Package for the Quantitative Analysis of Textual Data." *Journal of Open Source Software* 3(30): 774.

Recap: Workflow

1. **Corpus**
    - Saves character strings and variables in a data frame
    - Combines texts with document-level variables
2. **Tokens**
    - Stores tokens in a list of vectors
    - Positional (string-of-words) analysis is performed using and `textstat_collocations()`, `tokens_ngrams()` and `tokens_select()` or `fcm()` with window option
3. **Document-feature matrix (DFM)**
    - Represents frequencies of features in documents in a matrix
    - The most efficient structure, but it does not have information on positions of words
    - Non-positional (bag-of-words) analysis are performed using many of the `textstat_*` and `textmodel_*` functions

Main function classes

- Text corpus: `corpus()`
- Tokenization: `tokens()`
- Document-feature matrix: `dfm()`
- Feature co-occurence matrix: `fcm()`
- Text statistics: `textstat_()`
- Text models: `textmodel_()`
- Plots: `textplot_()`

Useful RStudio keyboard shortcuts

- Insert pipe operator (%>%): Shift + Cmd/Cntrl + M
- Insert assignment operator (<-): Alt + -

More shortcuts for RStudio can be found here

Clarification

The following expressions result in the same output

```
data_corpus_inaugural %>%
    tokens()

tokens(data_corpus_inaugural)
```

Corpus

Corpus functions in **quanteda**

- `corpus()`
- `corpus_subset()`
- `corpus_reshape()`
- `corpus_segment()`
- `corpus_sample()`

Corpora in **quanteda**

- `data_corpus_inaugural`
- `data_corpus_irishbudget2010`
- Additional corpora in the **quanteda.corpora** package

Using **magrittr**'s pipe

```
data_corpus_inaugural %>%
    corpus_subset(President == "Obama") %>%
    ndoc()
```

```
## [1] 2
```

```
data_corpus_inaugural %>%
    corpus_subset(President == "Obama") %>%
    corpus_reshape(to = "sentences") %>%
    ndoc()
```

```
## [1] 198
```

Access number of types and tokens of corpus

```
ntype(data_corpus_inaugural) %>%
    head()
```

```
## 1789-Washington 1793-Washington      1797-Adams  1801-Jefferson
##             625              96             826             717
##  1805-Jefferson    1809-Madison
##             804             535
```

```
ntoken(data_corpus_inaugural) %>%
    head()
```

```
## 1789-Washington 1793-Washington      1797-Adams  1801-Jefferson
##            1538             147            2578            1927
##  1805-Jefferson    1809-Madison
##            2381            1263
```

Overview of document-level variables

```
head(docvars(data_corpus_inaugural))
```

```
##   Year  President FirstName
## 1 1789 Washington    George
## 2 1793 Washington    George
## 3 1797      Adams      John
## 4 1801  Jefferson    Thomas
## 5 1805  Jefferson    Thomas
## 6 1809    Madison     James
```

Exercise

1. Based on `data_corpus_inaugural`, create an object `data_corpus_postwar` (speeches since 1945).
2. What speech has most tokens? What speech has most types?

Note: You can find the documentation and examples using `?` followed by the name of the function.

Solution

```
data_corpus_postwar <- data_corpus_inaugural %>%
    corpus_subset(Year > 1945)

# number of tokens per speech
data_corpus_postwar %>%
    ntoken() %>%
    sort(decreasing = TRUE) %>%
    head()
```

```
##     1985-Reagan     1981-Reagan 1953-Eisenhower      2009-Obama
##            2921            2790            2757            2711
##       1989-Bush     1949-Truman
##            2681            2513
```

```
# number of types per speech
data_corpus_postwar %>%
    ntype() %>%
    sort(decreasing = TRUE) %>%
    head()
```

```
##      2009-Obama     1985-Reagan     1981-Reagan 1953-Eisenhower
##             938             925             902             900
##      2013-Obama       1989-Bush
##             814             795
```

Adding document-level variables

```
# new docvar: PresidentFull
docvars(data_corpus_inaugural, "Order") <- 1:ndoc(data_corpus_inaugural)

head(docvars(data_corpus_inaugural, "Order"))
```

```
## [1] 1 2 3 4 5 6
```

```
# new docvar: PresidentFull
docvars(data_corpus_inaugural, "PresidentFull") <-
    paste(docvars(data_corpus_inaugural, "FirstName"),
          docvars(data_corpus_inaugural, "President"),
          sep = " ")

head(docvars(data_corpus_inaugural))
```

```
##   Year  President FirstName Order     PresidentFull
## 1 1789 Washington    George     1 George Washington
## 2 1793 Washington    George     2 George Washington
## 3 1797      Adams      John     3        John Adams
## 4 1801  Jefferson    Thomas     4  Thomas Jefferson
## 5 1805  Jefferson    Thomas     5  Thomas Jefferson
## 6 1809    Madison     James     6     James Madison
```

Exercise

1. Use `data_corpus_inaugural` and reshape the entire corpus to the level of sentences and store the new corpus. What is the number of documents of the reshaped corpus?
2. Add a document-level variable to the reshaped corpus that counts the tokens per *sentence*.
3. Keep only sentences that are longer than 10 words.
4. Reshape the corpus back to the level of documents and store the corpus as `data_corpus_inaugural_subset`.
5. Optional: find a more efficient solution.

Solution

```
corp_sentences <- corpus_reshape(data_corpus_inaugural, to = "sentences")

docvars(corp_sentences, "number_tokens") <- ntoken(corp_sentences,
                                                   remove_punct = TRUE)

ndoc(corp_sentences)
```

```
## [1] 5016
```

```
corp_sentence_subset <- corp_sentences %>%
    corpus_subset(number_tokens > 10)
ndoc(corp_sentence_subset)
```

```
## [1] 4267
```

```
data_corpus_inaugural_subset <- corp_sentence_subset %>%
    corpus_reshape(to = "documents")

sum(ntoken(data_corpus_inaugural))
```

```
## [1] 149145
```

```
sum(ntoken(data_corpus_inaugural_subset))
```

Keywords-in-context

Keywords-in-context

Keywords-in-context (`kwic()`) returns a list of one or more keywords and its immediate context.

```
mykwic <- kwic(data_corpus_inaugural, pattern = "security",
     window = 3)

head(mykwic, 3)
```

```
##
##  [1789-Washington, 1497] government for the | security |
##      [1813-Madison, 321]       seas and the | security |
##      [1817-Monroe, 1610]      may form some | security |
##
##  of their union
##  of an important
##  against these dangers
```

```
# Check number of occurrences
nrow(mykwic)
```

```
## [1] 65
```

KWIC with multiword expressions

Use `phrase()` to look up multiword expressions

```
kwic(data_corpus_inaugural, pattern = phrase("United States"),
     window = 3) %>%
    head(3)
```

```
##
##  [1789-Washington, 434:435]       people of the | United States |
##  [1789-Washington, 530:531]        those of the | United States |
##       [1797-Adams, 525:526] Constitution of the | United States |
##
##  a Government instituted
##  . Every step
##  in a foreign
```

