Text Analysis in R with **quanteda**

Wednesday, Sept. 21, 2022 • 9:00-12:00
Lamont Library, Room B-30

Cole Crawford, Jess Cohen-Tanugi, Hugh Truslow

Outline

1. **Welcome and Introductions**

2. Text Analysis Fundamentals

3. Situating Quanteda

4. Getting started

5. Working with a corpus

Break

6. Tokenization

7. Document-feature matrix

8. Dictionary (sentiment) Analysis

9. Textual statistics

Instructors

Cole Crawford

Jess Cohen-Tanugi

Hugh Truslow

You!

- Name
- Affiliation
- How you use / want to use text analysis
- Popcorn to next person

These slides have been adapted, with permission, from Kenneth Benoit’s workshop at the 2019 Annual Conference of the International Association for Social Science Information Services and Technology (IASSIST), in Sydney, Australia, May 28, 2019, at the University of New South Wales.

Purpose of the workshop

- Provide an overview of the text analysis workflow
- Introduce tools for quantitative text analysis
- 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

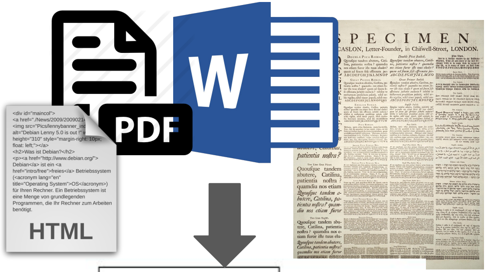
- Those familiar with text analysis methods but not in **R/quanteda**
- Those who have tried **R/quanteda** but want to learn more proficiency
- The merely "text-curious"
- No real prerequisites: We're all here to learn

Text analysis fundamentals

Text Analysis Workflow

Corpus Creation → Preprocessing → Reshaping → Analysis

Corpus Creation



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

Preprocessing

- Tokenization
- Lowercasing
- Stemming
- Lemmatization
- Stopword removal
- Punctuation removal
- Normalization
- Noise removal
- Enrichment and annotation

"Of all tax, income taxes are the worst."

['tax', 'incom', 'tax', 'worst']

Reshaping

Document term matrix or other representations

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

Analysis

Statistics
- Term frequencies
- Keyness
- Readability
- Lexical Diversity
- Similarity, distance

Models
- Supervised ML
- Unsupervised ML
- Scaling
- Word embeddings
- Topic modeling

Plots
- Keyness
- Networks
- Scaling
- Word Clouds

Corpus Creation

- Texts: Organized into *documents*
- Corpus: Collection of texts
  - Often with associated document-level metadata ("document variables" or *docvars*)
  - Examples: { “pages”: 326, “pub_date”: 1897, “genre”: “crime”}
  - Plural is corpora
- Domain knowledge
- Creating a corpus
  - Downloading
  - Web scraping
  - Transcribing
  - Copyright issues
- Document ingestion
  - Plaintext
  - HTML / XML
  - PDF
  - Images
  - Word

Preprocessing

- Bringing a text into a predictable, analyzable state for a specific task
  - Task: approach + domain
  - Human consumption (reading text) -> computer consumption (preprocessed text)
- Steps
  - Tokenization
  - Stemming
  - Lemmatization
- Definitions
  - Stems: words with suffixes removed (using a set of rules)
  - Lemmas: canonical word form
  - Tokens: a sequence of characters grouped together as a useful semantic unit
    - words
    - could also include punctuation characters or symbols
    - multi-word expressions
    - named entities
    - usually, but not always, delimited by spaces
  - Type: a unique token

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

Preprocessing (continued)

- Steps
    - Lowercasing
    - Stopword removal
    - Punctuation removal
    - Normalization
    - Noise removal
    - Enrichment and annotation
- Definitions
    - 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 linguistic 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)

Sample Preprocessing Pipeline

- Corpus
  - “A corpus is a set of documents.”
  - “This is the second document in the corpus.”
- 1. Lowercase
  - "a corpus is a set of documents."
  - "this 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 documents"
  - "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

```r
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"
```

```r
tokens_wordstem(toks)
```

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

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

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

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

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

Reshaping

Document Term Matrix

Document 1: "*A corpus is a set of documents*."

