quanteda initiative

Quantitative Text Analysis using R and **quanteda**

Kenneth Benoit

University of Münster (27–28 June 2019)

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 workshop 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 me

- Ken Benoit (http://kenbenoit.net)
- Professor of Computational Social Science, Department of Methodology, London School of Economics and Political Science
- 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

Example from my research

AJPS AMERICAN JOURNAL of POLITICAL SCIENCE

Measuring and Explaining Political Sophistication through Textual Complexity

Kenneth Benoit London School of Economics and Political Science
Kevin Munger Pennsylvania State University
Arthur Spirling New York University

Abstract: *Political scientists lack domain-specific measures for the purpose of measuring the sophistication of political communication. We systematically review the shortcomings of existing approaches, before developing a new and better method along with software tools to apply it. We use crowdsourcing to perform thousands of pairwise comparisons of text snippets and incorporate these results into a statistical model of sophistication. This includes previously excluded features such as parts of speech and a measure of word rarity derived from dynamic term frequencies in the Google Books data set. Our technique not only shows which features are appropriate to the political domain and how, but also provides a measure easily applied and rescaled to political texts in a way that facilitates probabilistic comparisons. We reanalyze the State of the Union corpus to demonstrate how conclusions differ when using our improved approach, including the ability to compare complexity as a function of covariates.*

Example from my research (cont)

Your turn!

1. Name?
2. Department, degree?
3. Research interests?
4. Previous experience with text analysis/R?
5. Why are you interested in the workshop?

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
- 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. "University of Bergen" or "United Kingdom"
- 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(c("A corpus is a set of documents.",
                 "This is the second document in the corpus"),
               remove_punct = TRUE)
toks
```

```
## tokens from 2 documents.
## text1 :
## [1] "A"         "corpus"    "is"        "a"         "set"       "of"
## [7] "documents"
##
## text2 :
## [1] "This"     "is"       "the"      "second"   "document" "in"
## [7] "the"      "corpus"
```

```
tokens_wordstem(toks)
```

```
## tokens from 2 documents.
## text1 :
## [1] "A"        "corpus"   "is"       "a"        "set"      "of"
## [7] "document"
##
## text2 :
## [1] "This"     "is"       "the"      "second"   "document" "in"
## [7] "the"      "corpus"
```

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

```
## tokens from 2 documents.
## text1 :
## [1] "corpus"   "set"      "document"
##
## text2 :
## [1] "second"   "document" "corpus"
```

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

```
## [1] "corpus"   "set"      "document" "second"
```

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] '1.4.5'
```

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

```
## [1] '0.75'
```

Reproduce example in **quanteda**

Create a text corpus

```
library(quanteda)
# Create text corpus
corp <- 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(corp)`.

```
summary(corp)
```

```
## Corpus consisting of 2 documents:
##
##   Text Types Tokens Sentences
##  text1     8      8         1
##  text2     8      9         1
##
## Source: /Users/kbenoit/Dropbox (Personal)/GitHub/quanteda/workshops/modules/* on x86_64 by kbenoit
## Created: Tue Jun 25 20:54:20 2019
## Notes:
```

Reproduce example in quanteda

Create a document-feature matrix

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

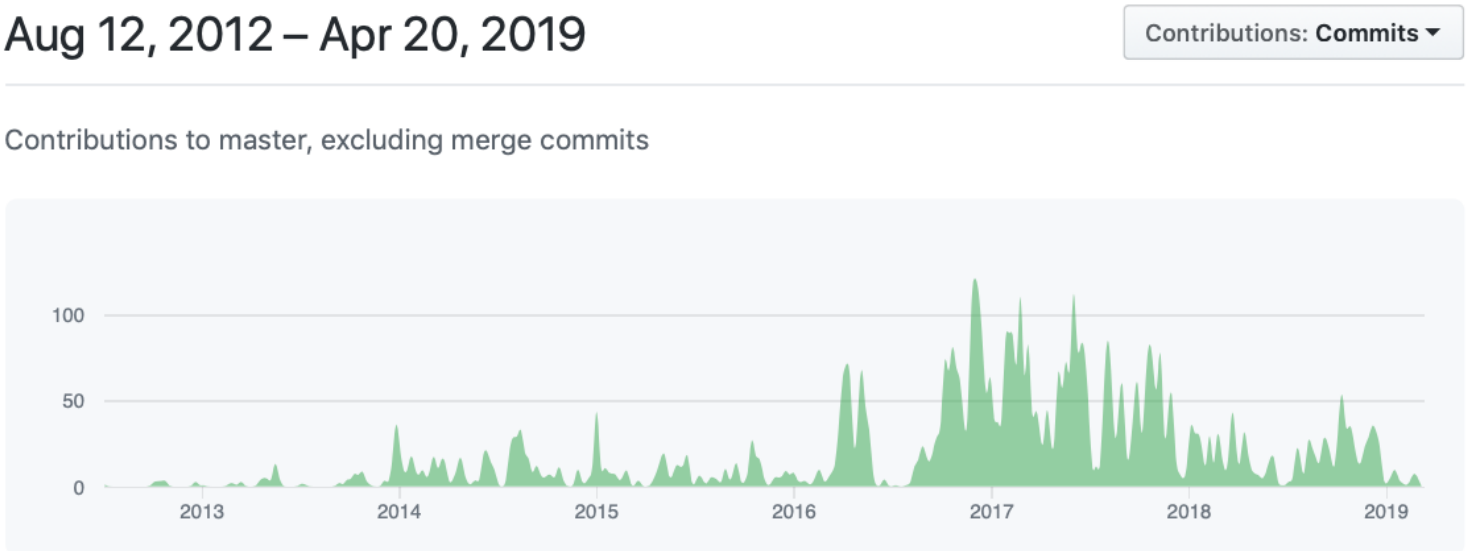
```
## Document-feature matrix of: 2 documents, 4 features (25.0% sparse).
## 2 x 4 sparse Matrix of class "dfm"
##        features
## docs    corpus set document second
##   text1      1   1        1      0
##   text2      1   0        1      1
```

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, 28 releases
- 8,300 commits, 270,000 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
- 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 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_()`

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()
```

```
## [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
## 1789-Washington 1789 Washington    George
## 1793-Washington 1793 Washington    George
## 1797-Adams      1797      Adams      John
## 1801-Jefferson  1801  Jefferson    Thomas
## 1805-Jefferson  1805  Jefferson    Thomas
## 1809-Madison    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
## 1789-Washington 1789 Washington    George     1 George Washington
## 1793-Washington 1793 Washington    George     2 George Washington
## 1797-Adams      1797      Adams      John     3        John Adams
## 1801-Jefferson  1801  Jefferson    Thomas     4  Thomas Jefferson
## 1805-Jefferson  1805  Jefferson    Thomas     5  Thomas Jefferson
## 1809-Madison    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
```

Solution (cont.)

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

sum(ntoken(data_corpus_inaugural))
```

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

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

```
## [1] 142617
```

Keywords-in-context

Keywords-in-context (KWIC)

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

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

head(kw_security, 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(kw_security)
```

```
## [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. BONUS: Send the results of the kwic output to `textplot_xray()`.

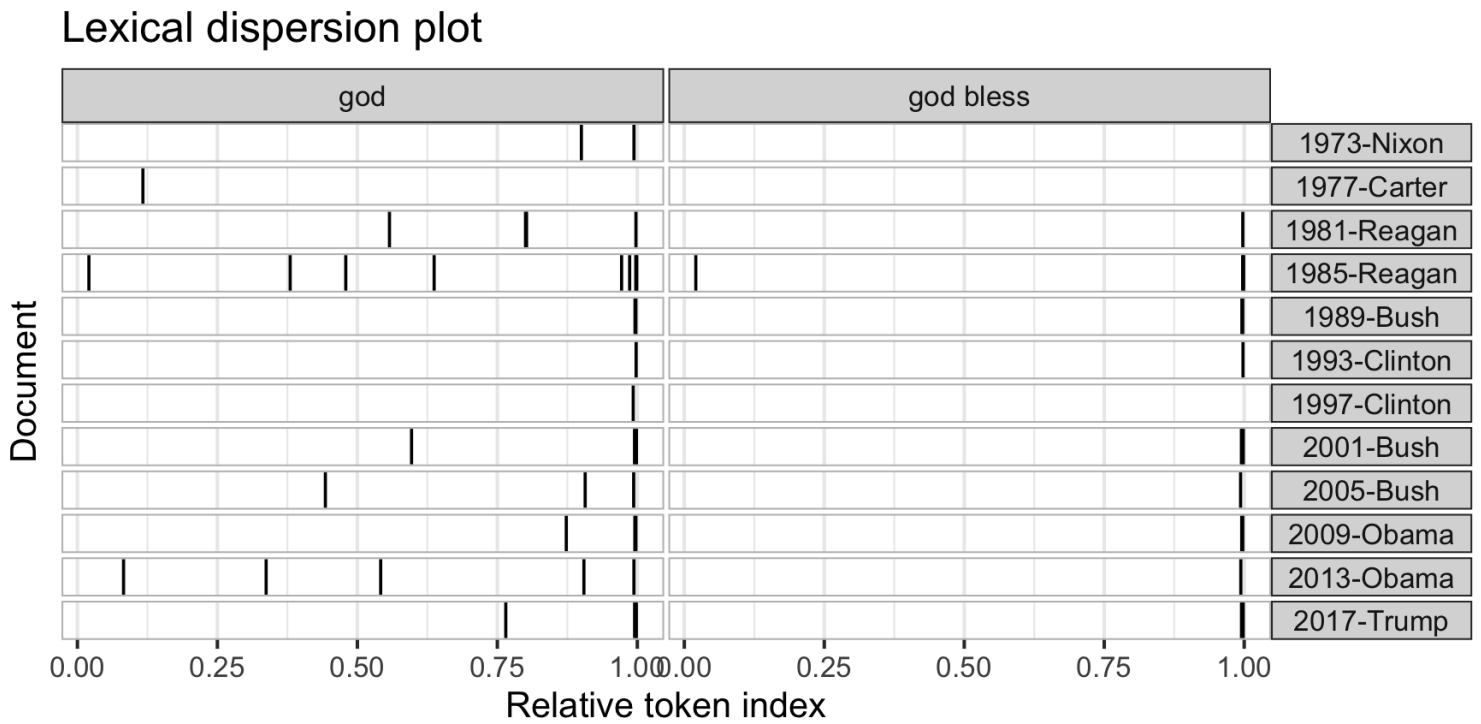
Solution

```
kw_god <- corpus_subset(data_corpus_inaugural, Year > 1970) %>%
  kwic("god", window = 3)

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

tail(kw_god_bless)
```

```
##
##   [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.
```

Solution (cont.)

```
textplot_xray(kw_god, kw_god_bless)
```



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: df[,7] [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: df[,2] [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 Year Chancellor Party
##  1949-Adenauer-CDU.txt  2136   7414       292 1949   Adenauer   CDU
##  1953-Adenauer-CDU.txt  2630   9708       403 1953   Adenauer   CDU
##  1957-Adenauer-CDU.txt  2202   7608       296 1957   Adenauer   CDU
##  1961-Adenauer-CDU.txt  2484   8796       402 1961   Adenauer   CDU
##
## Source: /Users/kbenoit/Dropbox (Personal)/GitHub/quanteda/workshops/modules/* on x86_64 by kbenoit
## Created: Tue Jun 25 20:54:23 2019
## Notes:
```

Exercise

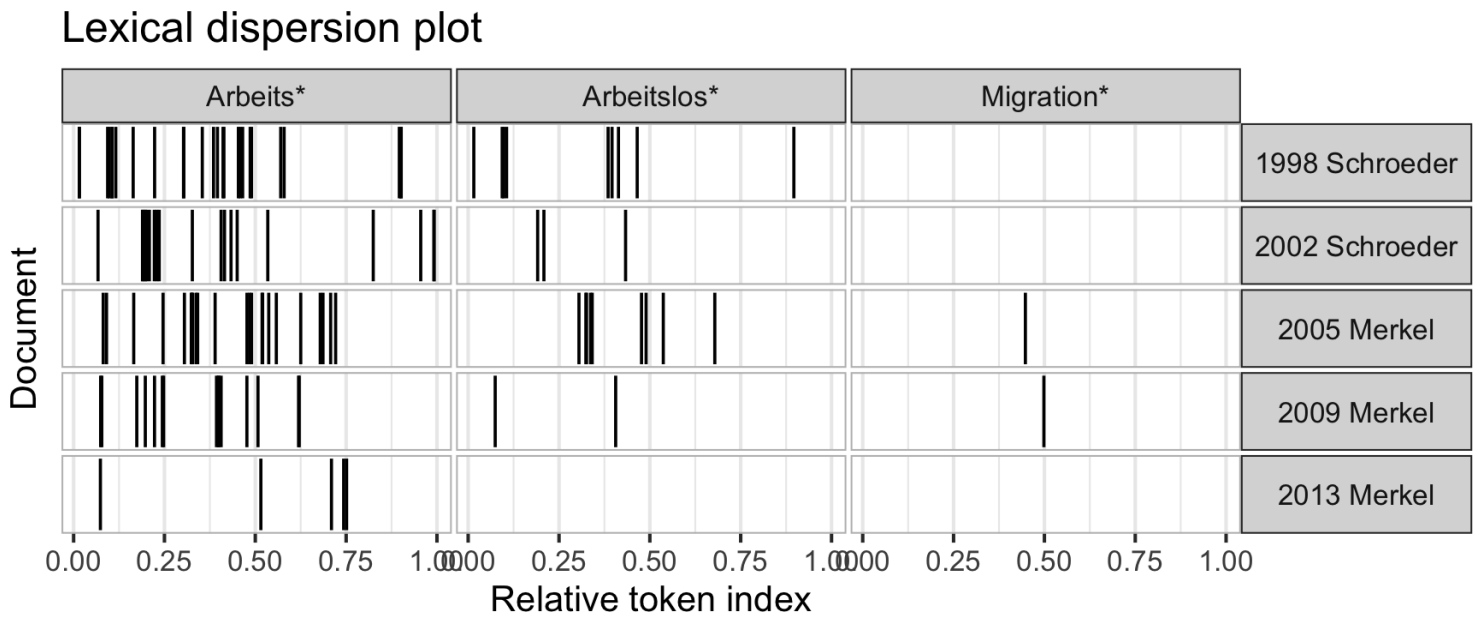
1. Create better `docnames` by pasting "Year" and "Chancellor"
2. Select speeches by Gerhard Schröder and Angela Merkel.
3. Check `?textplot_xray()`.
4. Choose words that might only appear in some of the speeches.

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*")
)
```

Solution (cont.)

```
plot_xray
```



Tokenization

Tokenization of a corpus

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

```
head(toks_immig[[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

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

```
head(toks_nopunct[[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

```
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)
```

```
##  [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_immig[[1]], 20)
```

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

```
head(toks_nopunct[[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(toks_nostopw[[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

```
toks_immig_select <- toks_immig %>%
  tokens_select(pattern = c("immig*", "migra*"),
                padding = TRUE)
head(toks_immig_select[[1]], 20)
```

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

```
toks_immig_no_pad <- toks_immig %>%
  tokens_select(pattern = c("immig*", "migra*"),
                padding = FALSE)
head(toks_immig_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
toks_immig_window <- tokens_select(toks_immig,
                                   pattern = c('immig*', 'migra*'),
                                   padding = TRUE, window = 5)
head(toks_immig_window[[1]], 20)
```

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

Compound multiwords expressions

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

```
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)
```

```
##
##  [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
```

Solution (cont.)

```
# 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]], 15)
```

```
##  [1] "happening"   "today"       "."           "It"          "is"
##  [6] "happening"   ","           "however"     ","           "because"
## [11] "Fianna_Fáil" "failed"      "to"          "heed"        "the"
```

Note: To get the sentences that mention one of the parties, call `corpus_reshape(to = "sentences")` first, repeat the subsequent steps and set a very large `window` (e.g., 1000).

```
# 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

```
# 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

```
# 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"
## [7] "ongoing_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, dictionary = 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

```
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

```
toks_ire <- tokens(data_corpus_irishbudget2010)

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

topfeatures(dfmat_ire)
```

```
##  the    .   to    ,   of  and   in    a   is that
## 3598 2371 1633 1548 1537 1359 1231 1013  868  804
```

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

topfeatures(dfmat_ire2)
```

```
##   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(dfmat_ire)
```

```
## [1] 5140
```

```
dfmat_ire_trim1 <- dfm_trim(dfmat_ire,
                        min_termfreq = 5)
nfeat(dfmat_ire_trim1)
```

```
## [1] 1265
```

```
dfmat_ire_trim2 <- dfm_trim(dfmat_ire,
                        min_docfreq = 5)
nfeat(dfmat_ire_trim2)
```

```
## [1] 864
```

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

```
## [1] 2716
```

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

```
dfmat_fruits <- dfm(texts)
dfm_weight(dfmat_fruits, 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(dfmat_fruits, 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
```

Solution (cont.)