Exercise

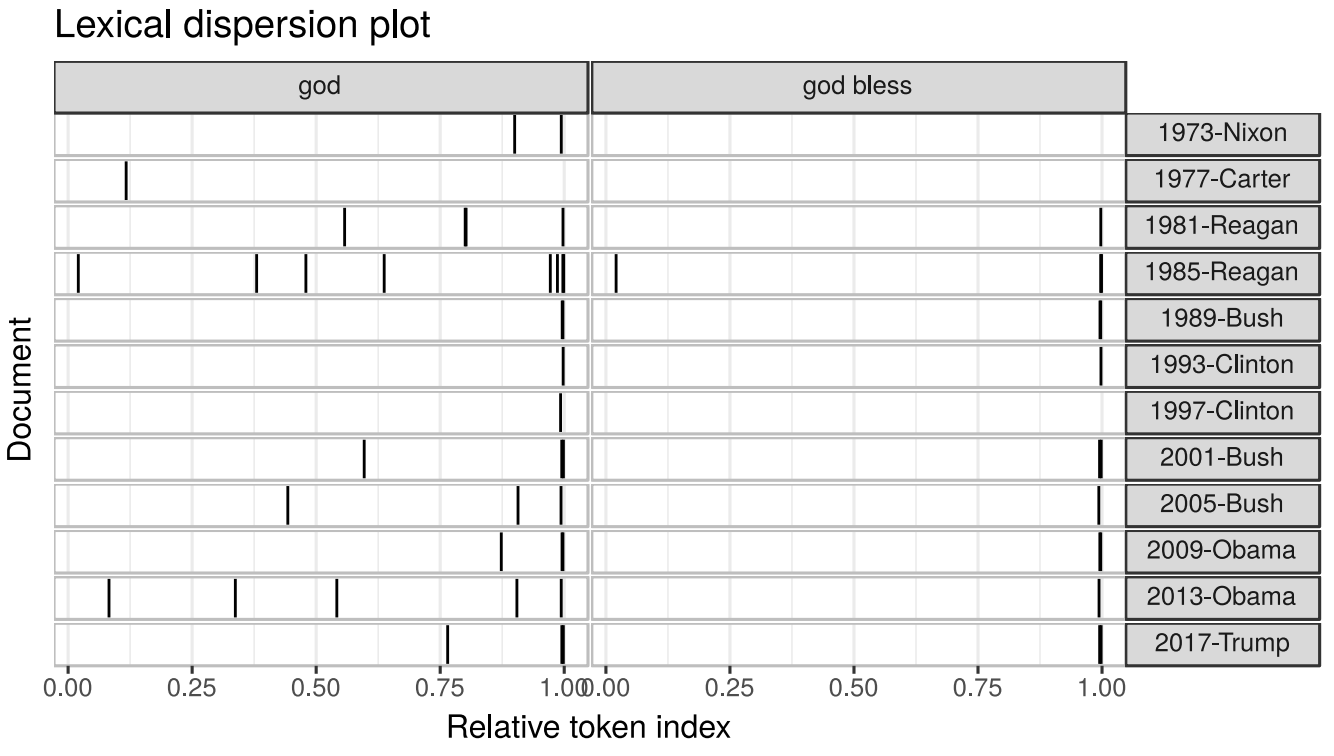
1. In what context were"God" and "God bless" used in US presidential inaugural speeches after 1970?
2. Look up keywords-in-context for `god`, `god bless` (wrapping the latter in `phrase()`) in one `kwic()` call.
3. Send the results of the kwic output to `textplot_xray()`.

Solution

```
mykwic1 <- corpus_subset(data_corpus_inaugural, Year > 1970) %>%
  kwic("god", window = 3)
mykwic2 <- corpus_subset(data_corpus_inaugural, Year > 1970) %>%
  kwic(phrase("god bless"), window = 3)

tail(mykwic2)
```

```
##
##   [2005-Bush, 2304:2305] freedom. May | God bless | you, and
##  [2009-Obama, 2699:2700]   Thank you. | God bless | you. And
##  [2009-Obama, 2704:2705]     you. And | God bless | the United States
##  [2013-Obama, 2303:2304]   Thank you. | God bless | you, and
##  [2017-Trump, 1652:1653]   Thank you, | God bless | you, and
##  [2017-Trump, 1657:1658]     you, and | God bless | America.
```

```
textplot_xray(mykwic1, mykwic2)
```



Tokenization

Tokenization of a corpus

```
immig_corp <- corpus(data_char_ukimmig2010)
toks <- tokens(immig_corp)
```

Tokenization of a corpus

```
immig_corp <- corpus(data_char_ukimmig2010)
toks <- tokens(immig_corp)
```

```
head(toks[[1]], 20)
```

```
##  [1] "IMMIGRATION"  ":"            "AN"           "UNPARALLELED"
##  [5] "CRISIS"       "WHICH"        "ONLY"         "THE"
##  [9] "BNP"          "CAN"          "SOLVE"        "."
## [13] "-"            "At"           "current"      "immigration"
## [17] "and"          "birth"        "rates"        ","
```

Remove punctuation

```
nopunct_toks <- tokens(data_char_ukimmig2010,
                       remove_punct = TRUE)
```

Remove punctuation

```
nopunct_toks <- tokens(data_char_ukimmig2010,
                       remove_punct = TRUE)
```

```
head(nopunct_toks[[1]], 20)
```

```
##  [1] "IMMIGRATION"  "AN"           "UNPARALLELED" "CRISIS"
##  [5] "WHICH"        "ONLY"         "THE"          "BNP"
##  [9] "CAN"          "SOLVE"        "At"           "current"
## [13] "immigration"  "and"          "birth"        "rates"
## [17] "indigenous"   "British"      "people"       "are"
```

Remove stopwords

```
nostop_toks <- tokens_select(toks, stopwords("en"),
                             selection = "remove")
# Note: equivalent to
nostop_toks <- tokens_remove(toks, stopwords("en"))
```

Remove stopwords

```
nostop_toks <- tokens_select(toks, stopwords("en"),
                             selection = "remove")
# Note: equivalent to
nostop_toks <- tokens_remove(toks, stopwords("en"))
```

```
head(nostop_toks[[1]], 20)
```

```
##  [1] "IMMIGRATION"  ":"            "UNPARALLELED" "CRISIS"
##  [5] "BNP"          "CAN"          "SOLVE"        "."
##  [9] "-"            "current"      "immigration"  "birth"
## [13] "rates"        ","            "indigenous"   "British"
## [17] "people"       "set"          "become"       "minority"
```

Customize stopword list

- Stopwords for other languges: check the **stopwords** package
- Remove feature from stopword list

```
"will" %in% stopwords("en")
```

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

```
my_stopwords_en <- stopwords("en")[!stopwords("en") %in% c("will")]

"will" %in% my_stopwords_en
```

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

```
head(toks[[1]], 20)
```

```
##  [1] "IMMIGRATION"  ":"            "AN"           "UNPARALLELED"
##  [5] "CRISIS"       "WHICH"        "ONLY"         "THE"
##  [9] "BNP"          "CAN"          "SOLVE"        "."
## [13] "-"            "At"           "current"      "immigration"
## [17] "and"          "birth"        "rates"        ","
```

```
head(nopunct_toks[[1]], 20)
```

```
##  [1] "IMMIGRATION"  "AN"           "UNPARALLELED" "CRISIS"
##  [5] "WHICH"        "ONLY"         "THE"          "BNP"
##  [9] "CAN"          "SOLVE"        "At"           "current"
## [13] "immigration"  "and"          "birth"        "rates"
## [17] "indigenous"   "British"      "people"       "are"
```

```
head(nostop_toks[[1]], 20)
```

```
##  [1] "IMMIGRATION"  ":"            "UNPARALLELED" "CRISIS"
##  [5] "BNP"          "CAN"          "SOLVE"        "."
##  [9] "-"            "current"      "immigration"  "birth"
## [13] "rates"        ","            "indigenous"   "British"
## [17] "people"       "set"          "become"       "minority"
```

Select certain terms

```
immig_toks <- tokens_select(toks, c("immig*", "migra*"),
                            padding = TRUE)
head(immig_toks[[1]], 20)
```

```
##  [1] "IMMIGRATION" ""            ""            ""            ""
##  [6] ""            ""            ""            ""            ""
## [11] ""            ""            ""            ""            ""
## [16] "immigration" ""            ""            ""            ""
```

Select certain terms

```
immig_toks <- tokens_select(toks, c("immig*", "migra*"),
                            padding = TRUE)
head(immig_toks[[1]], 20)
```

```
##  [1] "IMMIGRATION" ""            ""            ""            ""
##  [6] ""            ""            ""            ""            ""
## [11] ""            ""            ""            ""            ""
## [16] "immigration" ""            ""            ""            ""
```

```
immig_toks_no_pad <- tokens_select(toks, c("immig*", "migra*"),
                            padding = FALSE)
head(immig_toks_no_pad[[1]], 20)
```

```
##  [1] "IMMIGRATION" "immigration" "immigration" "immigrants"  "immigration"
##  [6] "immigration" "immigration" "immigrants"  "immigrant"   "immigrant"
## [11] "immigrants"  "immigrants"  "immigration" "Immigration" "immigration"
## [16] "immigrants"  "Immigration" "Immigration" "Immigration" "immigrants"
```