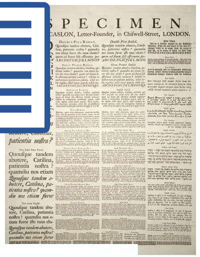
Document 2: "*This is the second document in the corpus.*"

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

Text Analysis Workflow

Corpus Creation → Preprocessing → Reshaping → Analysis

Corpus Creation



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

Preprocessing

- Tokenization
- Lowercasing
- Stemming
- Lemmatization
- Stopword removal
- Punctuation removal
- Normalization
- Noise removal
- Enrichment and annotation

"Of all tax, income taxes are the worst."

['tax', 'incom', 'tax', 'worst']

Reshaping

Document term matrix or other representations

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

Analysis

Statistics

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

Models

- Supervised ML
- Unsupervised ML
- Scaling
- Word embeddings
- Topic modeling

Plots

- Keyness
- Networks
- Scaling
- Word Clouds

Text analysis tools and libraries

Tools

- Voyant Tools: https://voyant-tools.org/
- MALLET: http://mallet.cs.umass.edu/quick-start.php
- Google NGrams: http://books.google.com/ngrams/
- Many commercial options, from GUIs to APIs
  - MonkeyLearn
  - IBM Watson
  - Google Cloud NLP
  - Amazon Comprehend
  - Many more ...

Python Text Analysis Libraries

NLTK

- **N**atural **L**anguage **T**ool**k**it
- Released 2001
- Focus: teaching and research
- Functions: Text manipulation and preprocessing
- String-oriented
- Less opinionated; many algorithms
- Not as performant
- Researcher > developer

spaCy

- Released 2015
- Focus: production ready NLP code
- Functions: text preprocessing, word vectors, tagging, classification, deep learning integration
- Object oriented
- Opinionated: state-of-the-art algorithms
- Developer > researcher

scikit learn

- Released 2007
- Focus: machine learning
- Functions: classification, clustering, regression

GENSIM topic modelling for humans

- Released 2009
- Focus: NLP, especially topic modeling and document similarity
- Functions: topic modeling, TF-IDF, fastText, word2vec, doc2vec

R Text Analysis Libraries

quanteda
Quantitative Analysis of Textual Data

tm

- Many methods for importing data and creating corpora
- Reading Word docs, PDFs
- Metadata management
- Preprocessing
- Analysis

- Succinct and powerful fullstack text analysis (https://quanteda.io/articles/pkgdown/replication/digital-humanities.html?q=topic%20modeling#topic-modelling)
- Preprocessing
- Document term matrices
- Metadata management
- Document classification
- Topic modeling
- Can work with tm corpora

- Others
  - spaCyR
  - OpenNLP
  - tidytext
  - corpus
- See: https://quanteda.io/articles/pkgdown/comparison.html

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, and check versions:

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

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)
```

Reproduce example in **quanteda**

Create a document-feature matrix

```
dfm(mycorp, remove_punct = TRUE,
dfm_remove = stopwords("en"), dfm_wordstem = TRUE)

dfmat_mycorp <- tokens(my_corp, remove_punct = TRUE) %>%
  dfm()

dfm_remove(dfmat_mycorp, stopwords("en") %>%
          dfm_wordstem(dfmat_mycorp, language = quanteda_options("language_stemmer"))
```

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

The **quanteda** infrastructure

Design of the package

- consistent grammar
- flexible for power users, simple for beginners
- analytic transparency and reproducibility
- compatibility 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
- https://quanteda.org

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 `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_()`

Naming conventions and useful shortcuts

Recommended naming conventions for objects

In the **quanteda** style guide, we recommend to name objects consistently, e.g.:

- Corpus: `corp_...`
- Tokens object: `toks_...`
- Dfm: `dfmat_...`
- Textmodel: `tmod_...`

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)
```

Working with corpus objects

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()
```

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

Access number of types and tokens of corpus

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

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

Overview of document-level variables

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

Exercise

1. Based on `data_corpus_inaugural`, create an object
   `data_corpus_postwar` (speeches since 1945).

2. What speech has the most tokens? What speech has the 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()

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

Adding document-level variables

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

head(docvars(data_corpus_inaugural, "Order"))

# 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))
```