```
dfm_weight(dfmat_fruits, 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
##   text1   0.2 0.2    0.2  0.2    0.2  0
##   text2   0.2 0      0.2  0      0.4  0.2
```

Sentiment analysis

Why are we analyzing Irish budget speeches?

Irish Wince as a Budget Proposal Cuts to the Bone

By **SARAH LYALL** DEC. 9, 2009

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
toks_dict <- data_corpus_irishbudget2010 %>%
    tokens() %>%
    tokens_lookup(dictionary = data_dictionary_LSD2015)

# transform to a dfm
dfmat_dict <- dfm(toks_dict)
```

Sentiment Analysis

```
print(dfmat_dict)
```

```
## 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(dfmat_dict, 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))
```

Plotting with ggplot2

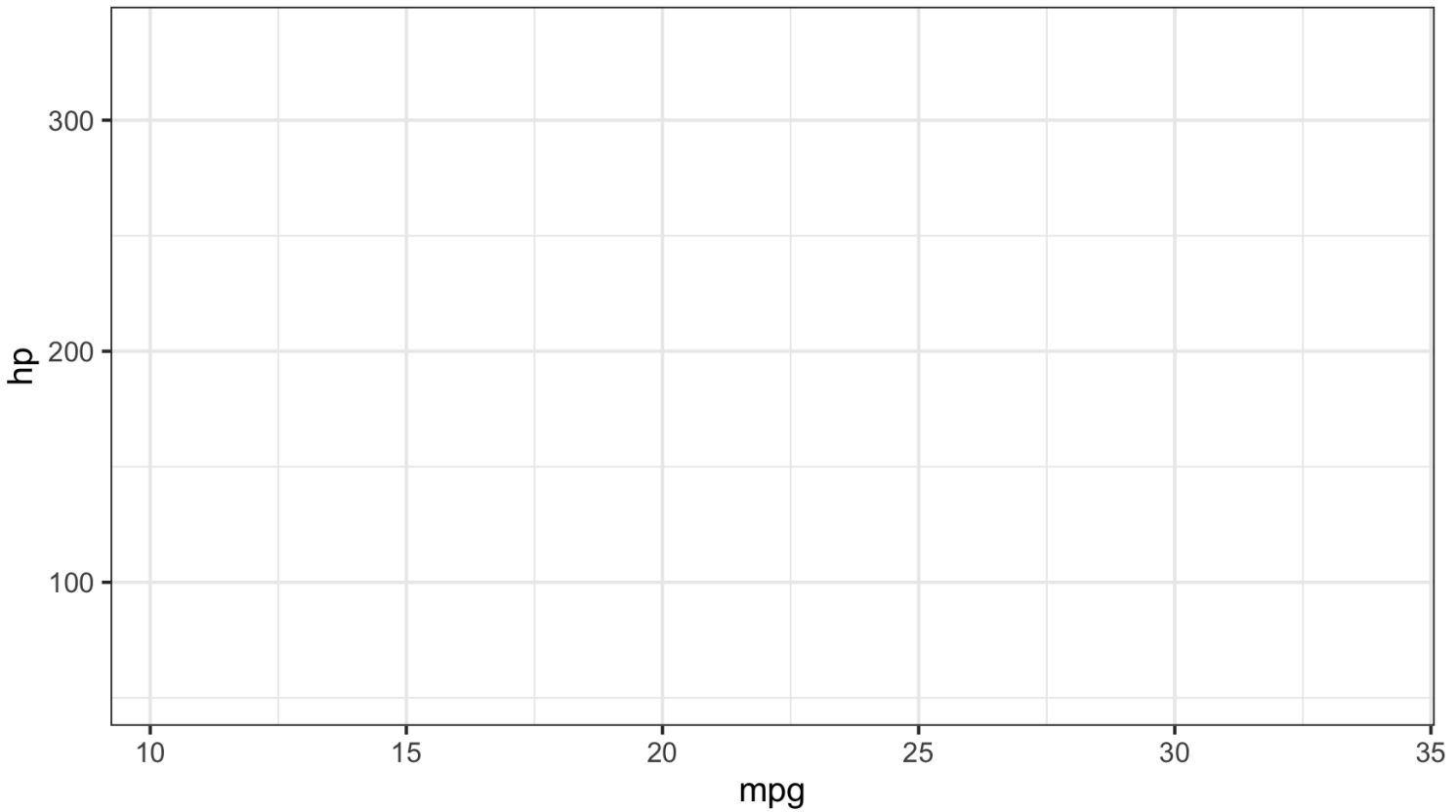
- Specify `data` (a data.frame)
- Specify the aestetics (x-axis, y-axis, colours, shapes etc.)
- Choose a geometric objects (e.g. scatterplot, boxplot)

```
# discover mtcars dataset
head(mtcars, 3)
```

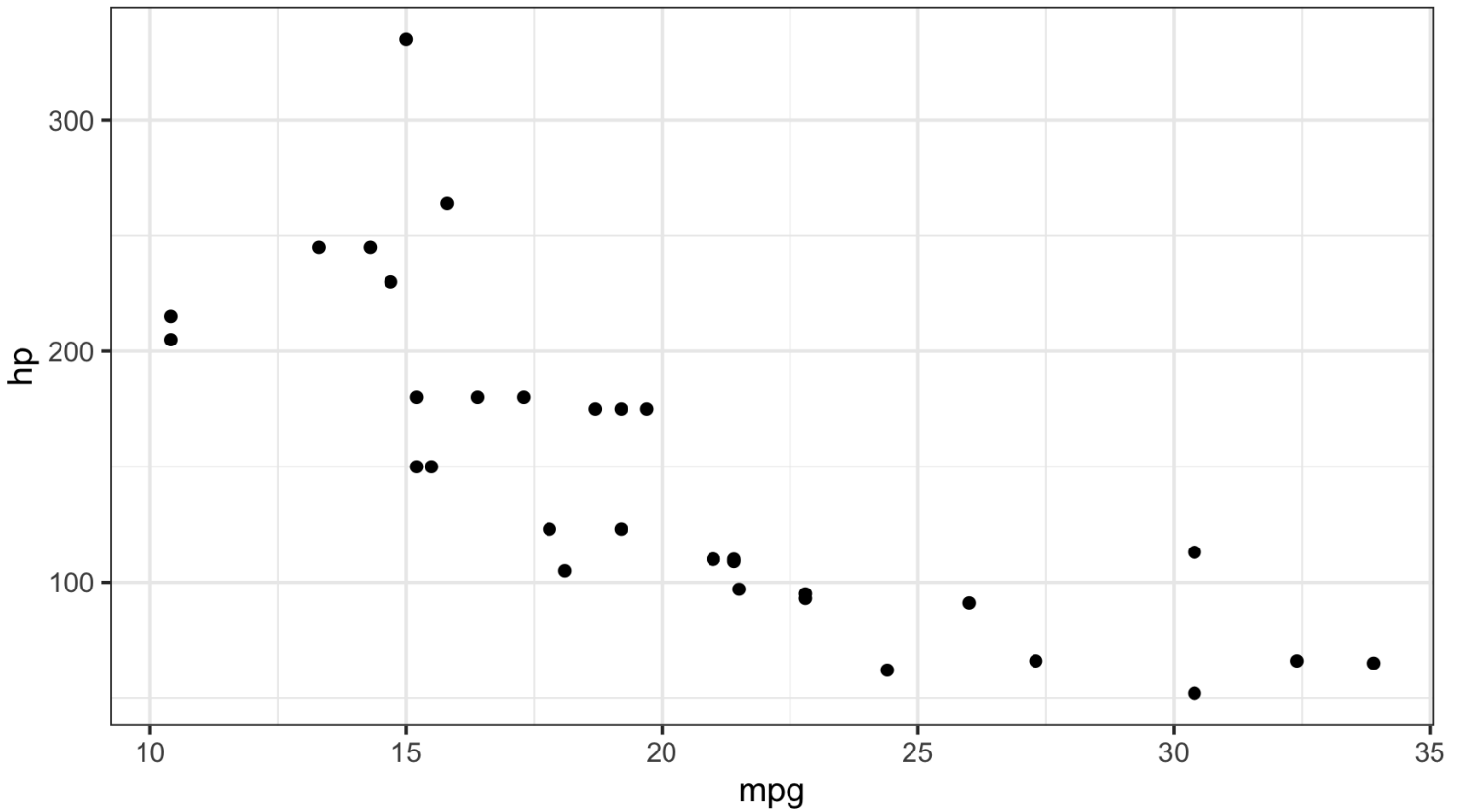
```
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

```
# specify data and axes
library(ggplot2)
myplot <- ggplot(data = mtcars,
                 aes(x = mpg, y = hp))
```

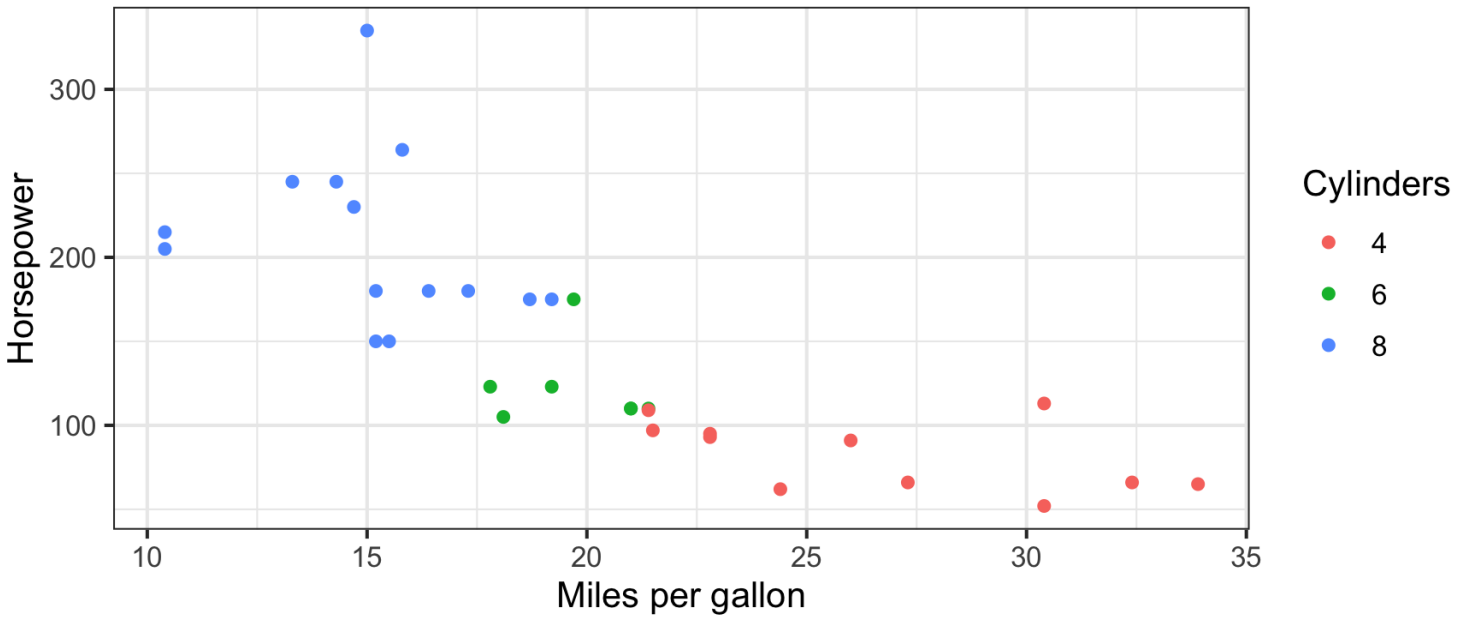
```
myplot
```



```
myplot +
  geom_point()
```



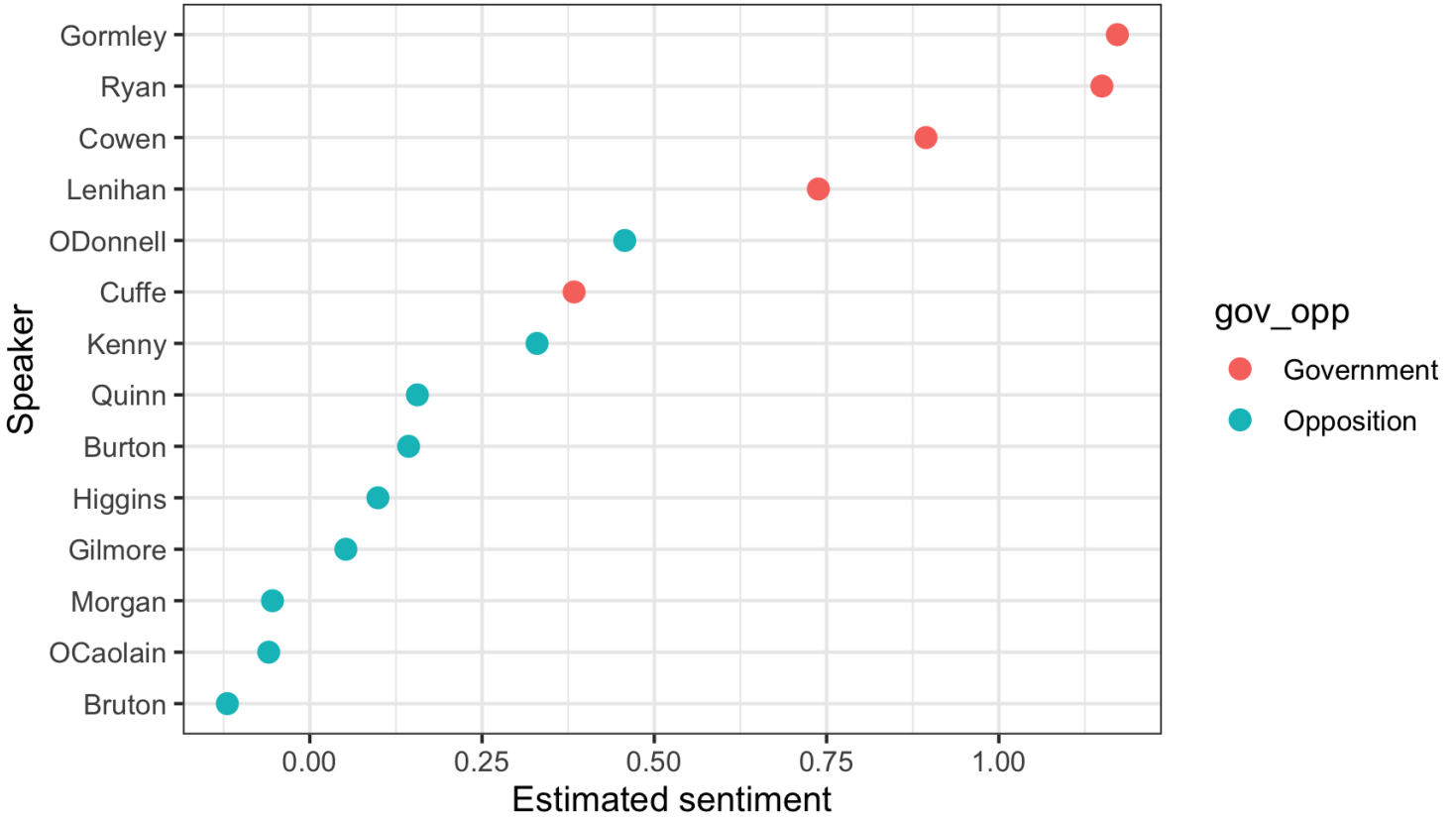
```
myplot +
  geom_point(aes(colour = factor(cyl))) +
  scale_colour_discrete(name = "Cylinders") +
  scale_shape_discrete(name = "Cylinders") +
  labs(x = "Miles per gallon",
       y = "Horsepower")
```



Plot sentiment

```
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
```

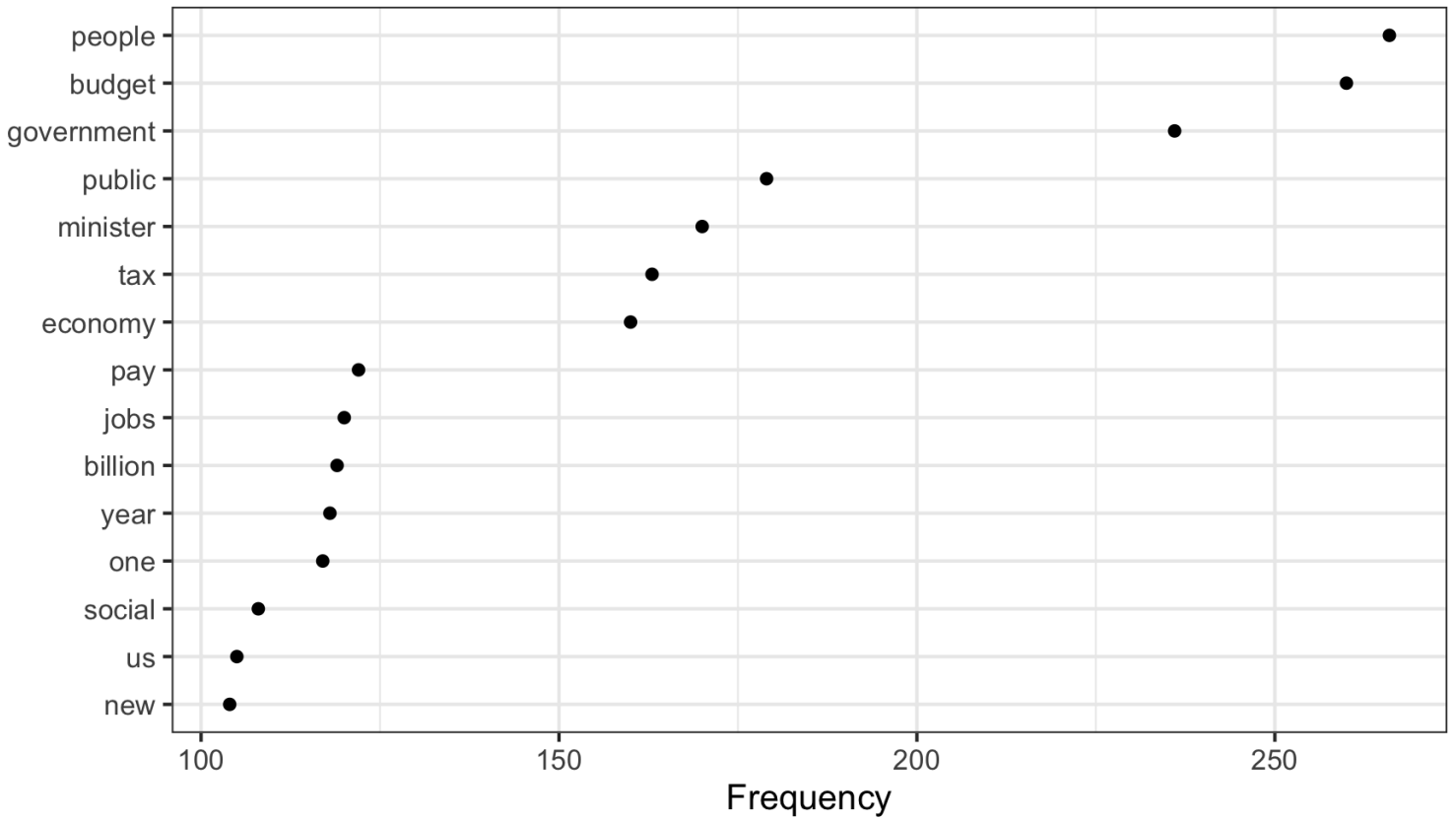

Gormley
Ryan
Cowen
Lenihan
ODonnell
Cuffe
Kenny
Quinn
Burton
Higgins
Gilmore
Morgan
OCaolain
Bruton
Speaker
0.00
0.25
0.50
0.75
1.00
Estimated sentiment
gov_opp
Government
Opposition

Textual statistics

Simple frequency analysis

```
library("ggplot2")
ire_dfm <- data_corpus_irishbudget2010 %>%
    dfm(remove = stopwords("en"), remove_punct = TRUE, remove_symbols = 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")
```

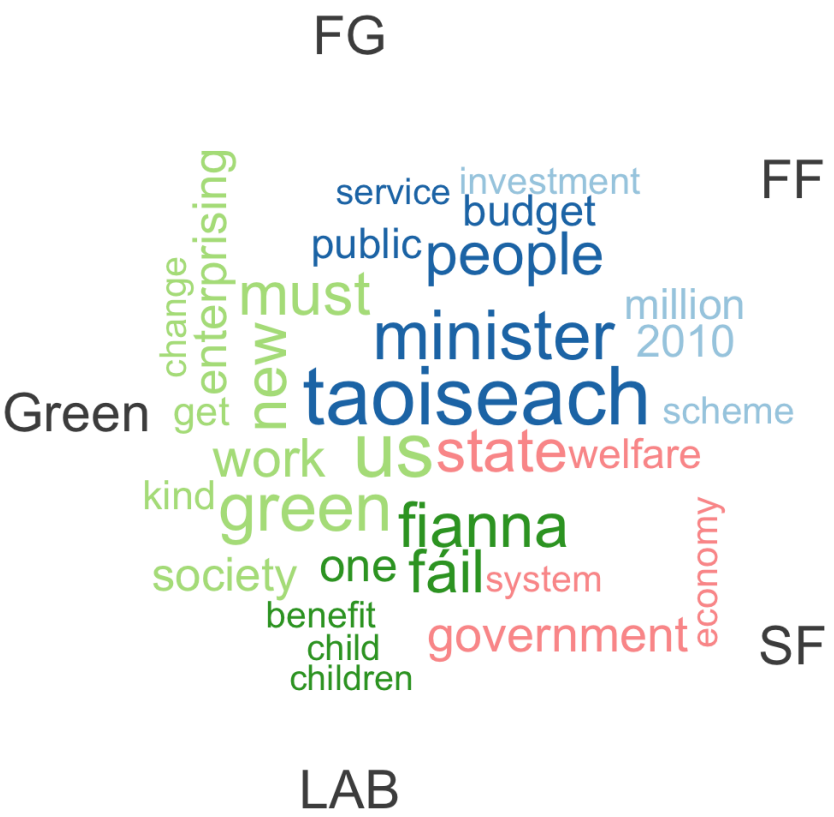
```
ire_freqplot
```



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)
```