Select certain terms and their context

```
# specify number of surrounding words using window
window_toks <- tokens_select(toks, c('immig*', 'migra*'),
                             padding = TRUE, window = 5)
head(window_toks[[1]], 20)
```

```
##  [1] "IMMIGRATION"  ":"            "AN"           "UNPARALLELED"
##  [5] "CRISIS"       "WHICH"        ""             ""
##  [9] ""             ""             "SOLVE"        "."
## [13] "-"            "At"           "current"      "immigration"
## [17] "and"          "birth"        "rates"        ","
```

Compound multiwords expressions

```
multi_kw <- kwic(toks, phrase(c('asylum seeker*',
                                'british citizen*')))
head(multi_kw, 5)
```

```
##
##  [BNP, 1724:1725]        the honour and benefit of | British citizenship |
##  [BNP, 1958:1959] all illegal immigrants and bogus |   asylum seekers    |
##  [BNP, 2159:2160]            region concerned. An' |    asylum seeker    |
##  [BNP, 2192:2193]          country. Because every' |    asylum seeker    |
##  [BNP, 2218:2219]     there are currently no legal |   asylum seekers    |
##
##  has gone to people who
##  , including their dependents.
##  ' who has crossed dozens
##  ' in Britain has crossed
##  in Britain today. It
```

Compound multiwords expressions

Preserve these expressions in bag-of-word analysis:

```
comp_toks <- tokens_compound(toks, phrase(c('asylum seeker*',
                                            'british citizen*')))

comp_kw <- kwic(comp_toks, c('asylum_seeker*', 'british_citizen*'))
head(comp_kw, 4)
```

```
##
##  [BNP, 1724]        the honour and benefit of | British_citizenship |
##  [BNP, 1957] all illegal immigrants and bogus |   asylum_seekers    |
##  [BNP, 2157]            region concerned. An' |    asylum_seeker    |
##  [BNP, 2189]          country. Because every' |    asylum_seeker    |
##
##  has gone to people who
##  , including their dependents.
##  ' who has crossed dozens
##  ' in Britain has crossed
```

Exercise

1. Tokenize `data_corpus_irishbudget2010` and compound the following party names: `fianna fáil`, `fine gael`, and `sinn féin`.
2. Select only the three party names and the window of +-10 words
3. How can we extract only the full *sentences* in which at least one of the parties is mentioned?

Solution

```
# 1. Tokenize `data_corpus_irishbudget2010` and compound party names

toks_ire <- data_corpus_irishbudget2010 %>%
    tokens() %>%
    tokens_compound(phrase(c("fianna fáil", "fine gael", "sinn féin")))

nrow(kwic(toks_ire, "fianna_fáil"))
```

```
## [1] 66
```

```
nrow(kwic(toks_ire, phrase("fianna fáil")))
```

```
## [1] 0
```

```
# 2. Select only the three party names and the window of +-10 words
toks_ire_select <- toks_ire %>%
    tokens_keep(c("fianna_fáil", "fine_gael", "sinn_féin"), window = 10)

head(toks_ire_select[[2]], 22)
```

```
##  [1] "happening"   "today"       "."           "It"          "is"
##  [6] "happening"   ","           "however"     ","           "because"
## [11] "Fianna_Fáil" "failed"      "to"          "heed"        "the"
## [16] "warnings"    "and"         "drove"       "this"        "economy"
```

```r
# 3. Extract only full sentences in which
# at least one party is mentioned

toks_ire_sentences <- data_corpus_irishbudget2010 %>%
    corpus_reshape(to = "sentences") %>% # <-- important step!
    tokens() %>%
    tokens_compound(phrase(c("fianna fáil", "fine gael", "sinn féin"))) %>%
    tokens_keep(c("fianna_fáil", "fine_gael", "sinn_féin"),
                window = 1000)
```

N-grams and skipgrams

```
# Unigrams
tokens("insurgents killed in ongoing fighting")
```

```
## tokens from 1 document.
## text1 :
## [1] "insurgents" "killed"     "in"         "ongoing"    "fighting"
```

N-grams and skipgrams

```
# Unigrams
tokens("insurgents killed in ongoing fighting")
```

```
## tokens from 1 document.
## text1 :
## [1] "insurgents" "killed"     "in"         "ongoing"    "fighting"
```

```
# Bigrams
tokens("insurgents killed in ongoing fighting") %>%
    tokens_ngrams(n = 2)
```

```
## tokens from 1 document.
## text1 :
## [1] "insurgents_killed" "killed_in"         "in_ongoing"
## [4] "ongoing_fighting"
```

```
# Skipgrams
tokens("insurgents killed in ongoing fighting") %>%
    tokens_skipgrams(n = 2, skip = 0:1)
```

```
## tokens from 1 document.
## text1 :
## [1] "insurgents_killed" "insurgents_in"     "killed_in"
## [4] "killed_ongoing"    "in_ongoing"        "in_fighting"
```

Look up tokens from a dictionary

```
toks <- tokens(data_char_ukimmig2010)

dict <- dictionary(list(refugee = c('refugee*', 'asylum*'),
                        worker = c('worker*', 'employee*')))
print(dict)
```

```
## Dictionary object with 2 key entries.
## - [refugee]:
##   - refugee*, asylum*
## - [worker]:
##   - worker*, employee*
```

```
dict_toks <- tokens_lookup(toks, dict)
head(dict_toks, 2)
```

```
## tokens from 2 documents.
## BNP :
##  [1] "refugee" "worker"  "refugee" "refugee" "refugee" "refugee" "refugee"
##  [8] "refugee" "refugee" "refugee" "refugee" "refugee" "refugee" "worker"
##
## Coalition :
## [1] "refugee"
```

The transition from tokens() to dfm()

```
dfm(dict_toks)
```

```
## Document-feature matrix of: 9 documents, 2 features (38.9% sparse).
## 9 x 2 sparse Matrix of class "dfm"
##                features
## docs            refugee worker
##   BNP                12      2
##   Coalition           1      0
##   Conservative        0      0
##   Greens              4      1
##   Labour              4      4
##   LibDem              6      0
##   PC                  3      0
##   SNP                 1      0
##   UKIP                6      0
```

Summary of tokens functions

- `tokens()`
- `tokens_tolower()`/`tokens_toupper()`
- `tokens_wordstem()`
- `tokens_compound()`
- `tokens_lookup()`
- `tokens_ngrams()`
- `tokens_skipgrams()`
- `tokens_select()`/`tokens_remove()`/`tokens_keep()`
- `tokens_replace()`
- `tokens_sample()`
- `tokens_subset()`

Recall to use `?` to read the manual and examples for each function.

Exercise

1. Tokenize `data_corpus_irishbudget2010`
2. Convert the tokens object, remove punctuation, change to lowercase, remove stopwords, and stem
3. Get the number of types and tokens per speech

Solution

```
ire_toks <- data_corpus_irishbudget2010 %>%
    tokens(remove_punct = TRUE) %>%
    tokens_remove(stopwords("en")) %>%
    tokens_tolower() %>%
    tokens_wordstem()

ire_ntype <- ntype(ire_toks)
ire_ntoken <- ntoken(ire_toks)
```

Exercise

1. Create a document-feature matrix from the tokens object above
2. Get the 50 most frequent terms using `topfeatures()`
3. Rerun step 5, but change the preprocessing steps

Solution

```
ire_dfm <- dfm(ire_toks) # dfm() transforms to lower case by default

topfeatures(ire_dfm)
```

```
##   peopl  budget  govern  minist    year     tax  public economi     cut
##     273     272     271     204     201     195     179     172     172
##     job
##     148
```

```
## alternative approach without tokens() -- less control!
ire_dfm2 <- data_corpus_irishbudget2010 %>%
    dfm(remove_punct = TRUE,
        remove = stopwords("en"),
        stem = TRUE)

topfeatures(ire_dfm2)
```

```
##   peopl  budget  govern  minist    year     tax  public economi     cut
##     273     272     271     204     201     195     179     172     172
##     job
##     148
```

Select features based on frequencies

```
nfeat(ire_dfm)
```

```
## [1] 3460
```

```
ire_dfm_trim1 <- dfm_trim(ire_dfm,
                          min_termfreq = 5)
nfeat(ire_dfm_trim1)
```

```
## [1] 1040
```