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)


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

ndoc(corp_sentence_subset)
```

Solution (cont'd)

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

```
sum(ntoken(data_corpus_inaugural))
```

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

Outline

1. Welcome and Introductions
2. Text Analysis Fundamentals
3. Situating Quanteda
4. Getting started
5. Working with a corpus

Break

6. Tokenization
7. Document-feature matrix
8. Dictionary (sentiment) Analysis
9. Textual statistics

Tokenization

Tokenization of a corpus

```
corp_immig <- corpus(data_char_ukimmig2010)
toks_immig <- tokens(corp_immig)
```

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

Remove punctuation

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

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

Remove stopwords

```
toks_nostopw <- tokens_select(toks_immig, stopwords("en"),
                              selection = "remove")

# Note: equivalent to
toks_nostopw <- tokens_remove(toks_immig, stopwords("en"))

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

Customize stopwords list

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

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

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

"will" %in% my_stopwords_en
```

Review variations

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

head(toks_nopunct[[1]], 20)

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

Select certain terms

```
toks_immig_select <- toks_immig %>%
    tokens_select(pattern = c("immig*", "migra*"),
                  padding = TRUE)

head(toks_immig_select[[1]], 20)

toks_immig_no_pad <- toks_immig %>%
    tokens_select(pattern = c("immig*", "migra*"),
                  padding = FALSE)

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

Select certain terms and their context

```r
# specify number of surrounding words using window
toks_immig_window <- tokens_select(toks_immig,
                                   pattern = c('immig*', 'migra*'),
                                   padding = TRUE, window = 5)

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

Compound multiwords expressions

```
kw_multi <- kwic(data_char_ukimmig2010, phrase(c('asylum seeker*',
                                 'british citizen*')))

head(kw_multi, 5)
```

Compound multiwords expressions

Preserve these expressions in bag-of-word analysis:

```
toks_comp <- tokens(data_char_ukimmig2010) %>%
    tokens_compound(phrase(c('asylum seeker*',
                                   'british citizen*')))

kw_comp <- kwic(toks_comp, c('asylum_seeker*', 'british_citizen*'))

head(kw_comp, 4)
```

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"))

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

N-grams

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

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

Skipgrams

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

Look up tokens from a dictionary

```
toks <- tokens(data_char_ukimmig2010)

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

print(dict)


dict_toks <- tokens_lookup(toks, dictionary = dict)

head(dict_toks, 2)
```

The transition from tokens() to dfm()

dfm(dict_toks)

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

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

ire_ntype <- ntype(toks_ire)

ire_ntoken <- ntoken(toks_ire)
```

Document-feature matrix

Exercise

1. Create a document-feature matrix from the tokens object above.

2. Get the 50 most frequent terms using `topfeatures()`.

Solution

```r
toks_ire <- tokens(data_corpus_irishbudget2010)

dfmat_ire <- dfm(toks_ire) # dfm() transforms to lower case by default

topfeatures(dfmat_ire)


## alternative approach without tokens() -- less control!
dfmat_ire2 <- data_corpus_irishbudget2010 %>%
    dfm(remove_punct = TRUE,
        remove = stopwords("en"),
        stem = TRUE)
topfeatures(dfmat_ire2)
```

Select features based on frequencies

```
nfeat(dfmat_ire)

dfmat_ire_trim1 <- dfm_trim(dfmat_ire,
min_termfreq = 5)

nfeat(dfmat_ire_trim1)

dfmat_ire_trim2 <- dfm_trim(dfmat_ire,
                       min_docfreq = 5)

nfeat(dfmat_ire_trim2)

dfmat_ire_trim3 <- dfm_trim(dfmat_ire,
max_docfreq = 0.1,
docfreq_type = "prop")

nfeat(dfmat_ire_trim3)
```

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 the advantages and problems of each weighting scheme?

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

Solution

```
dfmat_fruits <- dfm(texts)

dfm_weight(dfmat_fruits, scheme = "count")


dfm_weight(dfmat_fruits, scheme = "boolean")


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

Sentiment analysis

Sentiment analysis on the Irish Budget corpus

Sentiment analysis using the Lexicoder Sentiment Dictionary (data_dictionary_LSD2015)

```
#Take a quick look at the dictionary
head(data_dictionary_LSD2015)
```

Options are: negative, positive, negating a positive, negating a negative. Here is more info on the dictionary.