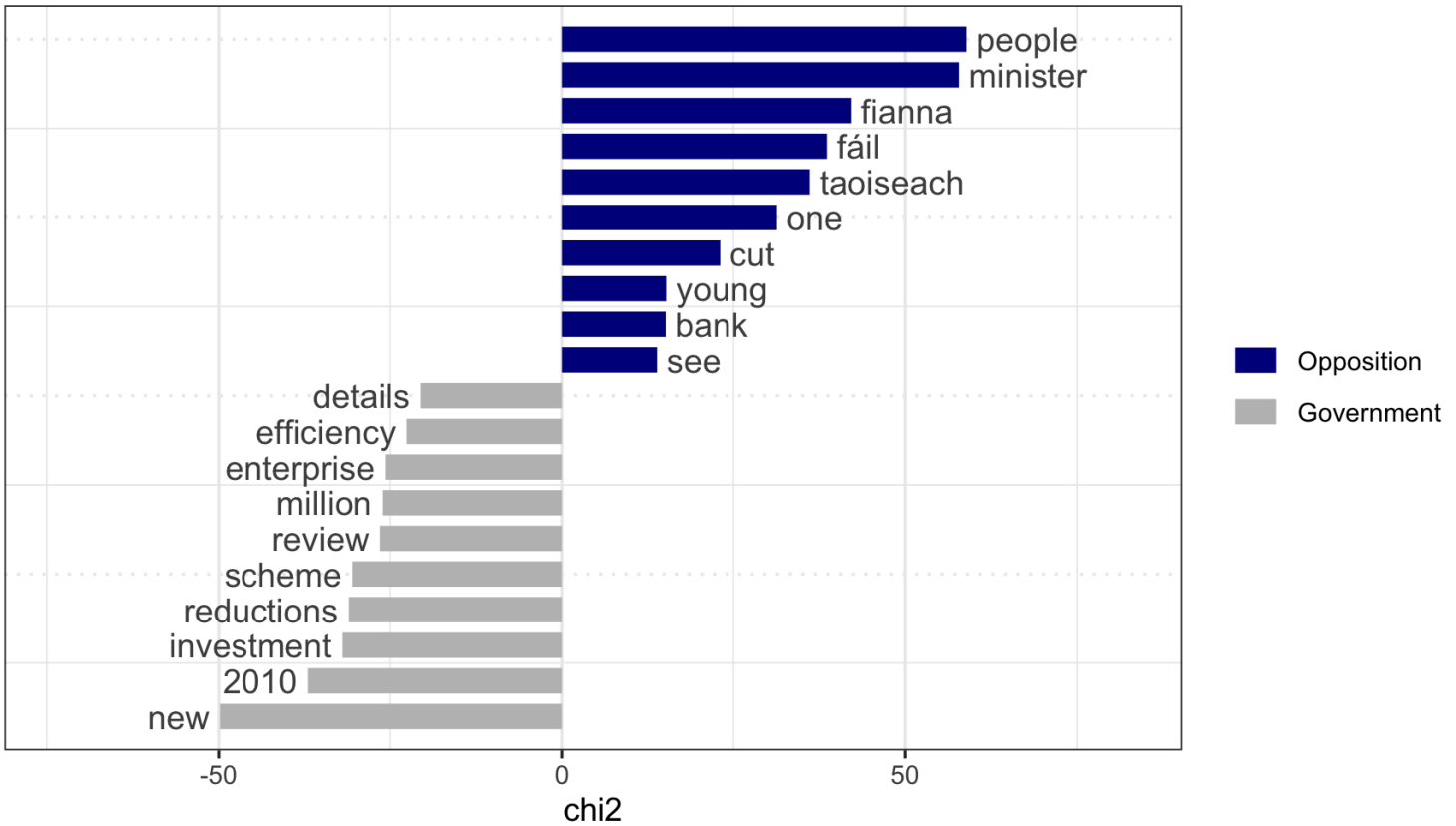


Relative frequency analysis (keyness)

"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
dfmat_key <- data_corpus_irishbudget2010 %>%
     dfm(groups = "gov_opp",
         remove = stopwords("english"),
         remove_punct = TRUE, remove_symbols = TRUE)

tstat_key <- textstat_keyness(dfmat_key,
                              target = "Opposition")

tplot_key <- textplot_keyness(tstat_key,
                            margin = 0.2,
                            n = 10)
```

```
tplot_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

Term frequency-inverse document frequency (tf-idf)

Example: We have 100 political party manifestos, each with 1000 words. The first document contains 16 instances of the word "environment"; 40 of the manifestos contain the word "environment".

- The *term frequency* is 16
- The *document frequency* is $100/40 = 2.5$, or $log(2.5) = 0.398$
- The *tf-idf* will then be $16 \times 0.398 = 6.37$
- If the word had only appeared only in 15 of the 100 manifestos, then the *tf-idf* would be 13.18
- tf-idf filters out common terms

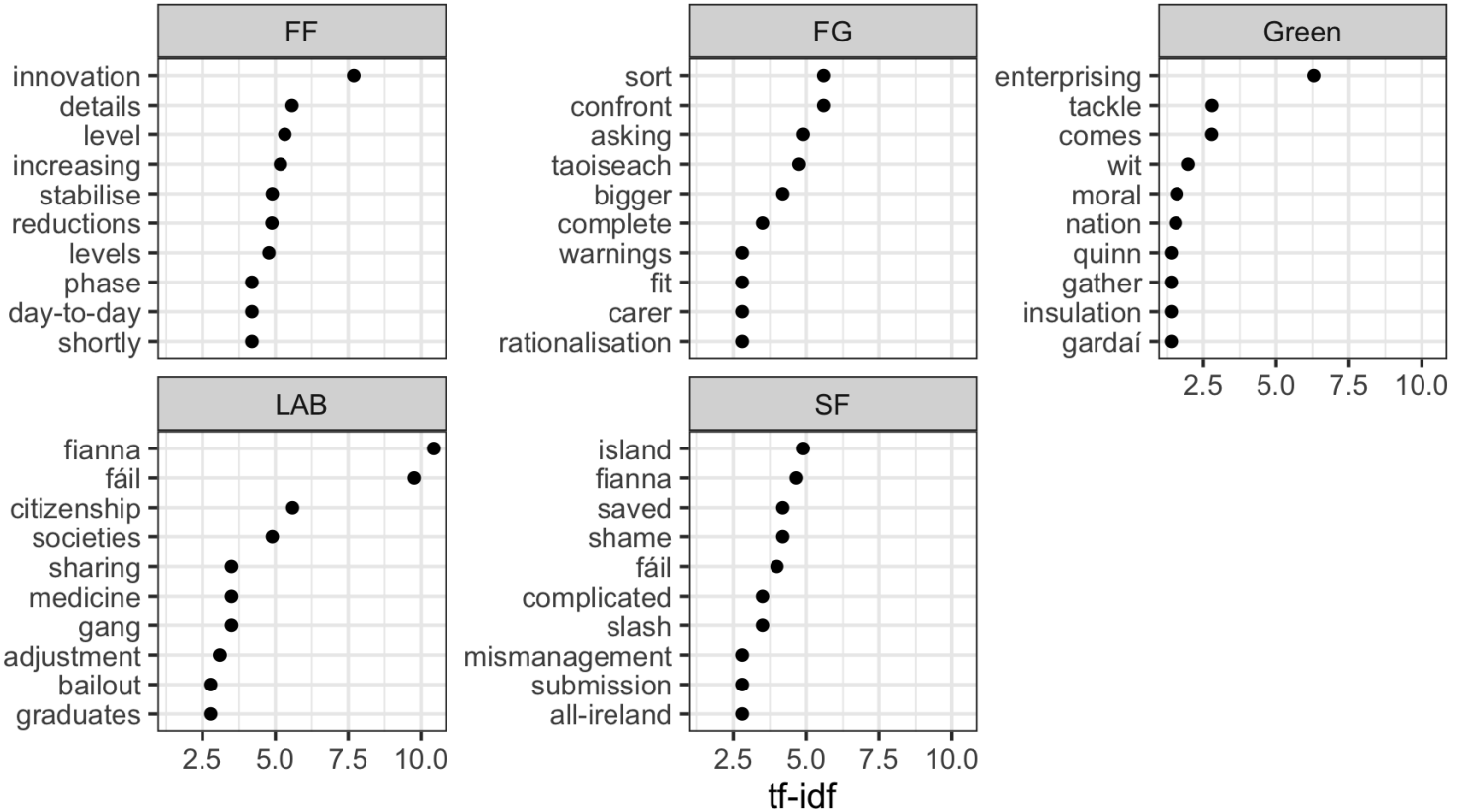
Apply tf-idf weighting

```
dfmat_ire_tfidf <- data_corpus_irishbudget2010 %>%
    dfm(remove_punct = TRUE, remove_numbers = TRUE, remove_symbols = TRUE) %>%
    dfm_group(groups = "party") %>%
    dfm_tfidf()
```

Plot tf-idf

```
tstat_freq_tfidf <- dfmat_ire_tfidf %>%
    textstat_frequency(n = 10, groups = "party", force = TRUE)

# plot frequencies
tplot_tfidf <- ggplot(data = tstat_freq_tfidf,
                      aes(x = factor(nrow(tstat_freq_tfidf):1),
                          y = frequency)) +
    geom_point() +
    facet_wrap(~ group, scales = "free_y") +
    coord_flip() +
    scale_x_discrete(breaks = factor(nrow(tstat_freq_tfidf):1),
                       labels = tstat_freq_tfidf$feature) +
    labs(x = NULL, y = "tf-idf")
```

```
tplot_tfidf
```


FF
innovation
details
level
increasing
stabilise
reductions
levels
phase
day-to-day
shortly
FG
sort
confront
asking
taoiseach
bigger
complete
warnings
fit
carer
rationalisation
Green
enterprising
tackle
comes
wit
moral
nation
quinn
gather
insulation
gardaí
LAB
fianna
fáil
citizenship
societies
sharing
medicine
gang
adjustment
bailout
graduates
SF
island
fianna
saved
shame
fáil
complicated
slash
mismanagement
submission
all-ireland
2.5
5.0
7.5
10.0
tf-idf

Collocation analysis: a strings of words approach

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

head(tstat_col, 7)
```

```
##      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 31.81818
## 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
```

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

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

tstat_col_adjusted <- toks_ire_adjusted %>%
    textstat_collocations(min_count = 5, tolower = FALSE)

head(tstat_col_adjusted, 7)
```

```
##     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
```

```
nrow(tstat_col_adjusted)
```

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

Compound collocations

Before compounding

```
toks_ire <- toks_ire_adjusted
kwic(toks_ire, phrase("Green Party")) %>%
    head(4)
```

```
## [1] pattern
## <0 rows> (or 0-length row.names)
```

Compound collocations

Compound based on collocations

```
# get 50 most frequent collocations

# compound based on collocations
toks_ire_comp <- toks_ire %>%
    tokens_compound(tstat_col_adjusted[tstat_col_adjusted$z > 3])
```

After compounding

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

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

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

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

Compound collocations

Check compounded words

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

```
##   fianna_fáil public_servic social_welfar     next_year child_benefit
##            61            47            41            38            30
## minist_financ      per_week public_sector   young_peopl   €_4_billion
##            30            25            25            23            20
```

Similarity between texts

Cosine similarity

$$cos(A, B) = \frac{A \cdot B}{||A|| \cdot ||B||}$$

$$cos(A, B) = \frac{\sum A_i B_i}{\sqrt{\sum A_i^2}\sqrt{\sum B_i^2}}$$

```
(dfmat <- dfm(c("X Y Y Z Z Z ", "X X X X Y Y Y Y Y Z Z Z Z Z Z")))
```

```
## Document-feature matrix of: 2 documents, 3 features (0.0% sparse).
## 2 x 3 sparse Matrix of class "dfm"
##        features
## docs    x y z
##   text1 1 2 3
##   text2 4 5 6
```

```
textstat_simil(dfmat, method = "cosine")
```

```
## textstat_simil object; method = "cosine"
##       text1 text2
## text1 1.000 0.975
```

Cosine similarity

```
library(quanteda)
dfmat <- dfm(c("I use this sentence almost twice.",
    "I include this almost sentence twice.",
    "And here we have irrelevant content."))

tstat_simil <- textstat_simil(dfmat, method = "cosine")
tstat_simil
```

```
## textstat_simil object; method = "cosine"
##       text1 text2 text3
## text1 1.000 0.857 0.143
## text2 0.857 1.000 0.143
## text3 0.143 0.143 1.000
```

Why is the similarity between `text1`/`text2` and `text3` not 0?

BONUS: Transform matrix into dyadic dataframe

```
tstat_simil_matrix <- as.matrix(tstat_simil)
tstat_simil_matrix[lower.tri(tstat_simil_matrix)] <- NA

tstat_simil_df <- tstat_simil_matrix %>%
    reshape2::melt() %>%
    subset(Var1 != Var2) %>%
    subset(!is.na(value))

tstat_simil_df
```

```
##    Var1  Var2     valu
## 4 text1 text2 0.857142
## 7 text1 text3 0.1428571
## 8 text2 text3 0.1428571
```

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
tstat_simil1 <- data_corpus_irishbudget2010 %>%
    dfm() %>%
    dfm_group(groups = "party") %>%
    textstat_simil(method = "cosine",
                   margin = "documents")

tstat_simil1
```

```
## textstat_simil object; method = "cosine"
##          FF    FG Green   LAB    SF
## FF    1.000 0.954 0.945 0.963 0.968
## FG    0.954 1.000 0.950 0.983 0.979
## Green 0.945 0.950 1.000 0.951 0.938
## LAB   0.963 0.983 0.951 1.000 0.980
## SF    0.968 0.979 0.938 0.980 1.000
```

Solution

```
tstat_simil2 <- data_corpus_irishbudget2010 %>%
    dfm(remove = stopwords("en"),
        stem = TRUE, remove_punct = TRUE) %>%
    dfm_group(groups = "party") %>%
    textstat_simil(method = "cosine",
                   margin = "documents")

tstat_simil2
```

```
## textstat_simil object; method = "cosine"
##          FF    FG Green   LAB    SF
## FF    1.000 0.621 0.657 0.666 0.695
## FG    0.621 1.000 0.645 0.795 0.791
## Green 0.657 0.645 1.000 0.655 0.651
## LAB   0.666 0.795 0.655 1.000 0.810
## SF    0.695 0.791 0.651 0.810 1.000
```

Lexical diversity

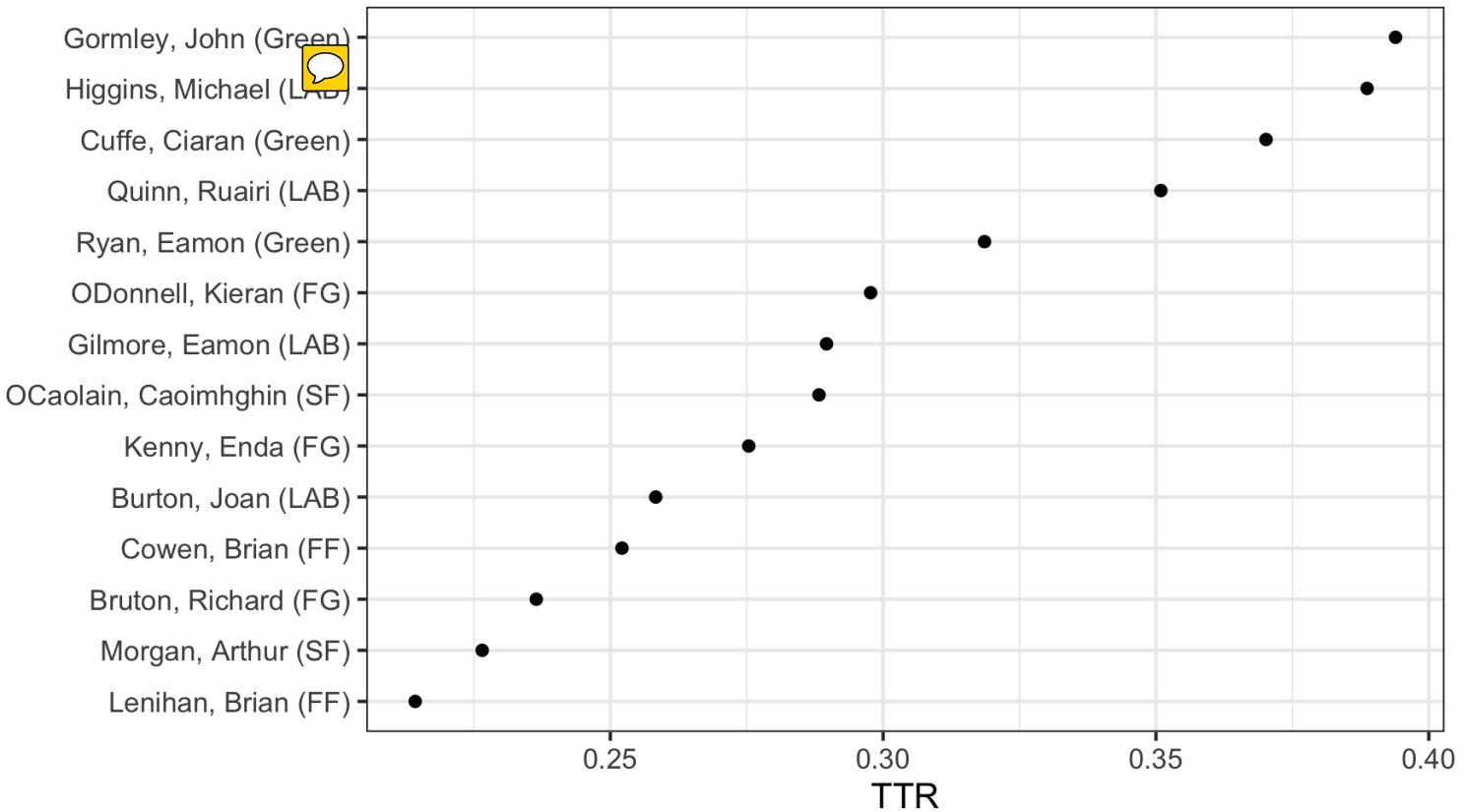
```
dfmat_ire <- data_corpus_irishbudget2010 %>%
    dfm(remove_punct = TRUE, remove_numbers = TRUE, remove_symbols = TRUE)