Select features based on frequencies

```
nfeat(ire_dfm)
```

```
## [1] 3460
```

```
ire_dfm_trim1 <- dfm_trim(ire_dfm,
                          min_termfreq = 5)
nfeat(ire_dfm_trim1)
```

```
## [1] 1040
```

```
ire_dfm_trim2 <- dfm_trim(ire_dfm,
                          min_docfreq = 5)
nfeat(ire_dfm_trim2)
```

```
## [1] 725
```

Select features based on frequencies

```
nfeat(ire_dfm)
```

```
## [1] 3460
```

```
ire_dfm_trim1 <- dfm_trim(ire_dfm,
                          min_termfreq = 5)
nfeat(ire_dfm_trim1)
```

```
## [1] 1040
```

```
ire_dfm_trim2 <- dfm_trim(ire_dfm,
                          min_docfreq = 5)
nfeat(ire_dfm_trim2)
```

```
## [1] 725
```

```
ire_dfm_trim3 <- dfm_trim(ire_dfm,
                          max_docfreq = 0.1,
                          docfreq_type = "prop")
nfeat(ire_dfm_trim3)
```

```
## [1] 1637
```

Exercise

1. Create a dfm from the two documents below, and weight it using `dfm_weight()`.
2. For `dfm_weight()` try out the `scheme` arguments "count", "boolean" and "prop".
3. What are advantages and problems of each weighting scheme?

```
texts <- c("apple is better than banana",
           "banana banana apple much better")
```

Solution

```
texts_dfm <- dfm(texts)
dfm_weight(texts_dfm, scheme = "count")
```

```
## Document-feature matrix of: 2 documents, 6 features (25.0% sparse).
## 2 x 6 sparse Matrix of class "dfm"
##        features
## docs    apple is better than banana much
##   text1     1  1      1    1      1    0
##   text2     1  0      1    0      2    1
```

```
dfm_weight(texts_dfm, scheme = "boolean")
```

```
## Document-feature matrix of: 2 documents, 6 features (25.0% sparse).
## 2 x 6 sparse Matrix of class "dfm"
##        features
## docs    apple is better than banana much
##   text1     1  1      1    1      1    0
##   text2     1  0      1    0      1    1
```

```
dfm_weight(texts_dfm, scheme = "prop")
```

```
## Document-feature matrix of: 2 documents, 6 features (25.0% sparse).
## 2 x 6 sparse Matrix of class "dfm"
##        features
## docs    apple  is better than banana much
```

Sentiment analysis

Sentiment analysis

Sentiment analysis using the Lexicoder Sentiment Dictionary (`data_dictionary_LSD2015`)

```
summary(data_dictionary_LSD2015)
```

```
##              Length Class  Mode
## negative     2858   -none- character
## positive     1709   -none- character
## neg_positive 1721   -none- character
## neg_negative 2860   -none- character
```

```
docvars(data_corpus_irishbudget2010, "gov_opp") <-
    ifelse(docvars(data_corpus_irishbudget2010, "party") %in%
               c("FF", "Green"),
           "Government", "Opposition")

# tokenize and apply dictionary
dict_toks <- data_corpus_irishbudget2010 %>%
    tokens() %>%
    tokens_lookup(dictionary = data_dictionary_LSD2015)

# transform to a dfm
dict_dfm <- dfm(dict_toks)
```

```
print(dict_dfm)
```

```
## Document-feature matrix of: 14 documents, 4 features (12.5% sparse).
## 14 x 4 sparse Matrix of class "dfm"
##                            features
## docs                        negative positive neg_positive neg_negative
##   Lenihan, Brian (FF)            188      397            2            1
##   Bruton, Richard (FG)           163      147            5            2
##   Burton, Joan (LAB)             225      266            8            3
##   Morgan, Arthur (SF)            260      249            5            2
##   Cowen, Brian (FF)              150      368            1            2
##   Kenny, Enda (FG)               104      146            3            3
##   ODonnell, Kieran (FG)           49       84            4            0
##   Gilmore, Eamon (LAB)           164      176            3            0
##   Higgins, Michael (LAB)          37       42            1            0
##   Quinn, Ruairi (LAB)             34       40            1            1
##   Gormley, John (Green)           17       56            0            0
##   Ryan, Eamon (Green)             24       78            1            2
##   Cuffe, Ciaran (Green)           38       56            0            0
##   OCaolain, Caoimhghin (SF)      154      145            1            1
```

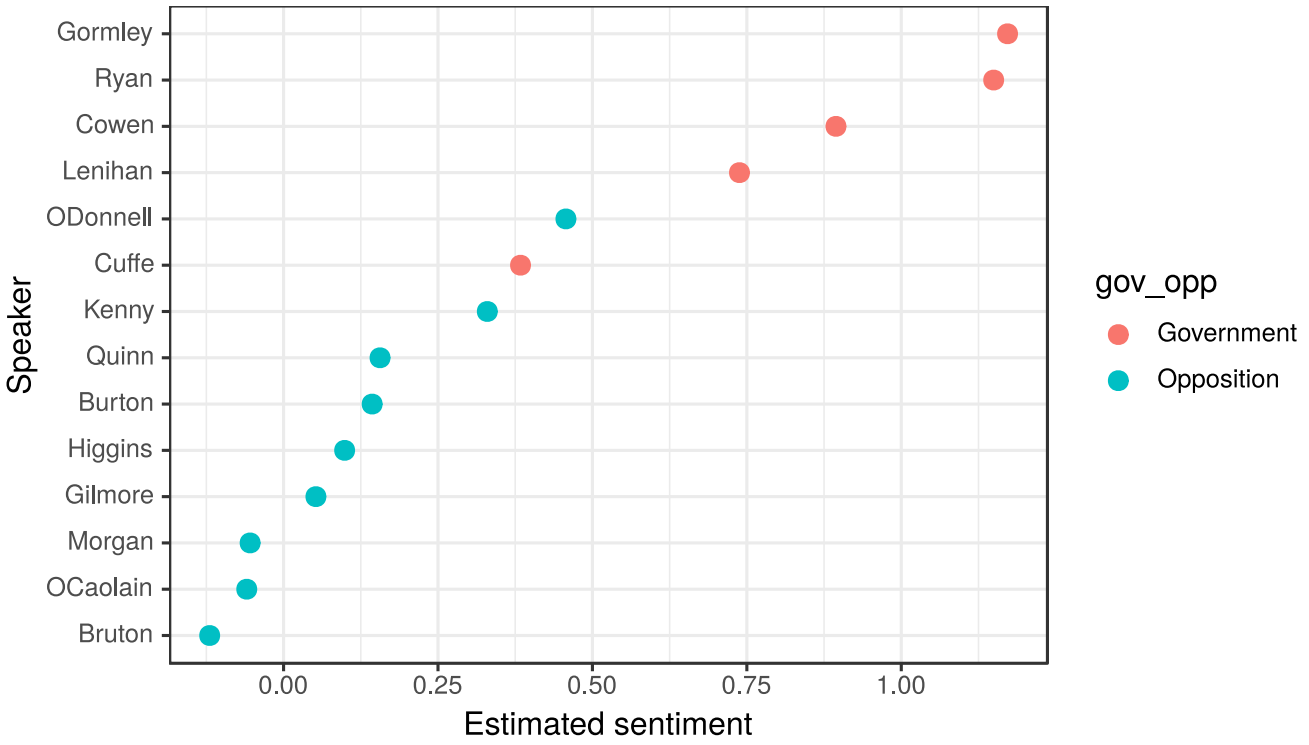
Convert to a data frame using `convert()`

```
dict_output <- convert(dict_dfm, to = "data.frame")
```

Estimate sentiment

```
dict_output$sent_score <- log((dict_output$positive +
                                   dict_output$neg_negative + 0.5) /
                              (dict_output$negative +
                                   dict_output$neg_positive+ 0.5))

dict_output <- cbind(dict_output, docvars(data_corpus_irishbudget2010))

library(ggplot2)
plot_sent_ire <- ggplot(dict_output,
                        aes(x = reorder(name, sent_score),
                        y = sent_score,
                        colour = gov_opp)) +
    geom_point(size = 3) +
    coord_flip() +
    labs(x = "Speaker", y = "Estimated sentiment")
```

```
plot_sent_ire
```