```
# simple example
txt <- "This aggressive policy will not win friends."
tokens_lookup(tokens(txt), dictionary = data_dictionary_LSD2015, exclusive = FALSE)
## tokens from 1 document.
## text1 :
## [1] "This"   "NEGATIVE"   "policy"   "will"   "NEG_POSITIVE" "POSITIVE" "."
```

Sentiment analysis

```
#Here are the names of the current docvars with this corpus,
#which consists of 14 documents.
names(docvars(data_corpus_irishbudget2010))

#We're going to add a docvar that says if a document is from a party associated
#with the 2010 gov, or the opposition:
docvars(data_corpus_irishbudget2010, "gov_opp") <-
  ifelse(docvars(data_corpus_irishbudget2010, "party") %in% c("FF", "Green"),
         "Government", "Opposition")
```

Sentiment analysis

```
# tokenize and apply dictionary
toks_dict <- data_corpus_irishbudget2010 %>%
  tokens() %>%
  tokens_lookup(dictionary = data_dictionary_LSD2015)
head(toks_dict)
# transform to a dfm
dfmat_dict <- dfm(toks_dict)

#What did we get?
head(dfmat_dict)
```

Sentiment analysis

Use convert() to create a data frame

```
#Convert our output to a data frame
dict_output <- convert(dfmat_dict, to = "data.frame")
```

Estimate sentiment using a log formula

```
#Add a column to 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))
```

Sentiment analysis

```
View(dict_output)
View(docvars(data_corpus_irishbudget2010))

#Combine the data to get a bigger dataframe with the docvars info included
dict_output <- cbind(dict_output, docvars(data_corpus_irishbudget2010))

View(dict_output)
```

Sentiment analysis

Plot our sentiment analysis using ggplot. If you haven't used ggplot, it's confusing, some bare bones:

- Specify data (a data.frame)
- Specify the aesthetics (x-axis, y-axis, colours, shapes etc.)
- Choose a geometric object (e.g. scatterplot, boxplot)

```
#We're going to do a scatterplot of negative vs positive from dict_output
#to show you the very basics. First, specify data and axes:
library(ggplot2)
myplot <- ggplot(data = dict_output,
                 aes(x = negative, y = positive))
myplot
```

Sentiment analysis

```
#Specify a geometry -- you can put aesthetics in there, too.
myplot +
  geom_point(aes(colour = party)) +
  labs(x = "Negative Statements",
       y = "Positive Statements")
```

Sentiment analysis

Let’s plot our sentiment analysis using ggplot. If you haven’t used ggplot, it’s confusing; some bare bones:

- Specify data (a data.frame)
- Specify the aesthetics (x-axis, y-axis, colours, shapes etc.)
- Choose a geometric object (e.g. scatterplot, boxplot)

```
#We're going to do a scatterplot of negative vs positive from dict_output
#to show you the very basics. First, specify data and axes:
library(ggplot2)
myplot <- ggplot(data = dict_output,
                 aes(x = negative, y = positive))
myplot
#Then, specify a geometry -- you can put aesthetics in there, too.
myplot +
  geom_point(aes(colour = party)) +
  labs(x = "Negative Statements",
       y = "Positive Statements")
```

Sentiment analysis

```
#Now let's do something more meaningful.
tplot_sent <- 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")

tplot_sent
```

Textual statistics

Textual statistics: Simple frequency analysis

```
#Simple frequency analysis
ire_dfm <- data_corpus_irishbudget2010 %>%
  dfm(remove = stopwords("en"),
      remove_punct = TRUE)
ire_freq <- ire_dfm %>%
  textstat_frequency(n=15, ties_method = "first")
head(ire_freq)

#Plot
ire_freqplot <- ire_freq %>%
  ggplot(aes(x = frequency, y = reorder(feature, frequency))) +
  geom_point() +
  labs(x = "Frequency", y = NULL)
ire_freqplot
```

Textual statistics: Frequency analysis for groups

```
#Frequency analysis for groups
ire_dfm_group <- ire_dfm %>%
  dfm_group(groups = "party")

#Word cloud (don't do it!)
set.seed(34)
textplot_wordcloud(ire_dfm_group,
                   comparison = TRUE,
                   max_words = 40)
```

Textual statistics: Relative frequency analysis

Relative frequency analysis, also called keyness, compares the frequency of words in one sample compared with another.