# estimate Type-Token Ratio
tstat_lexdiv <- textstat_lexdiv(dfmat_ire, measure = "TTR")

df_lexdiv <- cbind(tstat_lexdiv,
                   docvars(dfmat_ire))

head(df_lexdiv, 4)
```

```
##                                 document       TTR year debate number
## Lenihan, Brian (FF)   Lenihan, Brian (FF) 0.2142487 2010 BUDGET     01
## Bruton, Richard (FG) Bruton, Richard (FG) 0.2364312 2010 BUDGET     02
## Burton, Joan (LAB)     Burton, Joan (LAB) 0.2583187 2010 BUDGET     03
## Morgan, Arthur (SF)   Morgan, Arthur (SF) 0.2265236 2010 BUDGET     04
##                          foren    name party    gov_opp
## Lenihan, Brian (FF)     Brian Lenihan    FF Government
## Bruton, Richard (FG) Richard  Bruton    FG Opposition
## Burton, Joan (LAB)       Joan  Burton   LAB Opposition
## Morgan, Arthur (SF)    Arthur  Morgan    SF Opposition
```

Plot Type-Token Ratio

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

```
tplot_ttr
```

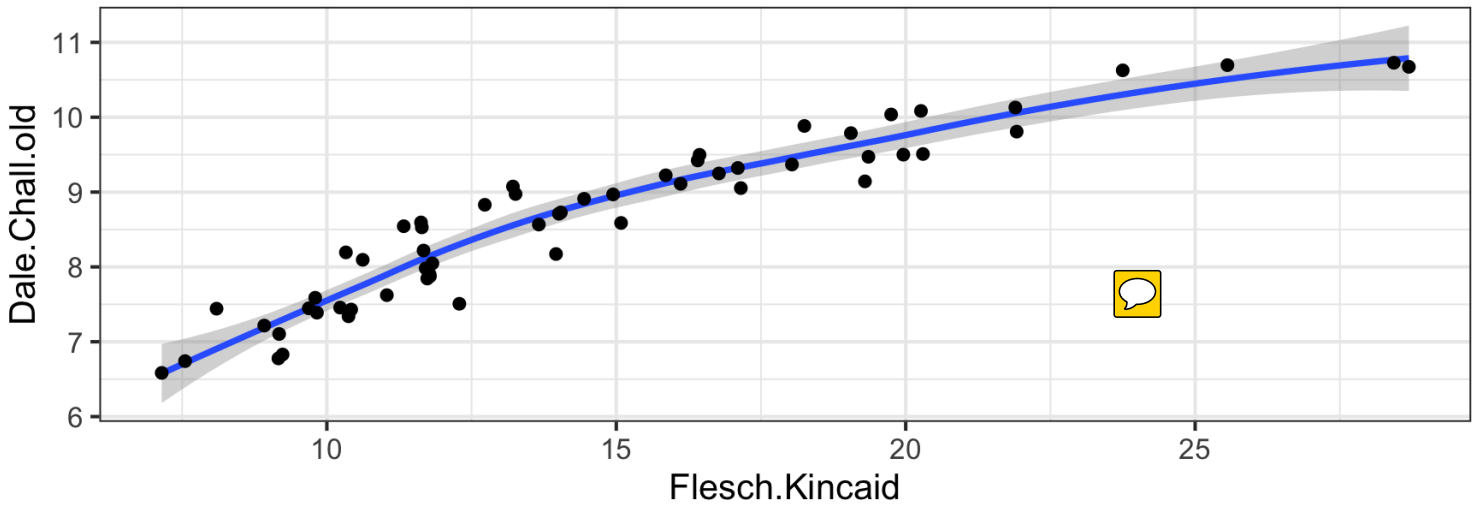

Gormley, John (Green)
Higgins, Michael (LAB)
Cuffe, Ciaran (Green)
Quinn, Ruairi (LAB)
Ryan, Eamon (Green)
ODonnell, Kieran (FG)
Gilmore, Eamon (LAB)
OCaolain, Caoimhghin (SF)
Kenny, Enda (FG)
Burton, Joan (LAB)
Cowen, Brian (FF)
Bruton, Richard (FG)
Morgan, Arthur (SF)
Lenihan, Brian (FF)
0.25
0.30
0.35
0.40
TTR

Readability

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

tplot_read <- ggplot(tstat_read, aes(x = Flesch.Kincaid,
                                     y = Dale.Chall.old)) +
    geom_smooth() +
    geom_point()
```

Plot relationship between readability measures

```
tplot_read
```



Correlation between readability measures

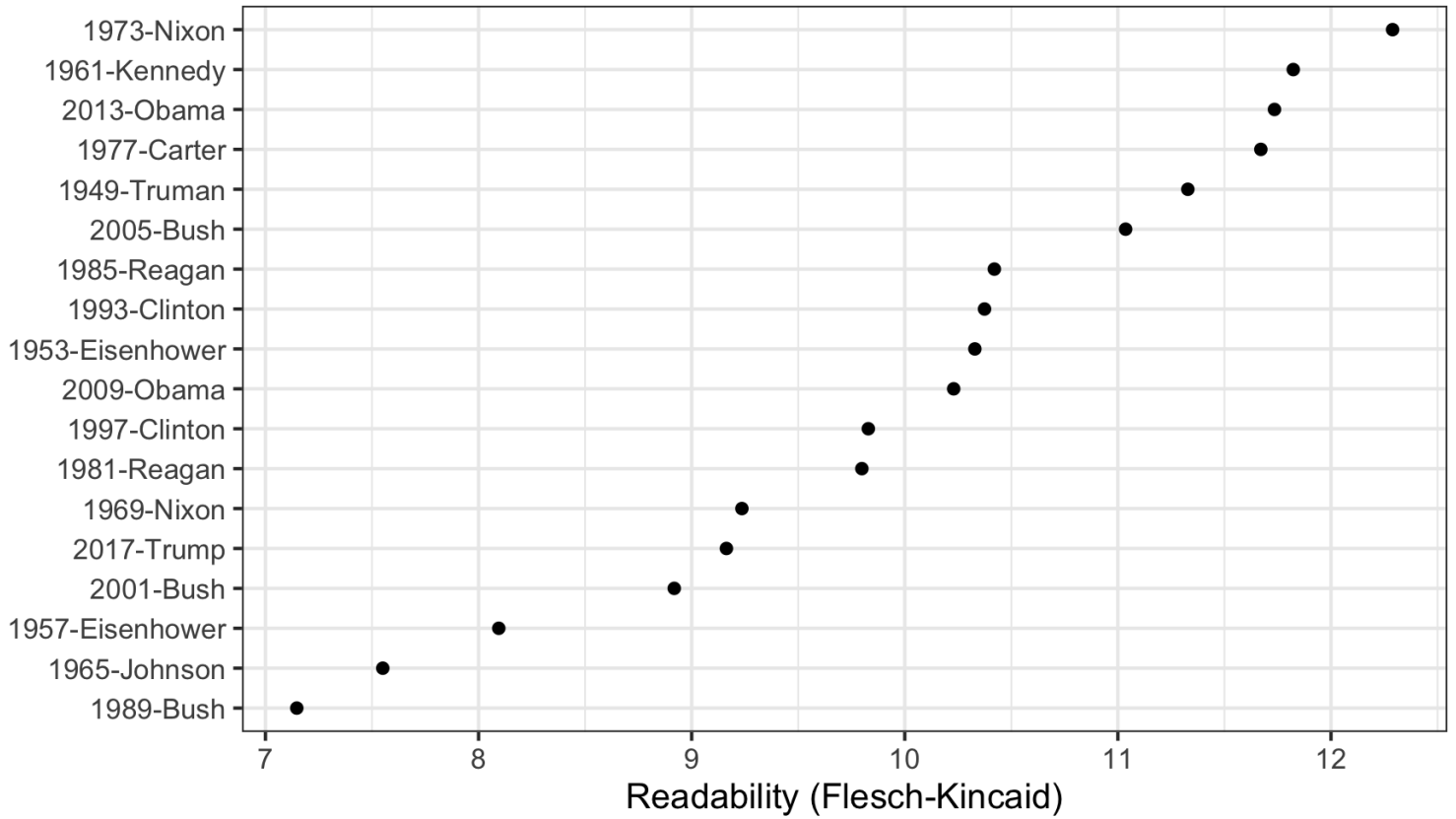
```
cor.test(tstat_read$Flesch.Kincaid, tstat_read$Dale.Chall.old)
```

```
##
##     Pearson's product-moment correlation
##
## data:  tstat_read$Flesch.Kincaid and tstat_read$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
```

Readability of inaugural speeches since 1945

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

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

```
tplot_read_us
```



See extensively Benoit et al. (2019)

Exercise: Wrapping it all up

1. Use `data_corpus_guardian` from the **quanteda.corpora** package using: `data_corpus_guardian <- quanteda.corpora::download('data_corpus_guardian')`.
2. Create a `tokens` object and remove stopwords and punctuation
3. Keep only the term "Brexit*" and a window of 5 words.
4. Use `fcm()` and create a feature co-occurence matrix.
5. Use `textplot_network()` to plot a network of co-occurences of the 40 most frequent terms.

Solution

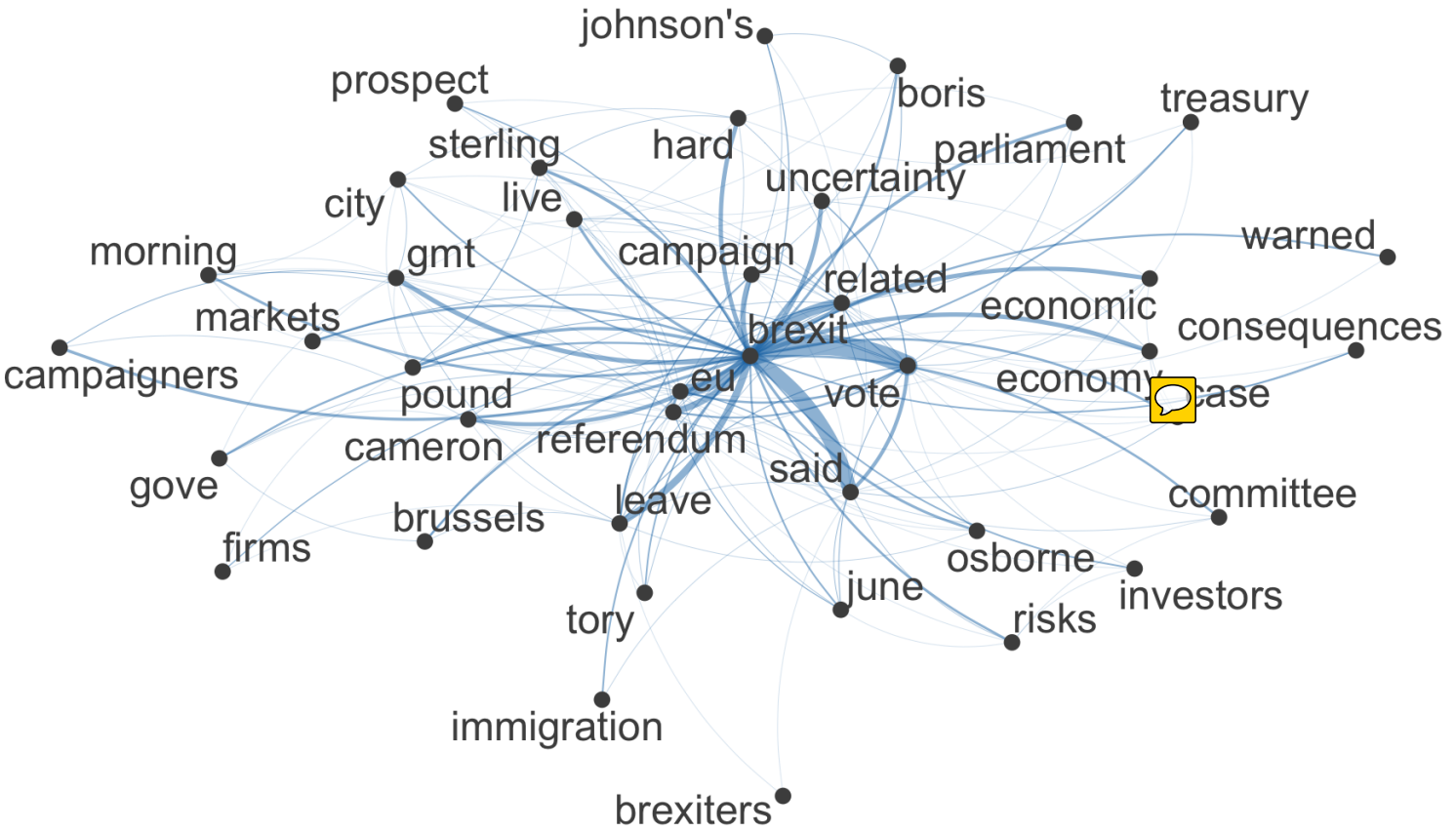
Import Guardian corpus using **quanteda.corpora**'s `download()` function.

```
data_corpus_guardian <- download('data_corpus_guardian')
```

```
toks_brexit <- data_corpus_guardian %>%
    tokens(remove_punct = TRUE, remove_symbols = TRUE) %>%
    tokens_remove(stopwords("en"),
                  padding = FALSE) %>%
    tokens_keep(pattern = "Brexit*",
                window = 5,
                case_insensitive = TRUE) %>%
    tokens_tolower()

fcmat_brexit <- toks_brexit %>%
    fcm(context = "window")

feat <- names(topfeatures(fcmat_brexit, 40))

fcmat_brexit_subset <- fcmat_brexit %>%
    fcm_select(feat)

tplot_brexit <- textplot_network(fcmat_brexit_subset)
```

```
## Registered S3 method overwritten by 'network':
##   method            from
##   summary.character quanteda
```

```
set.seed(134)
tplot_brexit
```



Supervised text classification

Supervised classification: separate documents into pre-existing categories

We need:

1. Hand-coded dataset (labeled), to be split into a *training set* (used to train the classifier) and a *validation/test set* (used to validate the classifier)
2. Method to extrapolate from hand coding to unlabeled documents (classifier): Naive Bayes, regularized regression, SVM, K-nearest neighbours, ensemble methods
3. Approach to validate classifier: cross-validation
4. Performance metric to choose best classifier and avoid overfitting: confusion matrix, accuracy, precision, recall, F1 scores

Goals of supervised classification

1. Flexibility
2. Efficiency
3. Interpretability

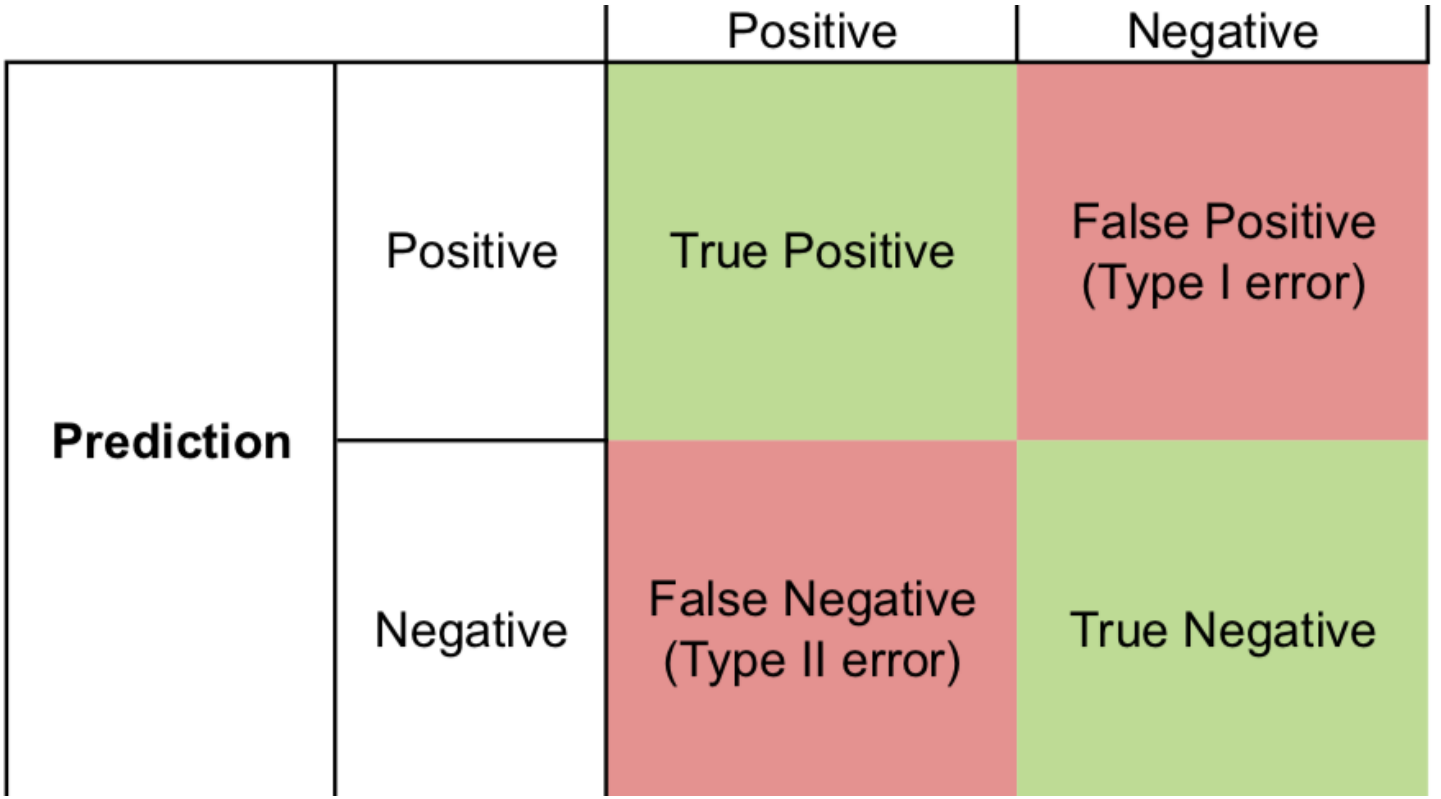
Unsupervised classification

- Unsupervised methods scale documents based on patterns of similarity from the term-document matrix, without requiring a training step
- Examples: Wordfish, topic models (to be covered soon)
- Relative advantage: You do not have to know the dimension being scaled (also a disadvantage!)

Basic principles of supervised learning

1. **Generalisation**: A classifier learns to correctly predict output from given inputs not only in previously seen samples but *also in previously unseen samples*
2. **Overfitting**: A classifier learns to correctly predict output from given inputs in previously seen samples but fails to do so in previously unseen samples. This causes poor prediction/generalisation.