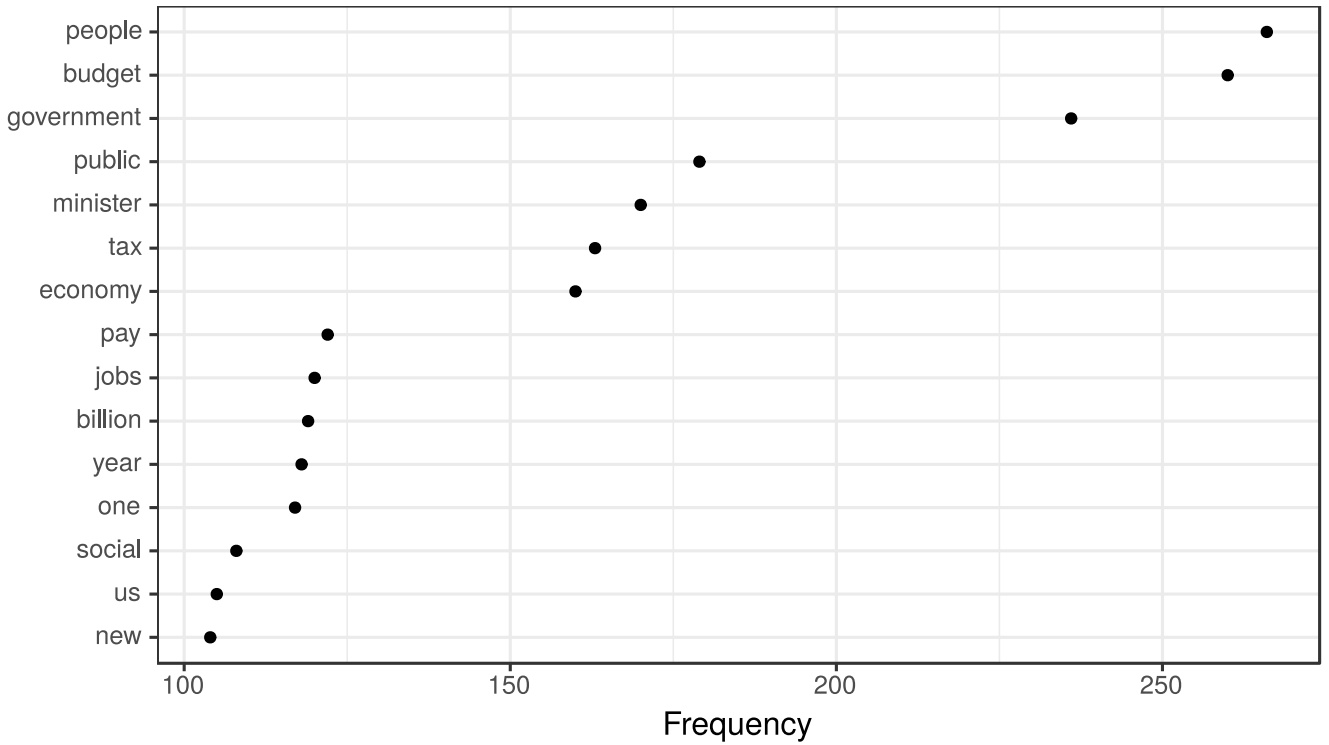


Textual statistics

Simple frequency analysis

```
ire_dfm <- data_corpus_irishbudget2010 %>%
    dfm(remove = stopwords("en"),
        remove_punct = TRUE)

ire_freqplot <- ire_dfm %>%
    textstat_frequency(n = 15) %>%
    ggplot(aes(x = reorder(feature, frequency),
               y = frequency)) +
    geom_point() +
    coord_flip() +
    labs(x = NULL, y = "Frequency")
```

```
ire_freqplot
```



Frequency analysis for groups

```
ire_dfm_group <- ire_dfm %>%
    dfm_group(groups = "party")
```

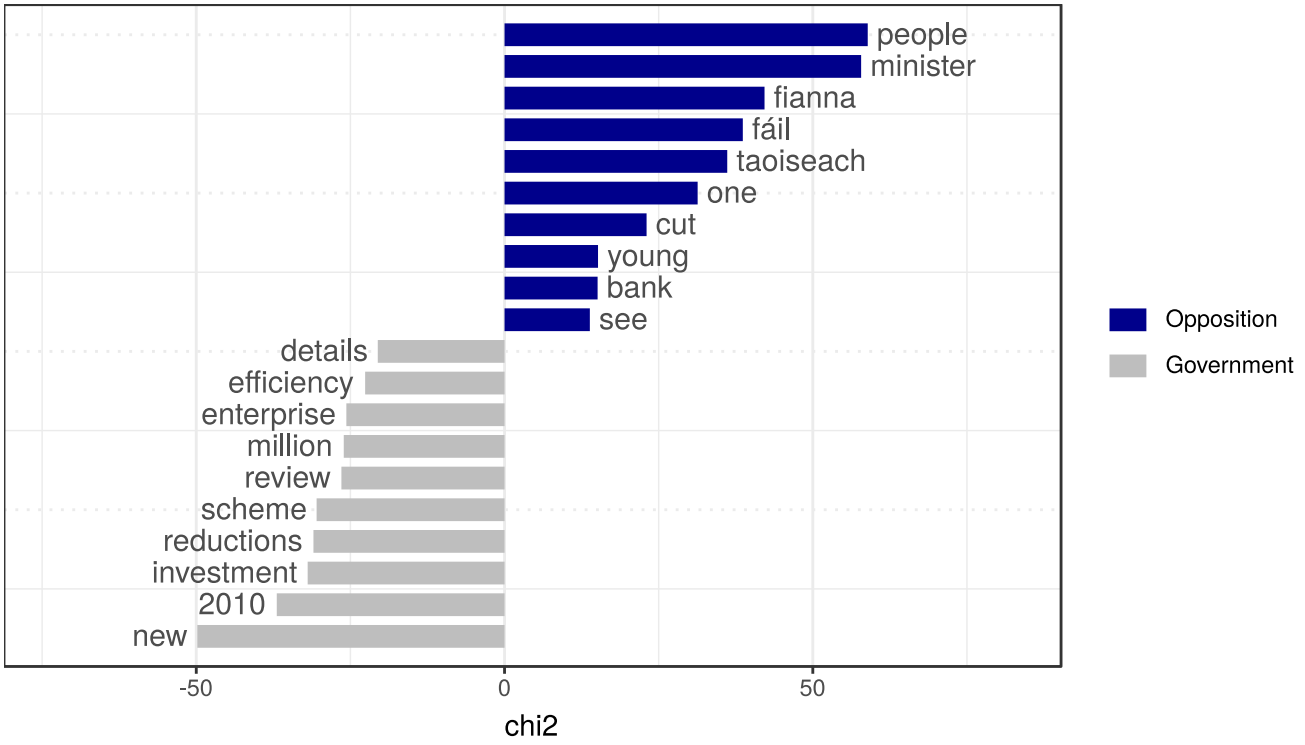
```
# plot a word cloud (not recommended...)
set.seed(34)
textplot_wordcloud(ire_dfm_group,
                   comparison = TRUE,
                   max_words = 40)
```

Relative frequency analysis (keyness)

"Keyness" assigns scores to features that occur differentially across different categories

```
docvars(data_corpus_irishbudget2010, "gov_opp") <-
    ifelse(docvars(data_corpus_irishbudget2010, "party") %in%
               c("FF", "Green"), "Government",
           "Opposition")

# compare government to opposition parties by chi^2
key_dfm <- data_corpus_irishbudget2010 %>%
     dfm(groups = "gov_opp",
         remove = stopwords("english"),
         remove_punct = TRUE)

ire_keyness <- textstat_keyness(key_dfm,
                                target = "Opposition")

plot_key <- textplot_keyness(ire_keyness,
                                margin = 0.2,
                                n = 10)
```

```
plot_key
```


people
minister
fianna
fáil
taoiseach
one
cut
young
bank
see
details
efficiency
enterprise
million
review
scheme
reductions
investment
2010
new
-50
0
50
chi2
Opposition
Government

Collocation analysis: a strings of words approach

```
ire_col <- data_corpus_irishbudget2010 %>%
    tokens() %>%
    textstat_collocations(min_count = 20, tolower = TRUE)

head(ire_col, 10)
```

```
##        collocation count count_nested length   lambda        z
## 1            it is   188            0      2 3.683338 36.89575
## 2         will be    142            0      2 4.132301 35.59424
## 3      this budget   107            0      2 4.295062 31.81840
## 4         we have    119            0      2 3.382721 29.50924
## 5       more than     56            0      2 6.297167 28.82909
## 6  social welfare     70            0      2 8.081143 28.82286
## 7          in the    347            0      2 1.855052 27.87367
## 8       have been     71            0      2 4.525510 26.86640
## 9        has been     62            0      2 4.673994 26.83567
## 10  child benefit     45            0      2 8.320640 24.96713
```

Exercise

1. Repeat the step above, but remove stopwords, and stem the `tokens` object.
2. Compare the most frequent collocations. What has changed?
3. Optional: Conduct a collocation analysis for the German inaugural speeches. Does it work well?