```
#"Keyness"
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)

#What did we do?
head(key_dfm)
```

Textual statistics: Relative frequency analysis

```
#Now we're ready to calculate keyness
ire_keyness <- textstat_keyness(key_dfm,
                                target = "Opposition")
head(ire_keyness)
#plot!
plot_key <- textplot_keyness(ire_keyness,
                             margin = 0.2,
                             n = 10)
plot_key
```

Textual statistics: Collocation analysis

In corpus linguistics, a collocation is a series of words or terms that co-occur more often than would be expected by chance.

```
ire_col <- data_corpus_irishbudget2010 %>%
  tokens() %>%
  textstat_collocations(min_count = 20, tolower = TRUE)
head(ire_col, 10)
```

Textual statistics: Exercise

1. Repeat the step above, but remove stopwords, and stem the tokens object.
2. Compare the most frequent collocations. What has changed?

Textual statistics: Solution

1. Repeat the step above, but remove stopwords, and stem the tokens object.
2. Compare the most frequent collocations. What has changed?

```
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)
nrow(col_adjusted)
```

Textual statistics: Lexical diversity

Compute lexical diversity through with the type-token ratio.

```
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)
```

Textual statistics: Lexical diversity

```
#Plot the type-token ratio
ggplot(df_lexdiv, aes(x = reorder(document, TTR),
                      y = TTR)) +
  geom_point() +
  coord_flip() +
  labs(x = NULL)
```

Topic Modeling

Topic Models

Topic models are algorithms for discovering the main **themes** that pervade a large and otherwise **unstructured** collection of documents, and for organizing the collection according to these themes, or **topics**.