How to assess the performance of a classifier?



Important measures

- **Accuracy**: $\frac{Correctly\ classified}{Total\ number\ of\ cases} = \frac{true\ positives + true\ negatives}{Total\ number\ of\ cases}$
- **Precision**: $\frac{true\ positives}{true\ positives + false\ positives}$
- **Recall**: $\frac{true\ positives}{true\ positives + false\ negatives}$
- **F1 score**: $2 \times \frac{Precision \times Recall}{Precision + Recall}$

How do we get a labelled training set?

1. External sources of annotation
2. Expert annotation
3. Crowd-sourced coding
    - Wisdom of crowds -- aggregated judgments by non-experts converge to judgments of experts, depending on difficulty of classification taks (Benoit et al. 2016)
    - Platforms: *Figure 8* (previously called *CrowdFlower*) or Amazon's *MTurk*

Naive Bayes: Probabilistic learning

Intuition: If we observe the term "fantastic"" in a text, how likely is this text a positive review?

1. Determine frequency of term in positive and negative reviews (prior).
2. Assess proability of features given a particular class.
3. Get probabiliy of a document belonging to each class (posterior).
4. Which posterior is highest?

Naive Bayes

Advantages

- Simple, fast, effective
- Relatively small training set required (if classes no too imbalanced)
- Easy to obtain probabilities

Disadvantages

- Assumption of conditional independence is problematic
- If feature is not in training set, it is disregarded for the classification

Naive Bayes in quanteda: Example 1

```
# 1. get training set
dfmat_train <- dfm(c("positive bad negative horrible",
                     "great fantastic nice"))
class <- c("neg", "pos")

# 2. train model
tmod_nb <- textmodel_nb(dfmat_train, class)

# 3. get unlabelled test set
dfmat_test <- dfm(c("bad horrible negative awful",
                    "nice bad great",
                    "great fantastic"))

# 4. predict class
predict(tmod_nb, dfmat_test, force = TRUE)
```

```
## Warning: 1 feature in newdata not used in prediction.
```

```
## text1 text2 text3
##   neg   pos   pos
## Levels: neg pos
```

Naive Bayes in quanteda: Example 2

Goal: predict whether US inaugural speech was delivered before or after 1945

```
# create unique ID
docvars(data_corpus_inaugural, "id") <- 1:ndoc(data_corpus_inaugural)

# create dummy (before/after 1945)
docvars(data_corpus_inaugural, "pre_post1945") <-
    ifelse(docvars(data_corpus_inaugural, "Year") > 1945, "Post 1945", "Pre 1945")

# sample 35 speeches
set.seed(123)
speeches_select <- sample(1:58, size = 35, replace = FALSE)

# create training and test sets
dfmat_train <- data_corpus_inaugural %>%
    corpus_subset(id %in% speeches_select) %>% # speech in id
    dfm()

dfmat_test <- data_corpus_inaugural %>%
    corpus_subset(!id %in% speeches_select) %>% # speech not in id
    dfm()
```

Naive Bayes in quanteda: Example 2

```
# number of documents in training set
ndoc(dfmat_train)
```

```
## [1] 35
```

```
# train Naive Bayes model
tmod_nb <- textmodel_nb(dfmat_train, y = docvars(dfmat_train, "pre_post1945"))

# number of documents in test set
ndoc(dfmat_test)
```

```
## [1] 23
```

```
# predict class of held-out test set
tmod_nb_predict <- predict(tmod_nb, newdata = dfmat_test,
                           force = TRUE) #set force to TRUE or use dfm_match
```

```
## Warning: 1268 features in newdata not used in prediction.
```

Naive Bayes in quanteda: Example 2

```
# create cross-table
tab <- table(actual = docvars(dfmat_test, "pre_post1945"),
             predicted = tmod_nb_predict)

print(tab)
```

```
##            predicted
## actual      Post 1945 Pre 1945
##   Post 1945         6        0
##   Pre 1945          3       14
```

Naive Bayes in quanteda: Example 2

Assess accuracy using the **caret** package

```
library(caret)
performance <- caret::confusionMatrix(tab)

# get measures of classification performance
performance$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.6666667            1.0000000            1.0000000
##       Neg Pred Value            Precision               Recall
##            0.8235294            1.0000000            0.6666667
##                   F1           Prevalence       Detection Rate
##            0.8000000            0.3913043            0.2608696
## Detection Prevalence    Balanced Accuracy
##            0.2608696            0.8333333
```

Topic models

Required packages: **topicmodels** and **stm**

```
install.packages("stm")
install.packages("topicmodels")
```

```
packageVersion("topicmodels")
```

```
## [1] '0.2.8'
```

```
packageVersion("stm")
```

```
## [1] '1.3.3'
```

What are topic models?

- Algorithm to find most important "themes/frames/topics" in a corpus
- Further information or training data not necessary (entirely unsupervised method)
- Researcher only needs to specify *number* of topics (more difficult than it sounds!)

Examples of a study using topic models

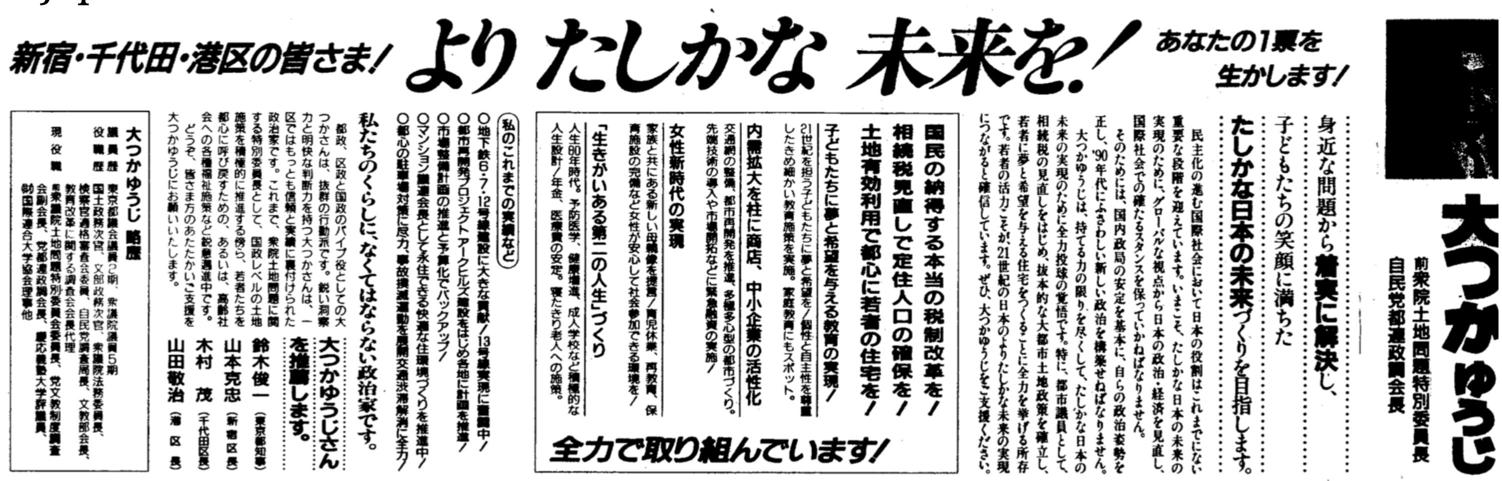
Catalinac (2016)

Amy Catalinac (2016). From Pork to Policy: The Rise of Programmatic Campaigning in Japanese Elections. The Journal of Politics 78 (1): 1–18.

- Japanese electoral system
  - Prior to 1994: Single-nontransferable-vote in multimember districts (SNTV-MMD)
  - Since 1994: mixed-member majoritarian (MMM) electoral system
- Expectatations:
  - More pork-barrel politics under SNTV-MMD
  - Under SNTV-MMD, candidates facing higher levels of intraparty competition relied more on pork-barrel politics

Catalinac (2016): Research design

A Japanese candidate manifesto



新宿・千代田・港区の皆さま！ より たしかな 未来を！ あなたの1票を生かします！

前衆院土地問題特別委員長
自民党都連政調会長
大つかゆうじ

身近な問題から着実に解決し、
子どもたちの笑顔に満ちた
たしかな日本の未来づくりを目指します。

民主化の進む国際社会において日本の役割はこれまでにない重要な段階を迎えています。いまこそ、たしかな日本の未来の実現のために、グローバルな視点から日本の政治・経済を見直し、国際社会での確たるスタンスを保っていかねばなりません。
そのためには、国内政局の安定を基本に、自らの政治姿勢を正し、'90年代にふさわしい新しい政治を構築せねばなりません。
大つかゆうじは、持てる力の限りを尽くして、たしかな日本の未来の実現のために全力投球の覚悟です。特に、都市議員として、相続税の見直しをはじめ、抜本的な大都市土地政策を確立し、若者に夢と希望を与える住宅をつくることに全力を挙げる所存です。若者の活力こそが、21世紀の日本のよりたしかな未来の実現につながると確信しています。ぜひ、大つかゆうじをご支援ください。

国民の納得する本当の税制改革を！
相続税見直しで定住人口の確保を！
土地有効利用で都心に若者の住宅を！

子どもたちに夢と希望を与える教育の実現！
21世紀を担う子どもたちに夢と希望を／個性と自主性を尊重したきめ細かい教育施策を実施。家庭教育にもスポット。

内需拡大を柱に商店、中小企業の活性化
交通網の整備、都市再開発を推進、多極多心型の都市づくり。先端技術の導入や市場開拓などに緊急融資の実施／

女性新時代の実現
家族と共にある新しい母親像を提言／育児休業、再教育、保育施設の完備など女性が安心して社会参加できる環境を／

「生きがいある第二の人生」づくり
人生80年時代。予防医学、健康増進、成人学校など積極的な人生設計／年金、医療費の安定。寝たきり老人への施策。

全力で取り組んでいます！

私のこれまでの実績など
○地下鉄6・7・12号線建設に大きな貢献／13号線実現に奮闘中／
○都市再開発プロジェクトアークヒルズ建設をはじめ各地に計画を推進／
○市場整備計画の推進と予算化でバックアップ／
○マンション議連会長として永住できる快適な住環境づくりを推進中／
○都心の駐車場対策に尽力、事故撲滅運動を展開交通渋滞解消に全力／

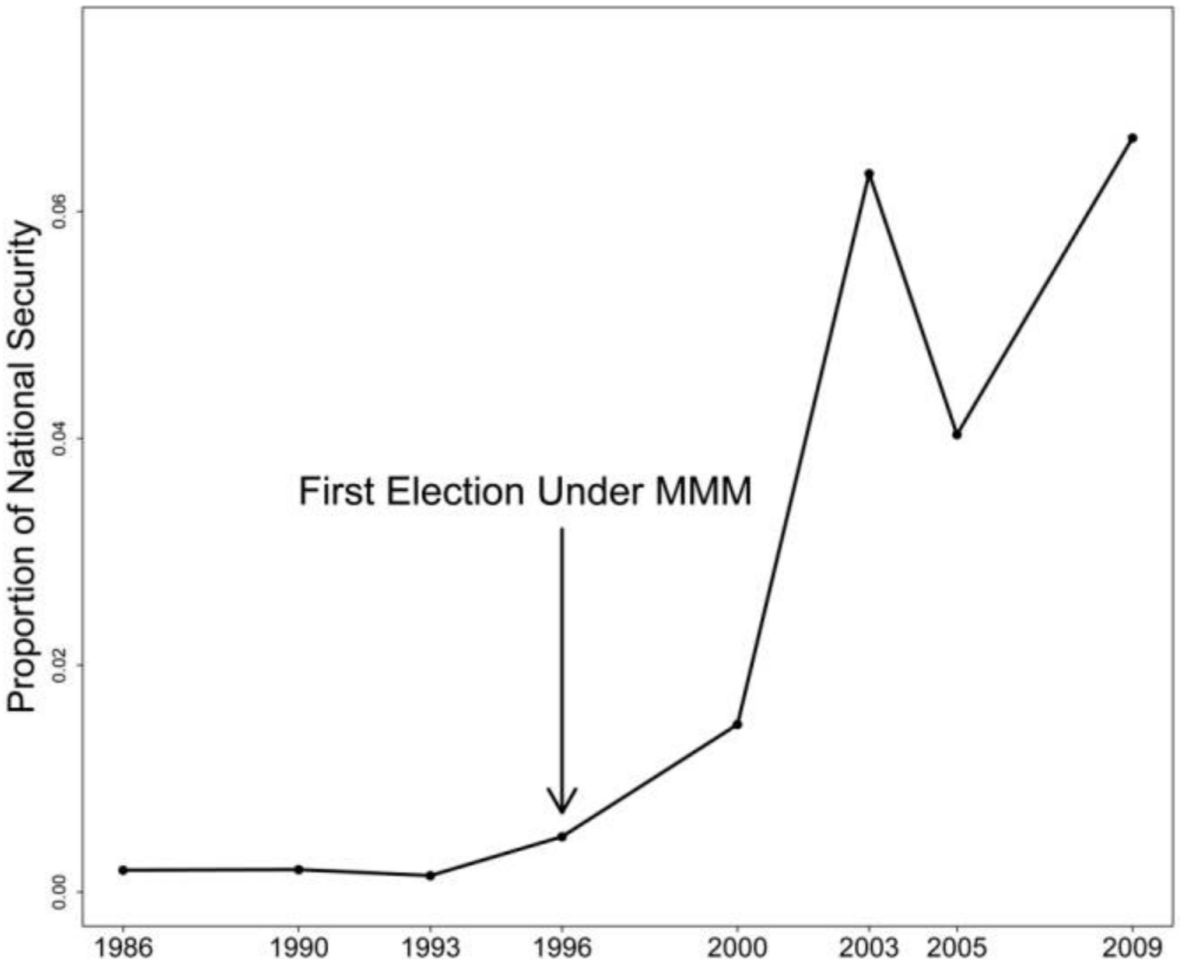
私たちのくらしに、なくてはならない政治家です。
都政、区政と国政のパイプ役としての大つかさんは、抜群の行動派です。鋭い洞察力と明快な判断力を持つ大つかさんは、一区ではもっとも信頼と実績に裏付けられた政治家です。これまで、衆院土地問題に関する特別委員長として、国政レベルの土地施策を積極的に推進する傍ら、若者たちを都心に呼び戻すための、あるいは、高齢社会への各種福祉施策など鋭意邁進中です。
どうぞ、皆さま方のあたたかいご支援を大つかゆうじにお願いいたします。

大つかゆうじさんを推薦します。
鈴木俊一（東京都知事）
山本克忠（新宿区長）
木村　茂（千代田区長）
山田敬治（港区長）

大つかゆうじ 略歴
議員歴　東京都議会議員2期、衆議院議員5期
役職歴　国土政務次官、文部政務次官、衆議院法務委員長、検察官適格審査会委員、自民党調査局長、文教部会長、教育改革に関する調査会会長代理
現役職　前衆議院土地問題特別委員会委員長、党文教制度調査会副会長、党都連政調会長、慶応義塾大学評議員、(財)国際連合大学協会理事他

Catalinac (2016): Results

Catalinac (2016): Results



Catalinac (2016): Results

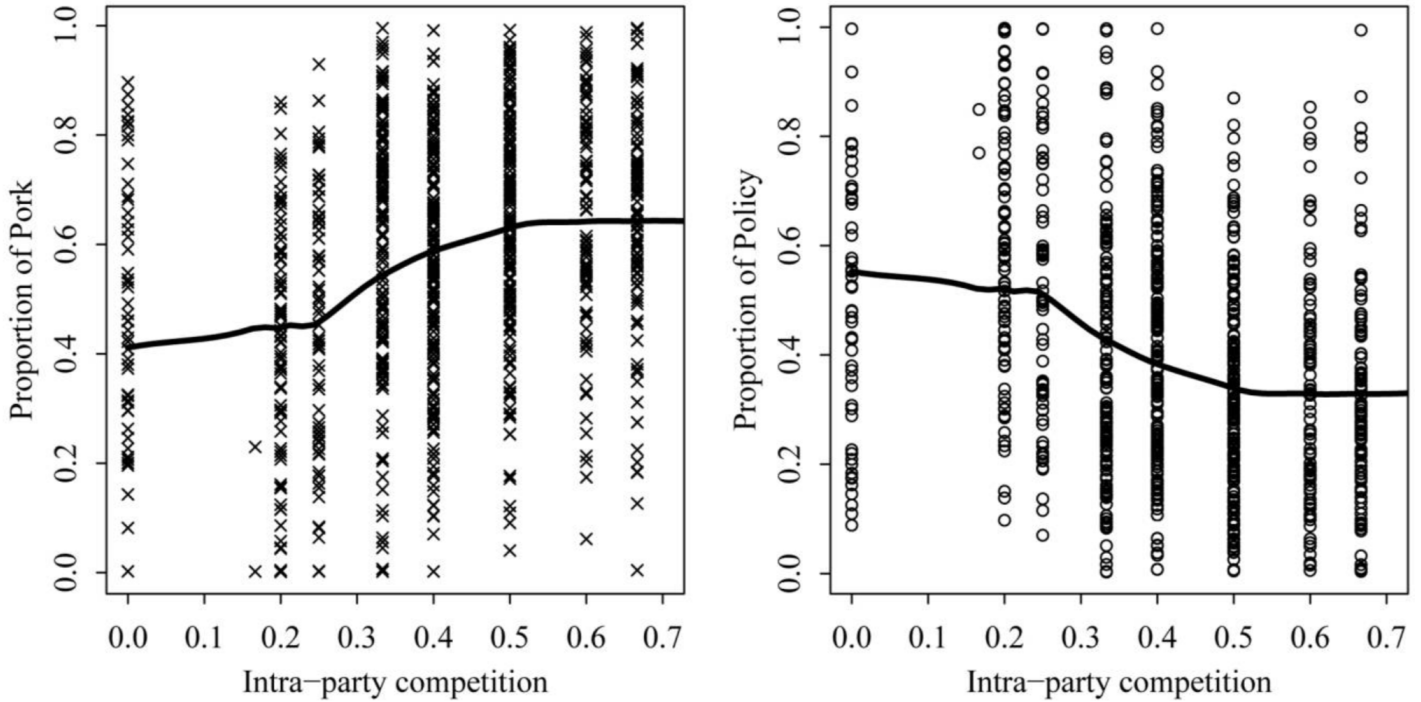


Figure 3. LDP candidates facing higher levels of intraparty competition discussed more pork and less policy than LDP candidates facing lower levels of intraparty competition under SNTV-MMD, where level of intraparty competition is measured by number of LDP opponents relative to district magnitude. The left panel shows that discussion of pork is higher at higher levels of intraparty competition. The right panel shows that discussion of policy is lower at higher levels of intraparty competition.