Solution

```
ire_toks_adjusted <- data_corpus_irishbudget2010 %>%
    tokens() %>%
    tokens_remove(pattern = stopwords("en")) %>%
    tokens_wordstem()

col_adjusted <- ire_toks_adjusted %>%
    textstat_collocations(min_count = 5, tolower = FALSE)

head(col_adjusted, 10)
```

```
##       collocation count count_nested length   lambda        z
## 1  public servic    68             0      2 5.467132 26.76130
## 2  social welfar    65             0      2 7.168046 25.84665
## 3  child benefit    36             0      2 6.875431 21.50678
## 4       per week    25             0      2 5.828731 19.40465
## 5      next year    34             0      2 5.200994 18.86080
## 6  public sector    30             0      2 4.089028 17.70144
## 7   Labour Parti    23             0      2 7.486375 17.12913
## 8  privat sector    21             0      2 6.077724 16.32048
## 9  Minist Financ    32             0      2 6.140661 16.26159
## 10 energi effici    15             0      2 7.647629 16.10136
```

```
nrow(col_adjusted)
```

```
## [1] 224
```

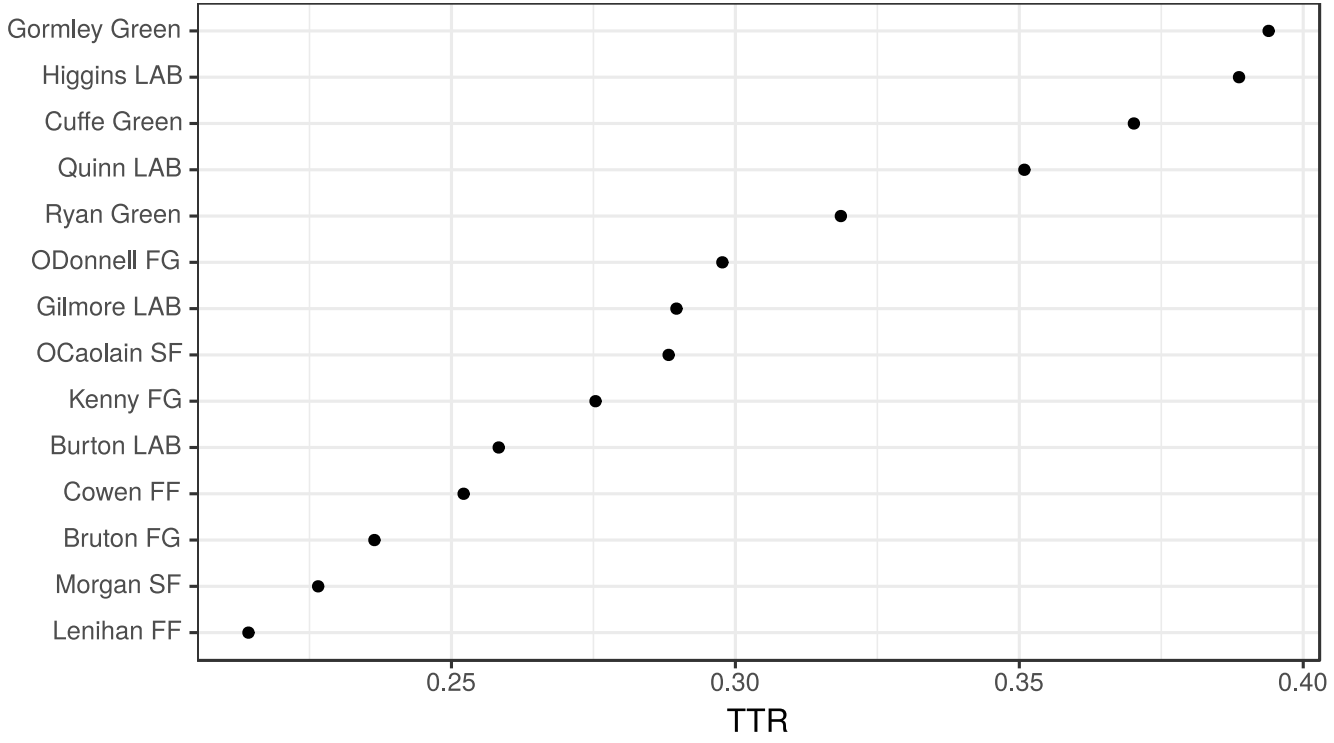
Lexical diversity

```
docnames(data_corpus_irishbudget2010) <-
    paste(docvars(data_corpus_irishbudget2010, "name"),
          docvars(data_corpus_irishbudget2010, "party"), sep = " ")

ire_dfm <- data_corpus_irishbudget2010 %>%
    dfm(remove_punct = TRUE,
        remove_numbers = TRUE)

# estimate Type-Token Ratio
ire_lexdiv <- textstat_lexdiv(ire_dfm, measure = "TTR")

df_lexdiv <- cbind(ire_lexdiv,
                   docvars(ire_dfm))

head(df_lexdiv)
```

```
##     document       TTR year debate number   foren    name party    gov_opp
## 1 Lenihan FF 0.2142487 2010 BUDGET     01   Brian Lenihan    FF Government
## 2  Bruton FG 0.2364312 2010 BUDGET     02 Richard  Bruton    FG Opposition
## 3 Burton LAB 0.2583187 2010 BUDGET     03    Joan  Burton   LAB Opposition
## 4  Morgan SF 0.2265236 2010 BUDGET     04  Arthur  Morgan    SF Opposition
## 5   Cowen FF 0.2521426 2010 BUDGET     05   Brian   Cowen    FF Government
## 6   Kenny FG 0.2753698 2010 BUDGET     06    Enda   Kenny    FG Opposition
```

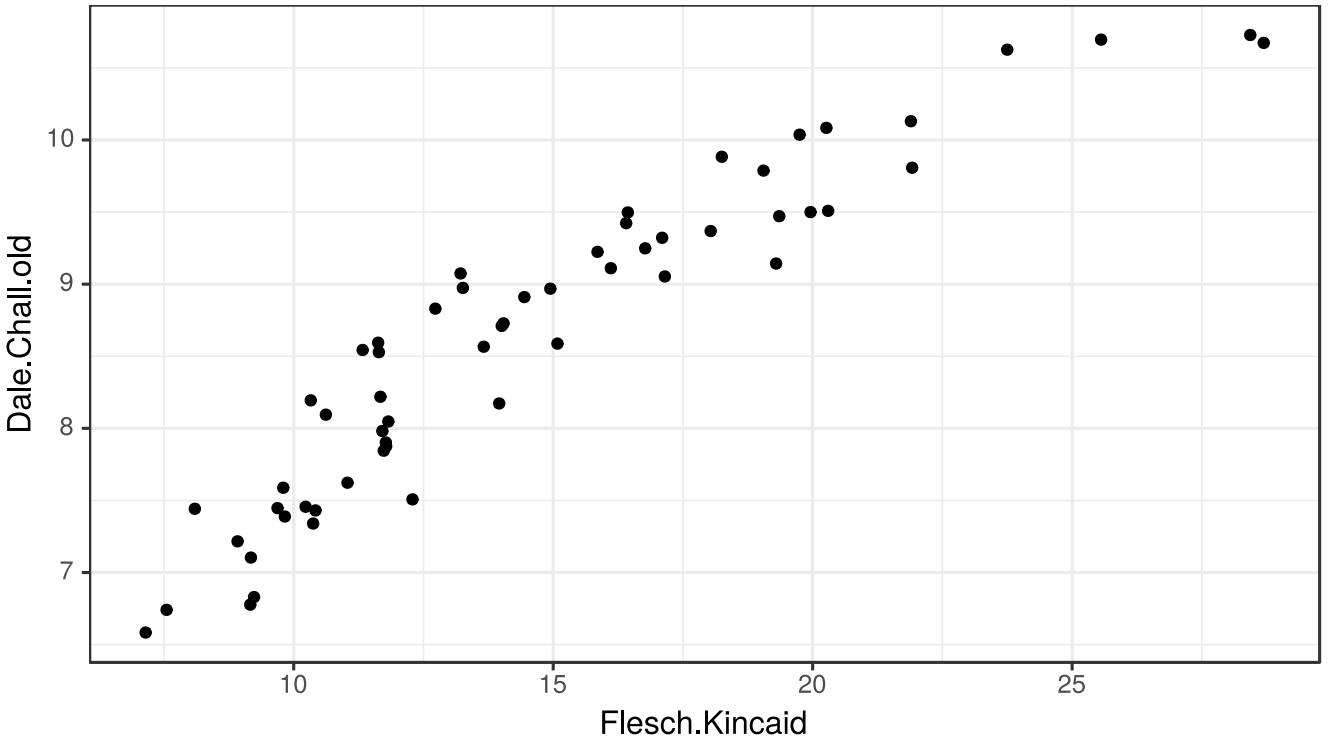
Plot Type-Token Ratio

```
ggplot(df_lexdiv, aes(x = reorder(document, TTR),
                      y = TTR)) +
    geom_point() +
    coord_flip() +
    labs(x = NULL)
```