(Blei 2012)

Definitions and Preparation

Documents: any collection of texts; should be comparable

Topics: a “pattern of tightly co-occurring terms” (Blei 2012)

Preparation Decisions

- Prepare your corpus: size (larger is better), cleanup, etc
- Choose the **number of topics** you want the model to produce

Topic Modeling: How do the algorithms work?

(Non-math explanations …)

- Latent Semantic Analysis (LSA)
- Latent Dirichlet Analysis (LDA)
- Others: Correlated Topic Model (CTM), Structural Topic Model (STM)

Steps (LDA)

1. Choose the topics
2. Generate the documents

Figure 1. The intuitions behind latent Dirichlet allocation. We assume that some number of "topics," which are distributions over words, exist for the whole collection (far left). Each document is assumed to be generated as follows. First choose a distribution over the topics (the histogram at right); then, for each word, choose a topic assignment (the colored coins) and choose the word from the corresponding topic. The topics and topic assignments in this figure are illustrative—they are not fit from real data. See Figure 2 for topics fit from data.

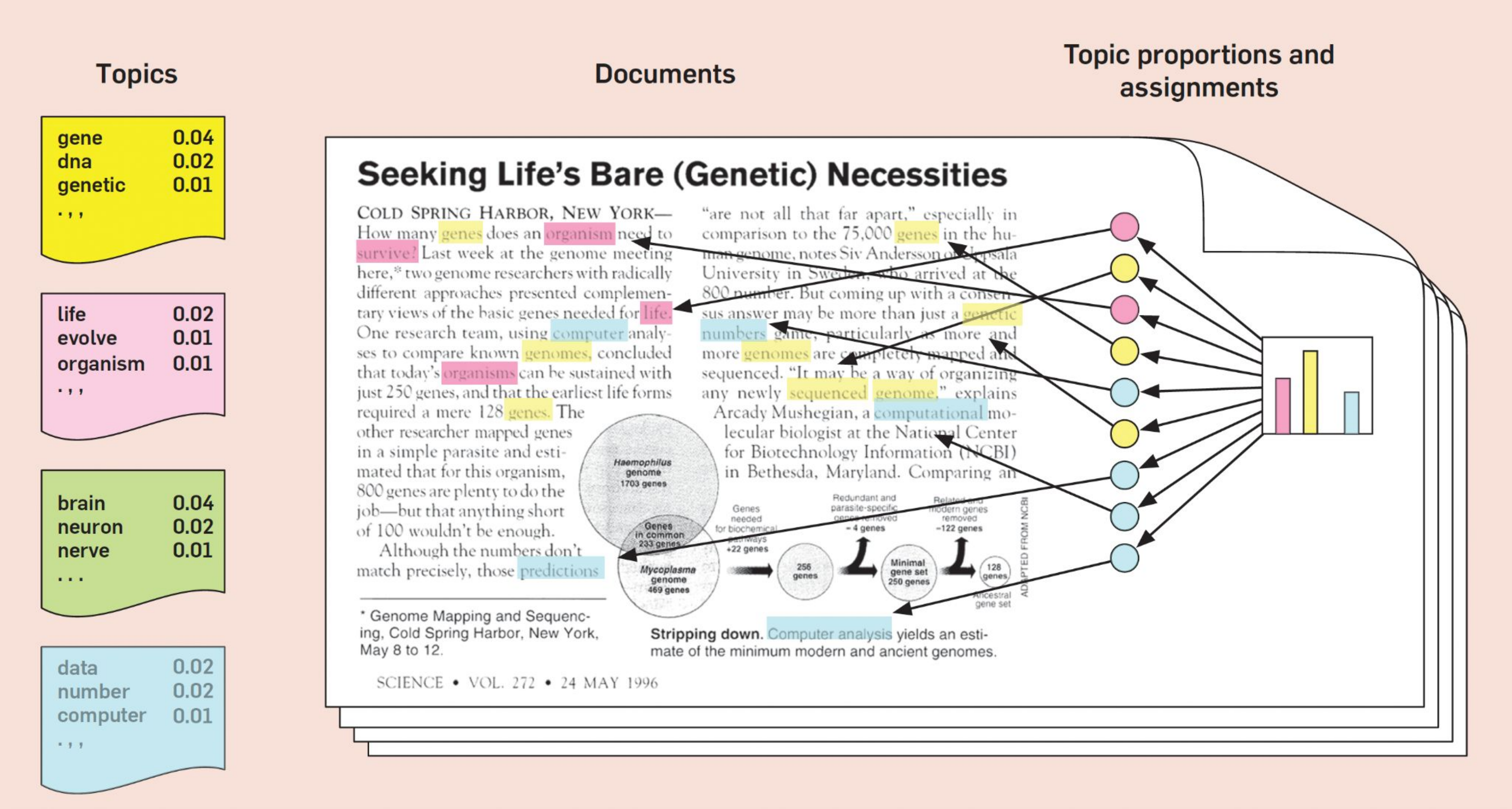
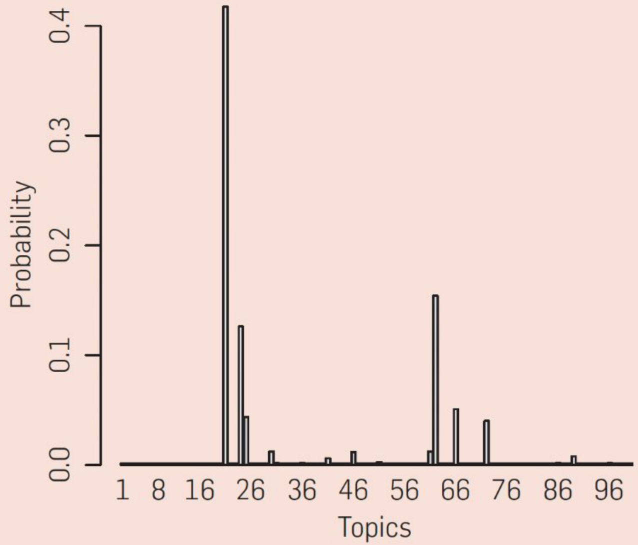


Figure 2. Real inference with LDA. We fit a 100-topic LDA model to 17,000 articles from the journal *Science*. At left are the inferred topic proportions for the example article in Figure 1. At right are the top 15 most frequent words from the most frequent topics found in this article.



"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Text Mining Project Example

[Mining the Dispatch]