Assumptions

- Each document consists of a mixture of topics (drawn from LDA distribution)
- Each topic is correlated more strongly with certain terms
- Most topic models rely on bag-of-words: order of words within document is irrelevant

Assumptions

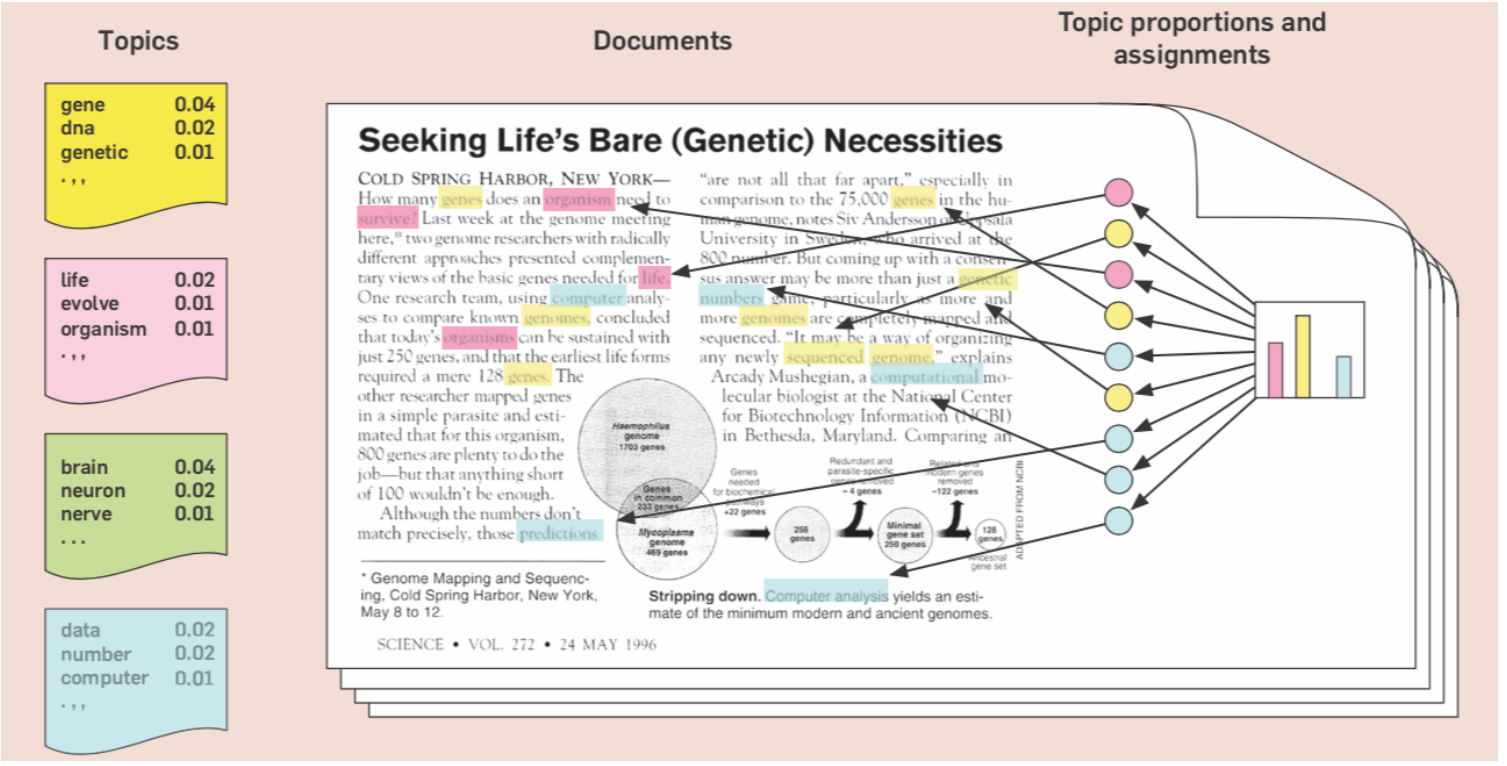
1) Topical Prevalence Matrix (DxK)

$$\boldsymbol{\theta} = \left[\begin{array}{c|cccc} & \textit{Topic}1 & \textit{Topic}2 & \dots & \textit{Topic}K \\ \hline \textit{Doc}1 & .2 & .1 & \dots & 0.05 \\ \textit{Doc}2 & .2 & .1 & \dots & .3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \textit{Doc}D & 0 & 0 & \dots & .5 \end{array}\right]$$

2) Topical Content Matrix (VxK)

$$\boldsymbol{\beta}^{\boldsymbol{T}} = \left[\begin{array}{c|cccc} & \textit{Topic}1 & \textit{Topic}2 & \dots & \textit{Topic}K \\ \hline \text{“}\textit{text}\text{”} & .02 & .001 & \dots & 0.001 \\ \text{“}\textit{data}\text{”} & .001 & .02 & \dots & 0.001 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \text{“}\textit{analysis}\text{”} & .01 & .01 & \dots & 0.0005 \end{array}\right]$$

Example from Blei (2012)



Latent Dirichlet Allocation

The most common form of topic modeling is Latent Dirichlet Allocation or LDA. LDA works as follows:

1. Specifcy *K* (number of topics). 2 Each word in the dfm is assigned randomly to one of the topics (involves a Dirichlet distribution). Numbers assigned across the topics add up to 1.
2. Topic assignments for each word are updated in an iterative fashion by updating the prevalence of the word across the topics, as well as the prevalence of the topics in the document.
3. Assignment stops after user-specified threshold, or when iterations begin to have littel impact on the probabilities assigned to words in the corpus.
4. Output:
    1. Identify words that are most frequently associated with each of the topics.
    2. Probability that each document within the corpus is associated with each of the topics. Probabilities add up to 1.

Source: Tutorial by Chris Bail.

Example: CMU 2008 Political Blog Corpus

- URL: http://www.sailing.cs.cmu.edu/main/?page_id=710

```
# load packages
library(quanteda)
library(readtext)
library(stm)

# url of blog posts
url_blogs <- "https://uclspp.github.io/datasets/data/poliblogs2008.zip"

# download blog posts
dat_blogs <- readtext(url_blogs)

# create a corpus
corp_blogs <- corpus(dat_blogs, text_field = "documents")

# sample 1,500 documents
set.seed(134)
corp_blogs_sample <- corpus_sample(corp_blogs, size = 1500)
```

STM example: create corpus

```
summary(corp_blogs_sample, 6)
```

```
## Corpus consisting of 1500 documents, showing 6 documents:
##
##                     Text Types Tokens Sentences  text           docname
##  poliblogs2008.csv.13132   162    322        11 13132 tpm0834400_0.text
##   poliblogs2008.csv.8362   196    324        10  8362 ha0831900_11.text
##   poliblogs2008.csv.8618   246    449        14  8618  ha0834700_3.text
##     poliblogs2008.csv.83   441    935        30    83  at0801200_4.text
##   poliblogs2008.csv.7206   169    266        11  7206  ha0821900_5.text
##  poliblogs2008.csv.11124   416    721        27 11124  tp0829400_9.text
##        rating day blog
##       Liberal 344  tpm
##  Conservative 319   ha
##  Conservative 347   ha
##  Conservative  12   at
##  Conservative 219   ha
##       Liberal 294   tp
##
## Source: /Users/kbenoit/Dropbox (Personal)/GitHub/quanteda/workshops/modules/* on x86_64 by kbenoit
## Created: Tue Jun 25 20:54:56 2019
## Notes:
```

STM example: create dfm and remove features

```
dfmat_blogs <- dfm(corp_blogs_sample,
                   stem = TRUE,
                   remove = stopwords("english"),
                   remove_punct = TRUE,
                   remove_numbers = TRUE)

dfmat_blogs_trim <- dfm_trim(dfmat_blogs, min_termfreq = 10,
                             min_docfreq = 5)
```

STM example: convert to stm

```
# convert dfm to stm format
stm_blogs <- convert(dfmat_blogs_trim,
                     to = "stm", docvars = docvars(dfmat_blogs))
```

Running the STM

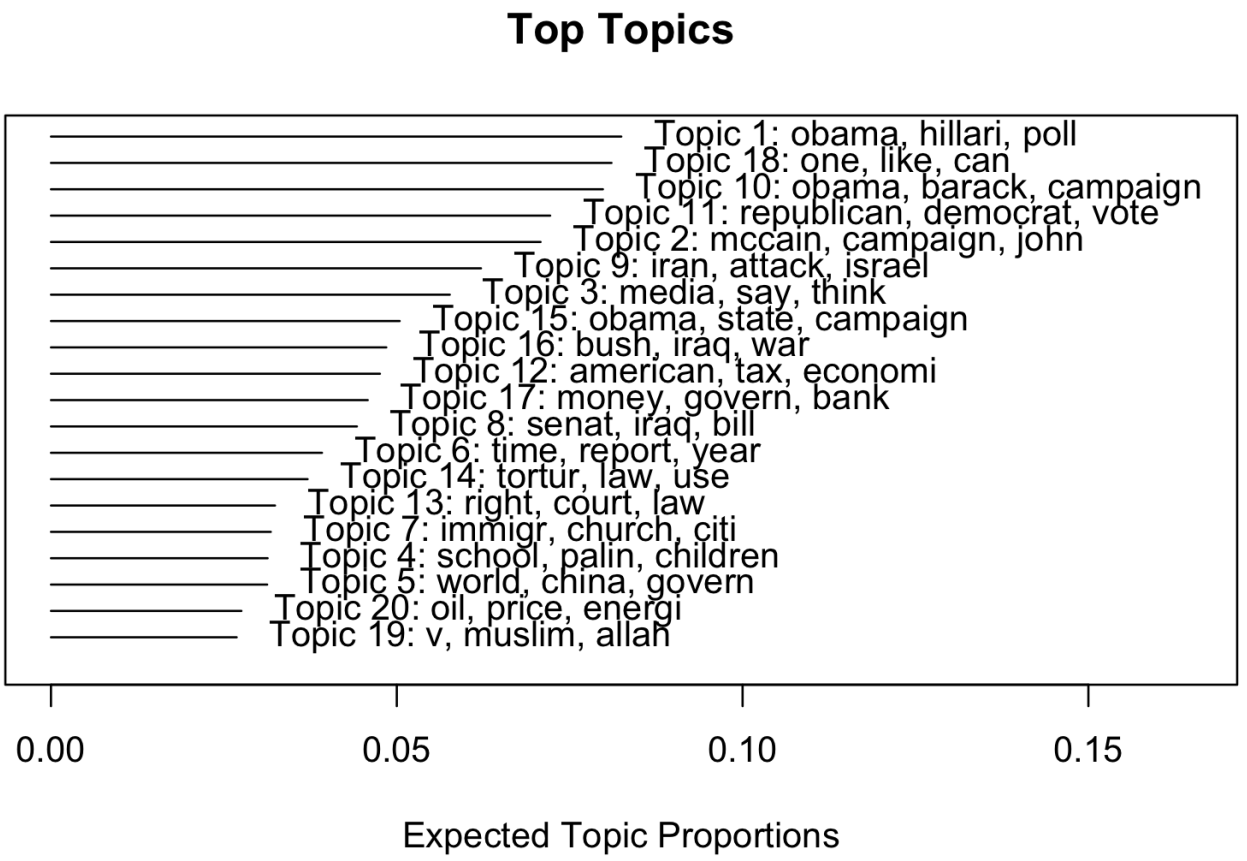
- Use the `stm()` function. `K` specifies the number of topics
- With `K = 0` the **stm** package selects the number of topics automatically (based on algorithm developed by Lee and Mimno (2014))
- Specify covariates with `prevalence`

Running the STM

```
# run structural topic model
stm_object <- stm(documents = stm_blogs$documents,
                  vocab = stm_blogs$vocab,
                  data = stm_blogs$meta,
                  prevalence = ~ rating + s(day),
                  K = 20,
                  seed = 12345)
```

```
## Beginning Spectral Initialization
##      Calculating the gram matrix...
##      Finding anchor words...
##      ..................
##      Recovering initialization...
##      ....................................
## Initialization complete.
## ..................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 1 (approx. per word bound = -7.352)
## ..................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 2 (approx. per word bound = -7.136, relative change = 2.942e-02)
## ..................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 3 (approx. per word bound = -7.098, relative change = 5.204e-03)
```

```
plot(stm_object)
```



Estimate effects

```
# pick and label topics of interest
topics_of_interest <- c(3, 6, 16, 18)
topic_labels <- c("Obama", "Financial Crisis", "Bush", "Iraq")
```

Identify topics of interest

```
labelTopics(stm_object, c(3, 6))
```

```
## Topic 3 Top Words:
##       Highest Prob: media, say, think, peopl, go, know, get
##       FREX: media, matthew, rush, fox, beck, chris, guy
##       Lift: bachmann, limbaugh, beck, hardbal, matthew, luntz, o'reilli
##       Score: bachmann, beck, matthew, limbaugh, clinton, hillari, o'reilli
## Topic 6 Top Words:
##       Highest Prob: time, report, year, new, news, global, warm
##       FREX: warm, climat, ice, global, coverag, newspap, ap
##       Lift: sheet, sulzberg, temperatur, solar, warm, murdoch, pinch
##       Score: sheet, climat, ice, warm, temperatur, sulzberg, solar
```

Identify topics of interest

```
labelTopics(stm_object, c(16, 18))
```

```
## Topic 16 Top Words:
##       Highest Prob: bush, iraq, war, presid, administr, said, year
##       FREX: bush, saddam, war, hussein, iraq, georg, administr
##       Lift: gonzal, wmd, saddam, shoe, hussein, powel, colin
##       Score: bush, iraq, gonzal, saddam, powel, iraqi, rice
## Topic 18 Top Words:
##       Highest Prob: one, like, can, time, just, think, even
##       FREX: film, love, movi, dowd, rememb, allen, exit
##       Lift: pole, filmmak, film, movi, documentari, tuzla, ala
##       Score: pole, film, dowd, movi, allen, mom, hollywood
```

Meanings of Output

- **Highest Probability**: Words with highest probabiliy of occuring in topic
- **FREX**: "Frequency and Exclusive" -- words that distinguish topic from all other topics
- **Lift & Score**: Indicators from **lda**/**maptpx** packages

Find representative documents of topic

```
# note: output too long for slide
findThoughts(stm_object, texts(corp_blogs_sample),
             topics = topics_of_interest, n = 3)
```

Estimate effect of covariates

```
# estimate effect
stm_effects <- estimateEffect(topics_of_interest ~ rating + s(day),
                              stmobj = stm_object,
                              meta = stm_blogs$meta,
                              uncertainty = "Global")
```

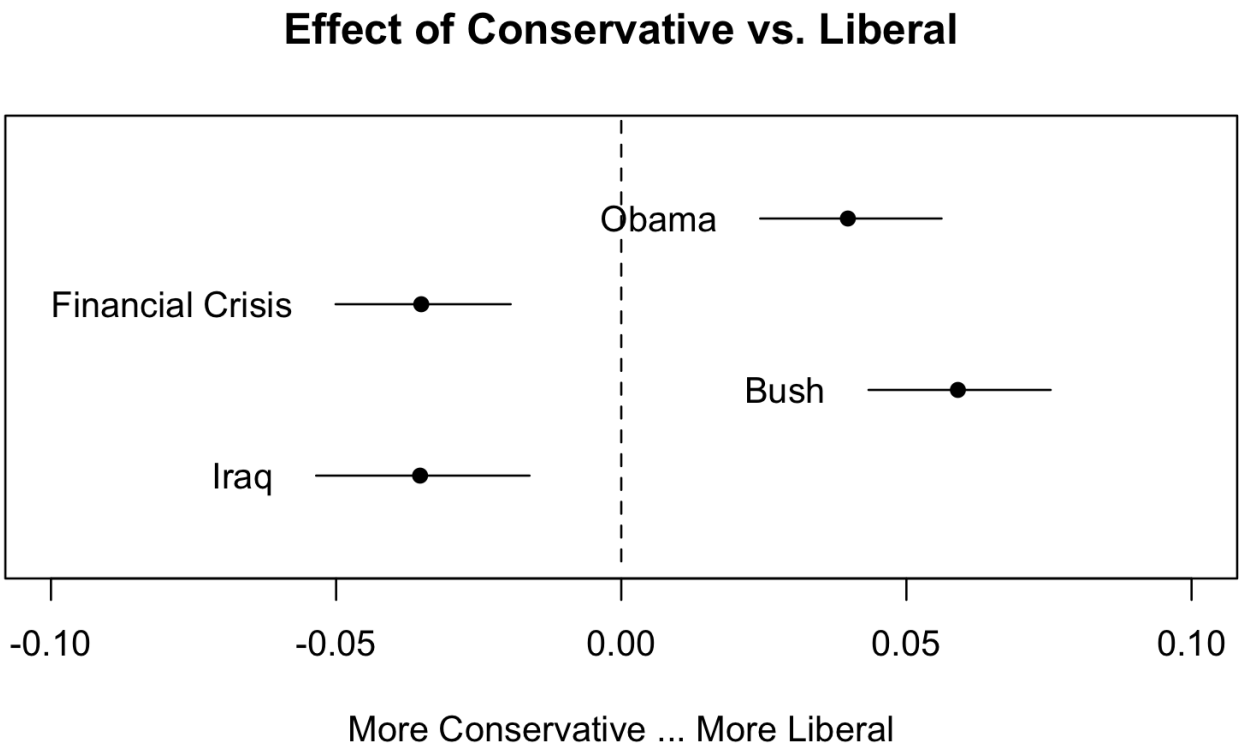
```
summary(stm_effects)
```

Plot influence of discrete variable

```
plot(stm_effects,
     covariate = "rating",
     topics = topics_of_interest,
     model = stm_object,
     method = "difference",
     cov.value1 = "Liberal",
     cov.value2 = "Conservative",
     xlim = c(-0.1, 0.1),
     labeltype = "custom",
     custom.labels = topic_labels, # pay attention to labelling!
     main = "Effect of Conservative vs. Liberal",
     xlab = "More Conservative ... More Liberal")
```

```
plot_effect_libcon <- recordPlot()
plot.new()
```

```
plot_effect_libcon
```