Readability

```
read_pres <- data_corpus_inaugural %>%
    textstat_readability(measure = c("Flesch.Kincaid", "Dale.Chall.old"))

ggplot(read_pres, aes(x = Flesch.Kincaid, y = Dale.Chall.old)) +
    geom_point()
```

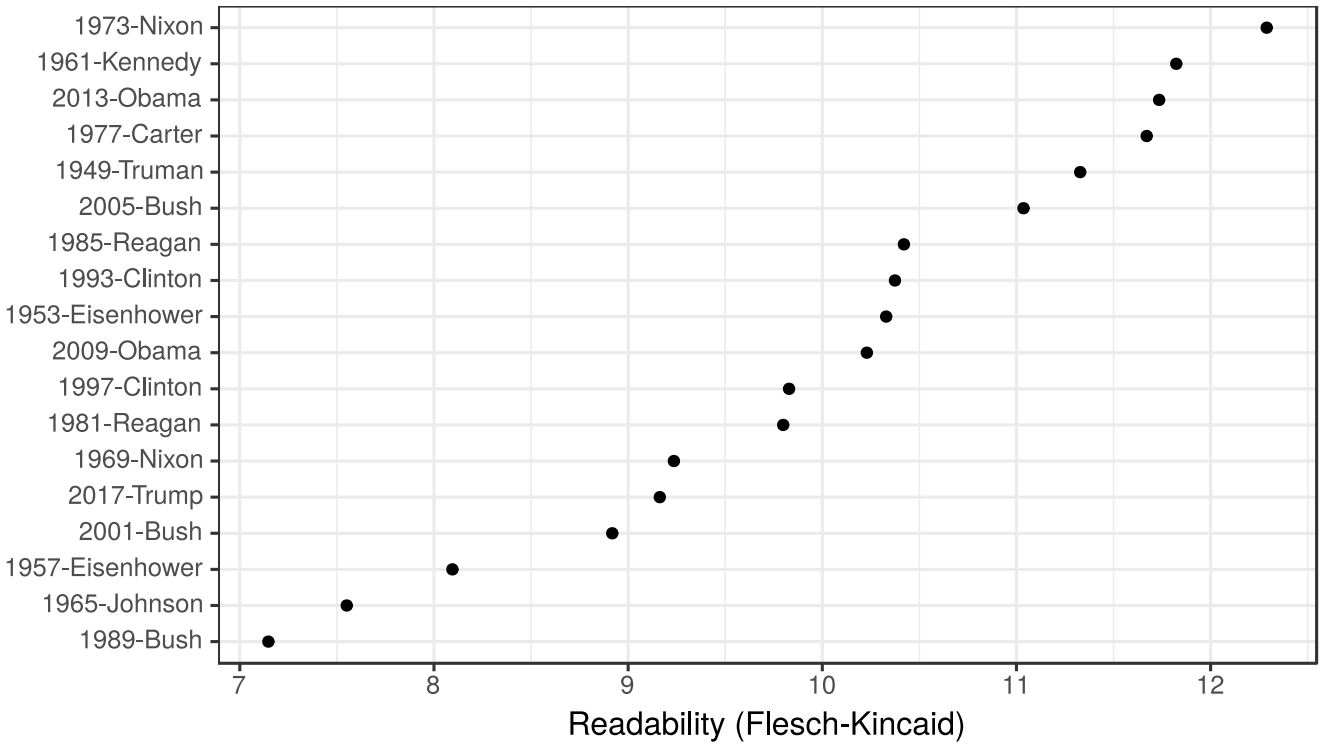


```
cor.test(read_pres$Flesch.Kincaid, read_pres$Dale.Chall.old)
```

```
##
##     Pearson's product-moment correlation
##
## data:  read_pres$Flesch.Kincaid and read_pres$Dale.Chall.old
## t = 19.714, df = 56, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8920247 0.9611137
## sample estimates:
##       cor
## 0.9349112
```

```
read_pres_subset <- data_corpus_inaugural %>%
    corpus_subset(Year > 1948) %>%
    textstat_readability(measure = "Flesch.Kincaid")

plot_read_us <- ggplot(read_pres_subset,
       aes(x = reorder(document, Flesch.Kincaid),
           y = Flesch.Kincaid)) +
    geom_point() +
    coord_flip() +
    labs(x = NULL, y = "Readability (Flesch-Kincaid)")
```

```
plot_read_us
```



See extensively Benoit et al. (forthcoming)

EXTRAS

Setting up a project and importing text files

Setting up a project in RStudio

1. File -> New Project
2. Chose existing folder or create new folder
3. Create folders in the project, e.g.: `code`, `data`, `output`
4. Open the `.Rproj` file: no need to specify working directory (`setwd()`)!

Import multiple text files

```
# load the readtext package
library(readtext)

# data_dir is the location of sample files on your computer.
data_dir <- system.file("extdata/", package = "readtext")

eu_data <- readtext(paste0(data_dir, "/txt/EU_manifestos/*.txt"),
                    docvarsfrom = "filenames",
                    docvarnames = c("unit",
                                    "context",
                                    "year",
                                    "language",
                                    "party"),
                    dvsep = "_",
                    encoding = "ISO-8859-1")
```

Structure of **readtext** data frame

```
head(eu_data)
```

```
## readtext object consisting of 6 documents and 5 docvars.
## # Description: data.frame [6 × 7]
##   doc_id                text                unit  context  year language party
## * <chr>                 <chr>               <chr> <chr>   <int> <chr>    <chr>
## 1 EU_euro_2004_de_PSE… "\"PES · PSE \".… EU    euro     2004 de       PSE
## 2 EU_euro_2004_de_V.t… "\"Gemeinsame\".… EU    euro     2004 de       V
## 3 EU_euro_2004_en_PSE… "\"PES · PSE \".… EU    euro     2004 en       PSE
## 4 EU_euro_2004_en_V.t… "\"Manifesto\n\"… EU    euro     2004 en       V
## 5 EU_euro_2004_es_PSE… "\"PES · PSE \".… EU    euro     2004 es       PSE
## 6 EU_euro_2004_es_V.t… "\"Manifesto\n\"… EU    euro     2004 es       V
```

Check encoding

```
file <- system.file("extdata/txt/EU_manifestos/EU_euro_2004_de_V.txt",
                    package = "readtext")

myreadtext <- readtext(file)
encoding(myreadtext)
```

```
## readtext object consisting of 1 document and 0 docvars.
## # Description: data.frame [1 × 2]
##   doc_id                text
##   <chr>                 <chr>
## 1 EU_euro_2004_de_V.txt "\"Gemeinsame\"..."

## Probable encoding: ISO-8859-1
##   (but note: detector often reports ISO-8859-1 when encoding is actually UTF-8.)
```

Exercise

1. Set up a RProj in a new folder.
2. Download ZIP file with inaugural speeches by German chancellors (https://tinyurl.com/corp-regierung)
3. Copy the folder into your RProj folder.
4. Import all text files using `readtext()`. (Hint: "name_of_folder/*" reads in all files)
5. Create a text corpus of this data frame.

Solution

```
library(readtext)
data_inaugural_ger <- readtext("data/inaugural_germany/*",
                               encoding = "UTF-8",
                               docvarsfrom = "filenames", dvsep="-",
                               docvarnames = c("Year",
                                               "Chancellor",
                                               "Party"))
```

```
data_corpus_inaugural_ger <- corpus(data_inaugural_ger)

summary(data_corpus_inaugural_ger, n = 4)
```

```
## Corpus consisting of 21 documents, showing 4 documents:
##
##                   Text Types Tokens Sentences
##  1949-Adenauer-CDU.txt  2136   7414       292
##  1953-Adenauer-CDU.txt  2630   9708       403
##  1957-Adenauer-CDU.txt  2202   7608       296
##  1961-Adenauer-CDU.txt  2484   8796       402
```

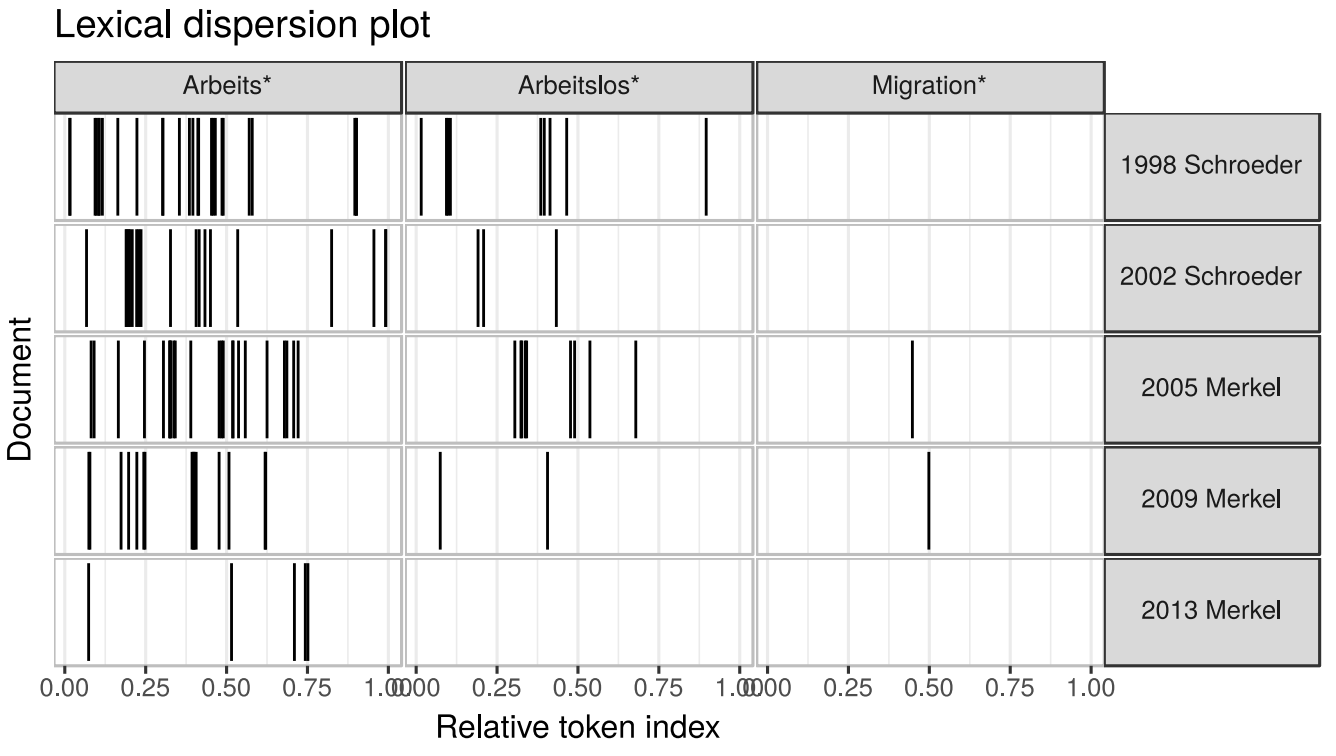
Exercise

1. Create better `docnames` by pasting "Year" and "Chancellor"
2. Select speeches by Gerhard Schröder and Angela Merkel.
3. Check `?textplot_xray()`.
4. Choose words that might only appear in some of the speeches.

Solution

```
docnames(data_corpus_inaugural_ger) <- paste(
    docvars(data_corpus_inaugural_ger, "Year"),
    docvars(data_corpus_inaugural_ger, "Chancellor"),
    sep = ", ")

## alternative

docnames(data_corpus_inaugural_ger) <- paste(data_inaugural_ger$Year,
data_inaugural_ger$Chancellor, sep = " ")

corp_schroeder_merkel <- data_corpus_inaugural_ger %>%
    corpus_subset(Chancellor %in% c("Merkel", "Schroeder"))

plot_xray <- textplot_xray(
    kwic(corp_schroeder_merkel, "Arbeits*"),
    kwic(corp_schroeder_merkel, "Arbeitslos*"),
    kwic(corp_schroeder_merkel, "Migration*")
)
```

```
plot_xray
```


Lexical dispersion plot
Arbeits*
Arbeitslos*
Migration*
1998 Schroeder
2002 Schroeder
2005 Merkel
2009 Merkel
2013 Merkel
Document
0.00
0.25
0.50
0.75
1.00
Relative token index

Handling Multi-word expressions

Compound collocations

Before compounding

```
kwic(ire_toks, phrase("Green Party")) %>%
    head(4)
```

```
## kwic object with 0 rows
```

Compound based on collocations

```
# get 50 most frequent collocations

# compound based on collocations
ire_toks_comp <- ire_toks_adjusted %>%
    tokens_compound(col_adjusted[col_adjusted$z > 3])
```

After compounding

```
nrow(kwic(ire_toks_comp, phrase("Fianna Fáil")))
```

```
## [1] 0
```

```
nrow(kwic(ire_toks_comp, phrase("Fianna_Fáil*")))
```

```
## [1] 70
```

Check compounded words

```
ire_toks_comp %>%
    tokens_keep(pattern = "*_*") %>%
    dfm() %>%
    topfeatures()
```

```
##   fianna_fáil public_servic social_welfar     next_year minist_financ
##            61            44            41            34            30
## child_benefit      per_week public_sector   young_peopl   €_4_billion
##            29            25            25            21            20
```

Similarity between texts

```
simil_toks <- dfm(c("A B",
                    "A B C",
                    "C D E"))

textstat_simil(simil_toks, method = "cosine",
               margin = "documents")
```

```
##           text1     text2
## text2 0.8164966
## text3 0.0000000 0.3333333
```

Exercise

1. Create a dfm from `data_corpus_irishbudget2010`
2. Group it by party and run `textstat_simil()`.
3. What parties are most similar?
4. Do the substantive conclusion change when removing stopwords and punctuation, and stemming the dfm?

Solution

```
# similarities for documents
sim_1 <- data_corpus_irishbudget2010 %>%
    dfm() %>%
    dfm_group(groups = "party") %>%
    textstat_simil(method = "cosine",
                   margin = "documents")

sim_1
```

```
##              FF        FG     Green       LAB
## FG    0.9540889
## Green 0.9454957 0.9495884
## LAB   0.9632787 0.9825914 0.9505133
## SF    0.9683231 0.9790162 0.9384105 0.9804133
```

```
sim_2 <- data_corpus_irishbudget2010 %>%
    dfm(remove = stopwords("en"),
        stem = TRUE, remove_punct = TRUE) %>%
    dfm_group(groups = "party") %>%
    textstat_simil(method = "cosine",
                   margin = "documents")

sim_2
```

```
##              FF        FG     Green       LAB
## FG    0.6212161
## Green 0.6574813 0.6448546
## LAB   0.6664573 0.7950098 0.6549404
## SF    0.6950464 0.7905741 0.6507070 0.8104237
```