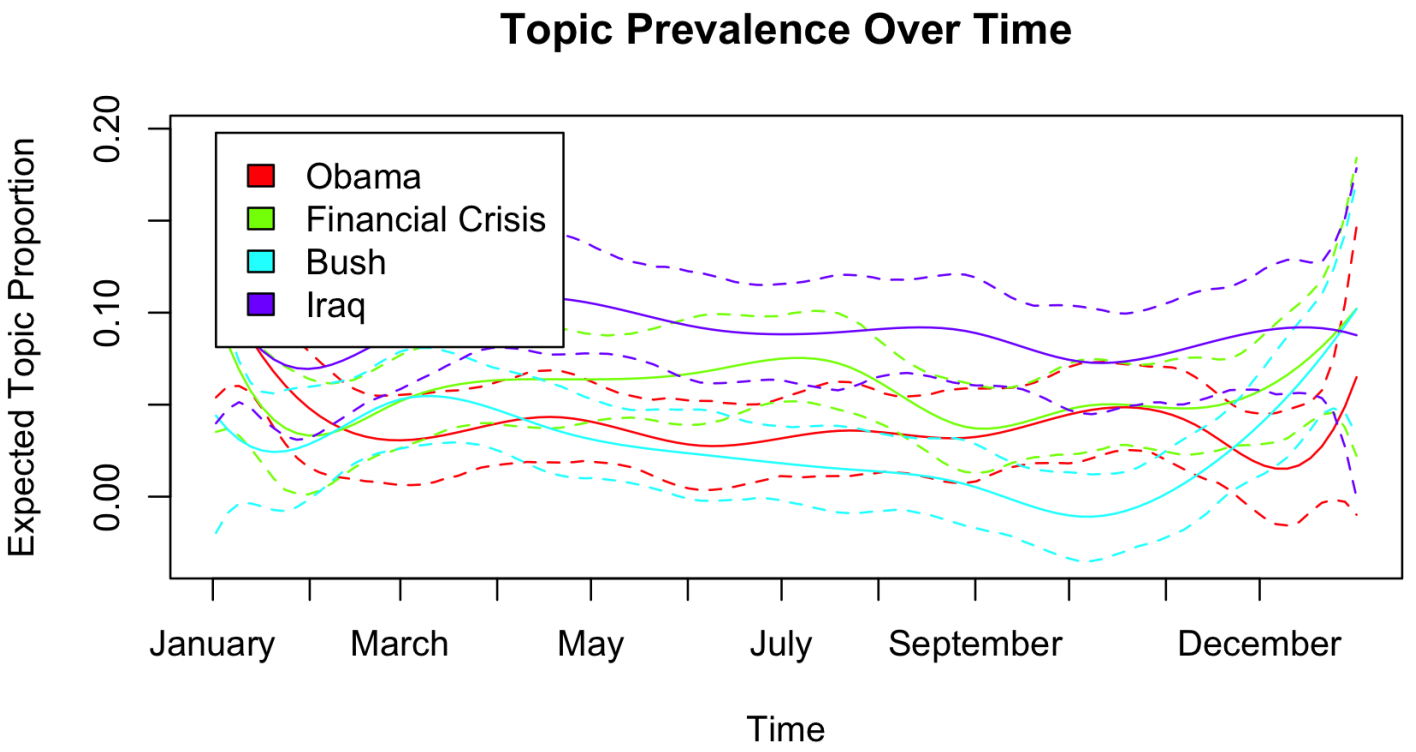


Plot effect of continuous variable

```
plot(stm_effects,
     covariate = "day",
     method = "continuous",
     topics = topics_of_interest,
     model = stm_object,
     labeltype = "custom",
     custom.labels = topic_labels,
     xaxt = "n",
     xlab = "Time",
     main = "Topic Prevalence Over Time"
)
# set the x-axis labels as month names
blog_dates <- seq(from = as.Date("2008-01-01"),
                  to = as.Date("2008-12-01"),
                  by = "month")

axis(1, blog_dates - min(blog_dates), labels = months(blog_dates))
```

```
plot_effect_time <- recordPlot()
plot.new()
```

```
plot_effect_time
```



Create (somewhat) better/tidyier plots

- **tidystm**: https://github.com/mikajoh/tidystm
- **stminsights**: https://github.com/cschwem2er/stminsights

References and useful descriptions

- Margaret Roberts (2017): Structural Topic Models
- Chris Bail (2018): Topic Modelling

Install packages

We will work with the `dplyr` and `ggplot2` packages.

```
install.packages("dplyr")
install.packages("ggplot2")
```

Having installed the package, we need to load them in the R environment.

```
library(dplyr)
library(ggplot2)
```

Gapminder: our sample dataset

We will use the `gapminder` dataset throughout this course.

```
install.packages("gapminder")
```

```
library(gapminder)
```

Getting to know a dataset

Understand the structure and variable coding of a dataset

```
str(gapminder)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1704 obs. of  6 variables:
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ year     : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ lifeExp  : num  28.8 30.3 32 34 36.1 ...
##  $ pop      : int  8425333 9240934 10267083 11537966 13079460 14880372 12881816 13867957 16317921 2
##  $ gdpPercap: num  779 821 853 836 740 ...
```

Print the first six rows

```
head(gapminder)
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779.
## 2 Afghanistan Asia       1957    30.3  9240934      821.
## 3 Afghanistan Asia       1962    32.0 10267083      853.
## 4 Afghanistan Asia       1967    34.0 11537966      836.
## 5 Afghanistan Asia       1972    36.1 13079460      740.
## 6 Afghanistan Asia       1977    38.4 14880372      786.
```

We can find out more information on the data using `?gapminder`.

Data wrangling with dplyr

- Important functions:
  - `select`: pick columns by name
  - `filter`: keep rows matching criteria
  - `arrange`: reorder rows
  - `mutate`: add new variables
  - `group_by`: group rows by category
  - `summarise`: reduce variables to values
- Data visualisation with `ggplot2`

Data wrangling: Sample data frame

```
(df <- data.frame(
  colour = c("green", "black", "black",
             "green", "black", "red"),
  value = 1:6))
```

```
##   colour value
## 1  green     1
## 2  black     2
## 3  black     3
## 4  green     4
## 5  black     5
## 6    red     6
```

Filter/subset a data frame

```
filter(df, colour == "green")
```

```
##   colour value
## 1  green     1
## 2  green     4
```

Applying the pipe (%>%) operator

```
df %>%
  filter(colour != "red") %>%
  filter(value >= 5)
```

```
##   colour value
## 1  black     5
```

```
df %>%
  filter(colour != "red" & value >= 5)
```

```
##   colour value
## 1  black     5
```

Exercise

1. Load `gapminder` dataset (`library(gapminder)`).
2. Type `?gapminder` and `View(gapminder)` to inspect the data.
3. Filter only observations from Norway (call new object `dat_nor`)
4. Filter only observations from Europe or Africa (`gap_europe_africa`)

Solution

```
# load gapminder data
library(gapminder)

# variable names
names(gapminder)
```

```
## [1] "country"   "continent" "year"      "lifeExp"   "pop"       "gdpPercap"
```

```
# number of rows and columns
dim(gapminder)
```

```
## [1] 1704    6
```

Solution

```
dat_nor <- gapminder %>%
  filter(country == "Norway")

nrow(dat_nor)
```

```
## [1] 12
```

```
gap_europe_africa <- gapminder %>%
  filter(continent %in% c("Europe", "Africa"))
```

Data visualisation with **ggplot2**

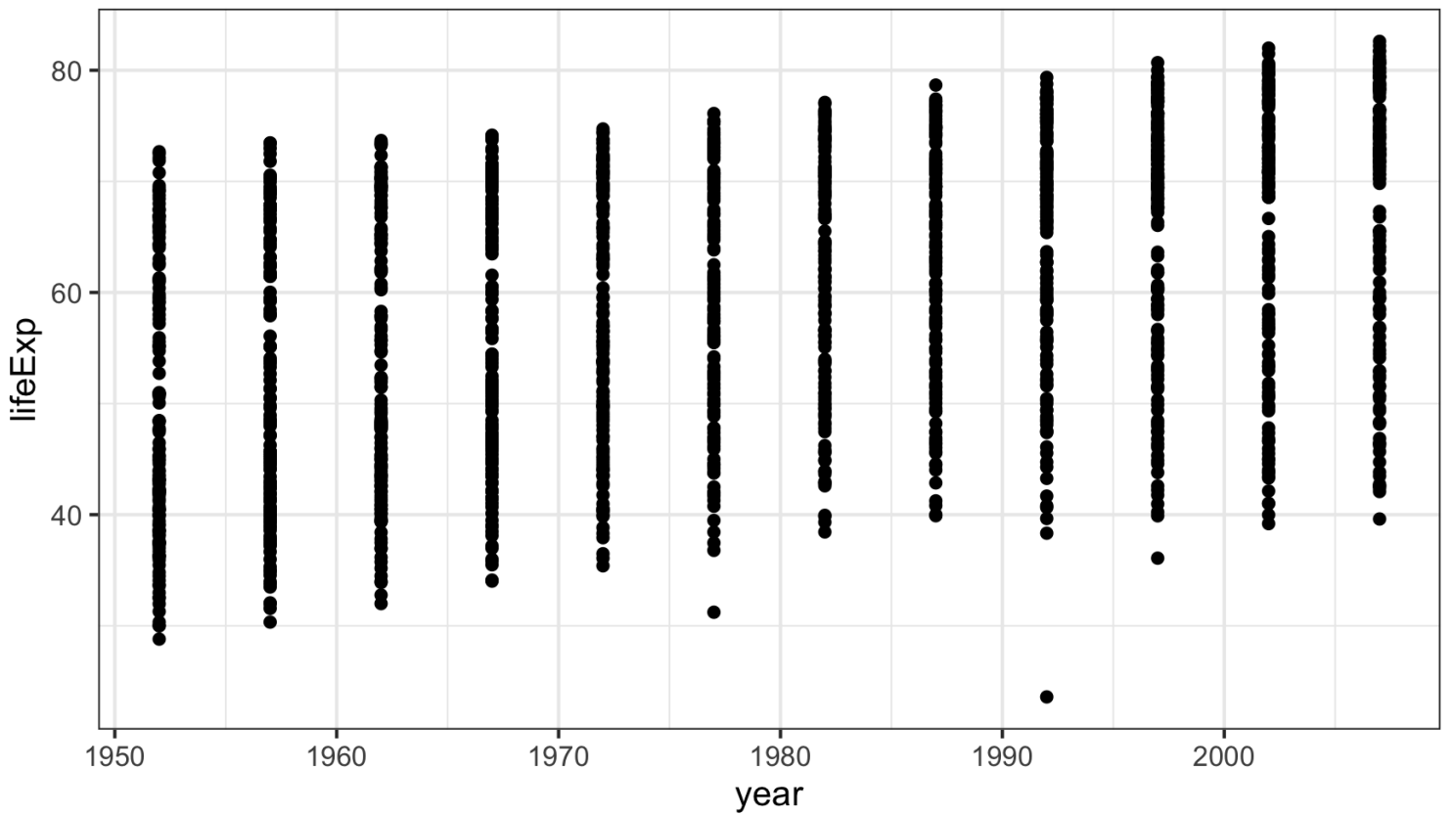
Basic structure:

1. Call `ggplot()`
2. Specify data with `data`
3. Specify coordinate system/axes with `aes()`

```
plot1 <- ggplot(data = gapminder,
                aes(x = year, y = lifeExp))
```

```
plot1
```

```
plot1 +
  geom_point()
```



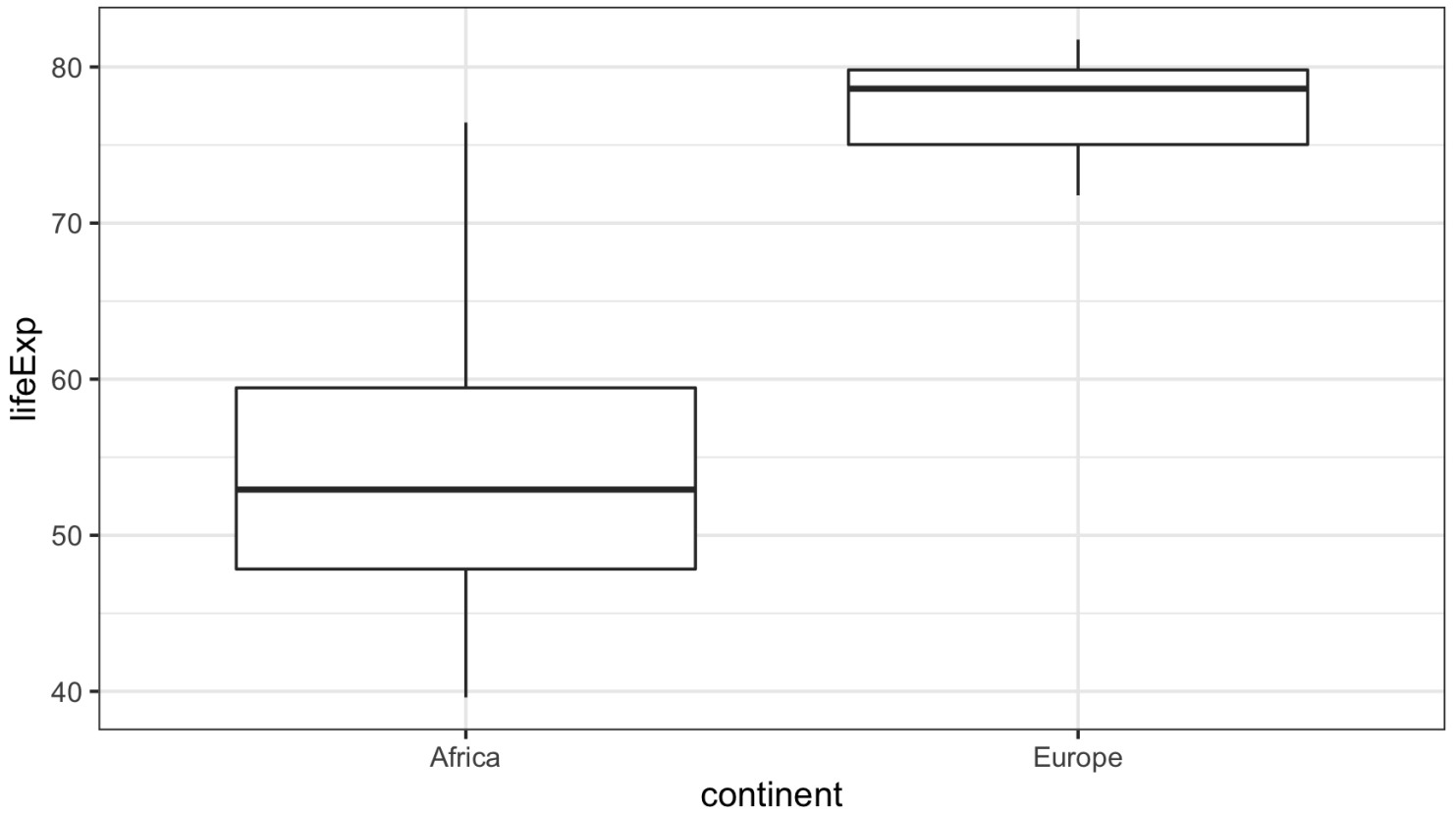
Hands-on Exercise

1. Use `gapminder`, create data frame with only with observations from Europe or Africa in the year 2007
2. Create a boxplot with `continent` on x-axis and `lifeExp` on y-axis

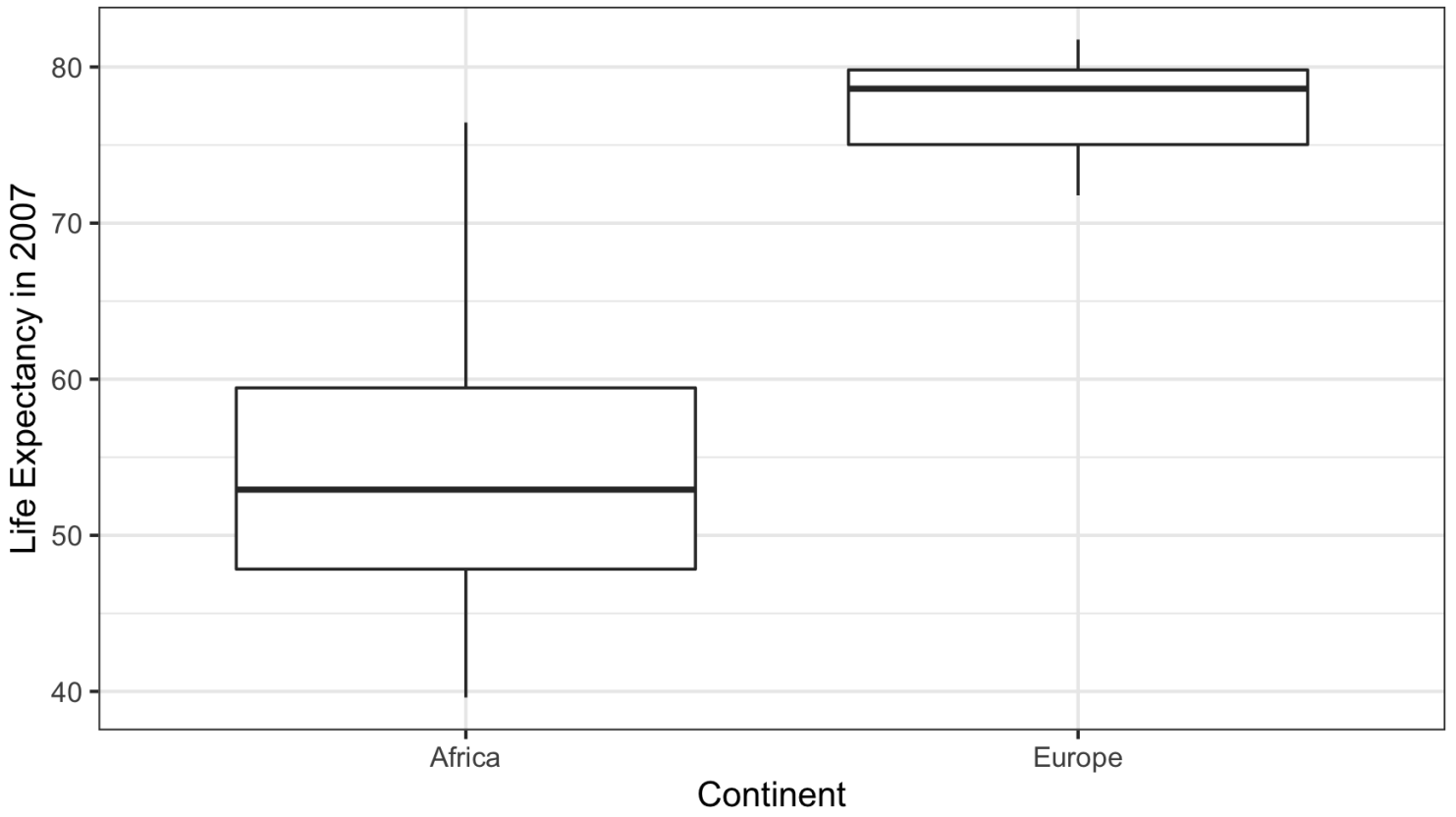
Solution

```
dat_europe_africa_2007 <- gapminder %>%
    filter(continent %in% c("Europe", "Africa") & year == 2007)

plot_boxplot <- ggplot(dat_europe_africa_2007,
       aes(x = continent, y = lifeExp)) +
    geom_boxplot()
```

```
plot_boxplot
```



```
plot_boxplot +
  labs(x = "Continent", y = "Life Expectancy in 2007")
```



General advice

- Save plots as PDF (vector graphic and small):
- Set the **ggplot2** theme globally

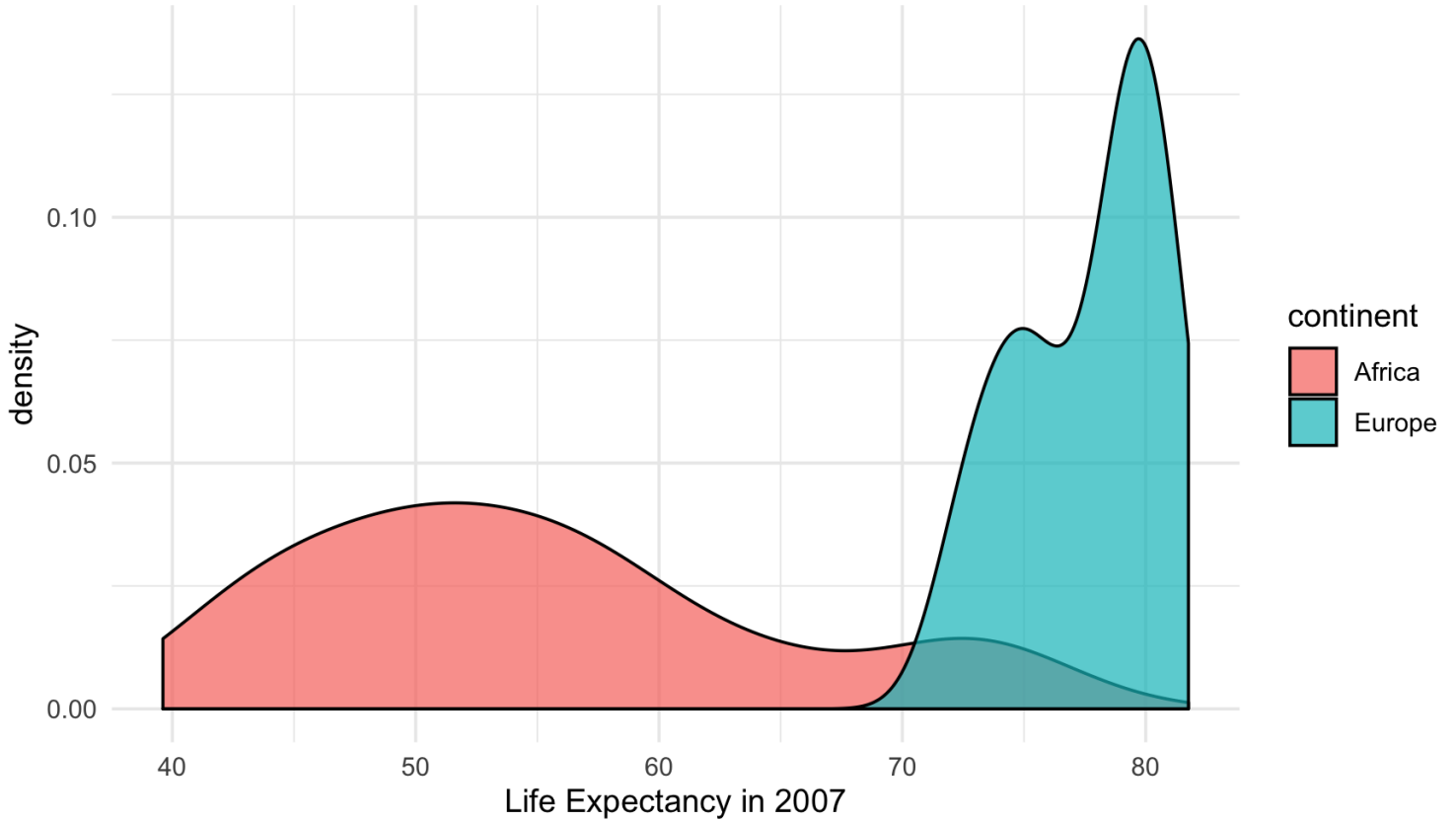
```
# example for saving a plot
ggsave("boxplot.pdf", plot_boxplot, width = 5, height = 7)
```

```
# example for setting a theme
ggplot2::theme_set(theme_minimal())
```

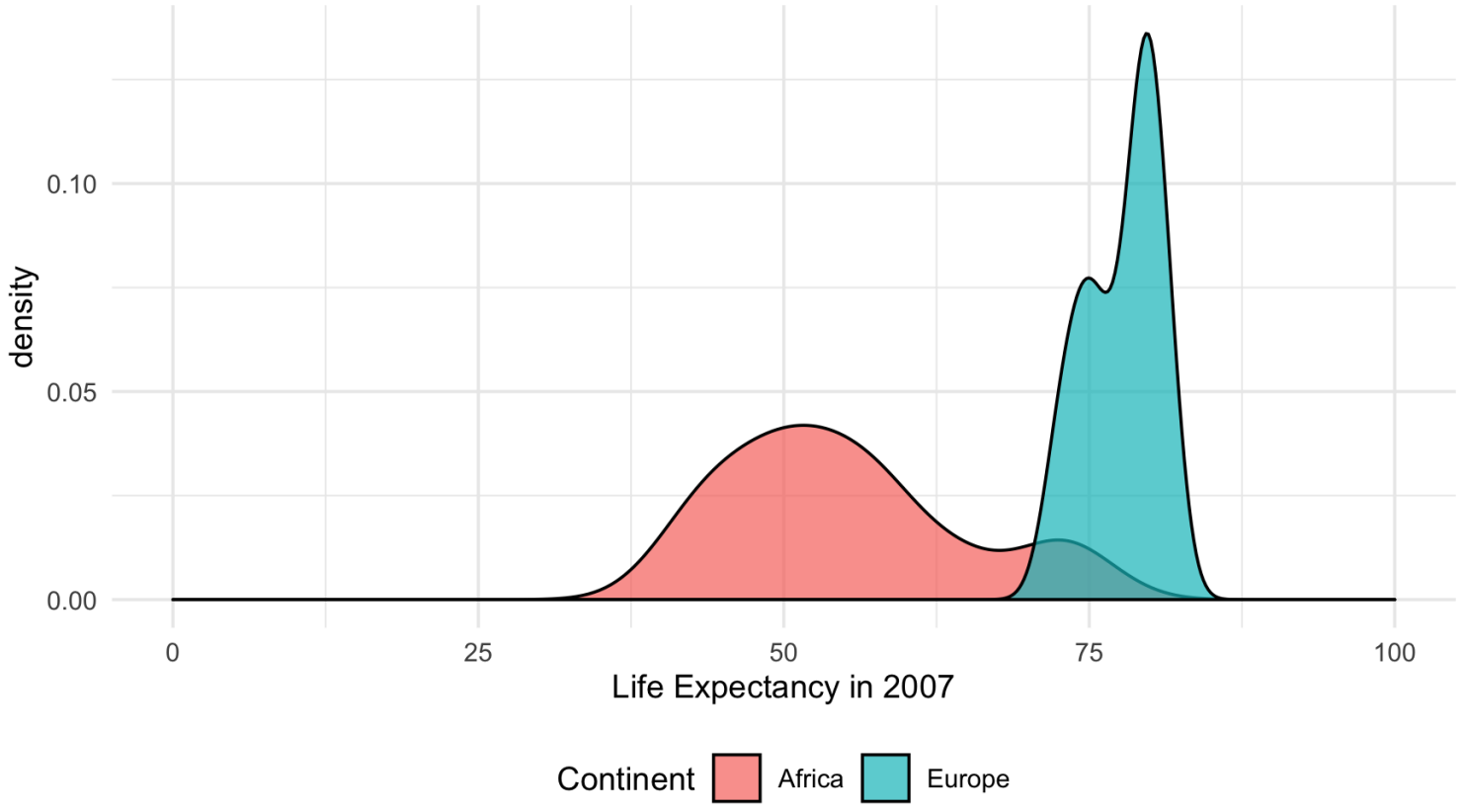
Density plots

```
plot_density <- ggplot(dat_europe_africa_2007,
       aes(x = lifeExp, fill = continent)) +
  geom_density(alpha = 0.7) +
  labs(x = "Life Expectancy in 2007")
```

```
plot_density
```



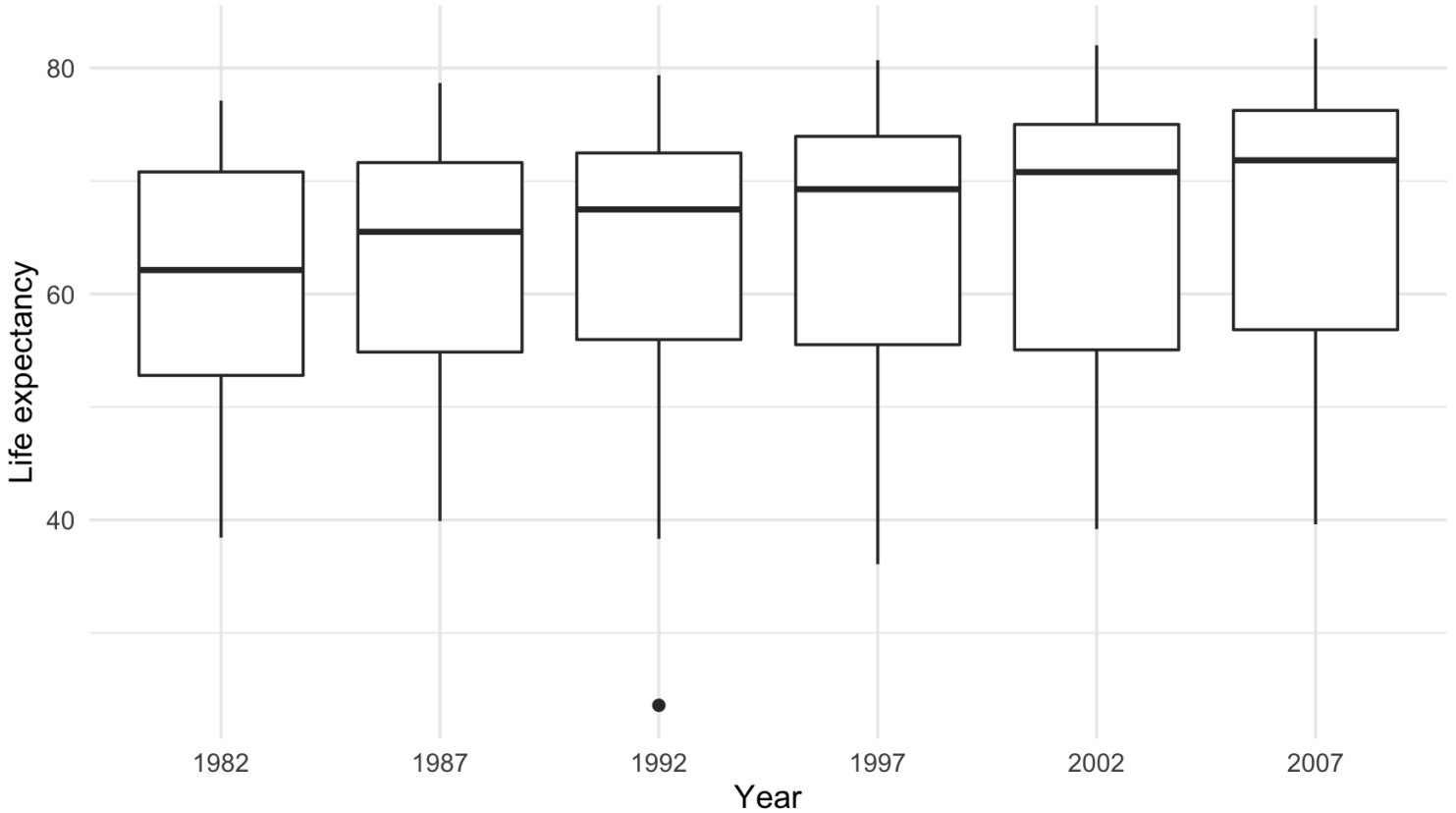
```
plot_density +
  scale_x_continuous(limits = c(0, 100)) +
  scale_fill_discrete(name = "Continent") +
  theme(legend.position = "bottom")
```



Boxplots with groups

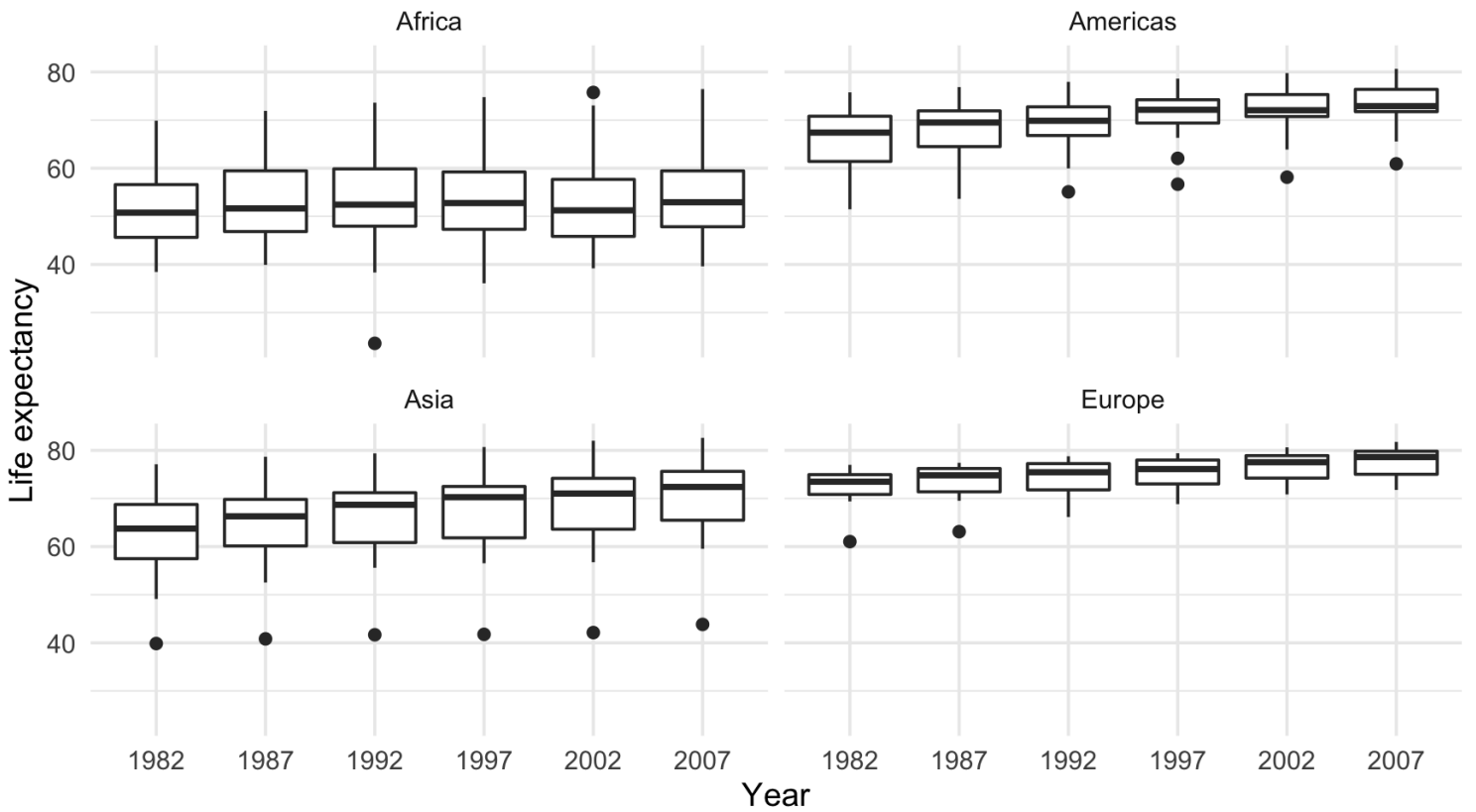
```
gapminder_post_1980 <- gapminder %>%
  filter(year > 1980 & continent != "Oceania")

plot_boxes <- ggplot(data = gapminder_post_1980,
       aes(x = as.factor(year), y = lifeExp)) +
  geom_boxplot() +
  labs(x = "Year", y = "Life expectancy")
```

```
plot_boxes
```



Add facets for a factor variable

```
plot_boxes + facet_wrap(~continent)
```



Arrange/reorder a dataset

```
gap_arranged <- gapminder %>%
  arrange(lifeExp) # ascending

head(gap_arranged)
```

```
## # A tibble: 6 x 6
##   country      continent  year lifeExp     pop gdpPercap
##   <fct>        <fct>     <int>   <dbl>   <int>     <dbl>
## 1 Rwanda       Africa     1992    23.6 7290203      737.
## 2 Afghanistan  Asia       1952    28.8 8425333      779.
## 3 Gambia       Africa     1952    30    284320      485.
## 4 Angola       Africa     1952    30.0 4232095     3521.
## 5 Sierra Leone Africa     1952    30.3 2143249      880.
## 6 Afghanistan  Asia       1957    30.3 9240934      821.
```

Arrange dataset

```
gap_arranged_desc <- gapminder %>%
  arrange(-lifeExp) # descending

head(gap_arranged_desc)
```

```
## # A tibble: 6 x 6
##   country          continent  year lifeExp       pop gdpPercap
##   <fct>            <fct>     <int>   <dbl>     <int>     <dbl>
## 1 Japan            Asia       2007    82.6 127467972    31656.
## 2 Hong Kong, China Asia       2007    82.2   6980412    39725.
## 3 Japan            Asia       2002    82   127065841    28605.
## 4 Iceland          Europe     2007    81.8    301931    36181.
## 5 Switzerland      Europe     2007    81.7   7554661    37506.
## 6 Hong Kong, China Asia       2002    81.5   6762476    30209.
```

Select variables

```
gapminder_select <- gapminder %>%
  select(year, country, lifeExp)

head(gapminder_select, 2)
```

```
## # A tibble: 2 x 3
##    year country     lifeExp
##   <int> <fct>         <dbl>
## 1  1952 Afghanistan    28.8
## 2  1957 Afghanistan    30.3
```

Reorder variables within dataframe

```
gapminder_reordered <- gapminder %>%
  select(pop, year, everything())

head(gapminder_reordered, 2)
```

```
## # A tibble: 2 x 6
##       pop  year country     continent lifeExp gdpPercap
##     <int> <int> <fct>       <fct>       <dbl>     <dbl>
## 1 8425333  1952 Afghanistan Asia         28.8      779.
## 2 9240934  1957 Afghanistan Asia         30.3      821.
```

Exercise

1. Select `country`, `continent`, `year`, `lifeExp` and `pop`
2. Arrange `gapminder` ascending by year *and* within year descending by population.

Solution

```
gapminder_arranged <- gapminder %>%
  select(-gdpPercap) %>%
  arrange(year, -pop)

# check View(gapminder_arranged)
```

Group dataset and summarise within groups

```
df
```

```
##   colour value
## 1  green     1
## 2  black     2
## 3  black     3
## 4  green     4
## 5  black     5
## 6    red     6
```

```
# get sum of value variable
df %>%
  summarise(sum = sum(value))
```

```
##   sum
## 1  21
```

Add grouping variable

```
df %>%
  group_by(colour) %>%
  summarise(sum = sum(value))
```

```
## # A tibble: 3 x 2
##   colour   sum
##   <fct>  <int>
## 1 black     10
## 2 green      5
## 3 red        6
```

Group and summarise data frame

```
df %>%
  group_by(colour) %>%
  summarise(total = n(),
            sum = sum(value))
```

```
## # A tibble: 3 x 3
##   colour total   sum
##   <fct>  <int> <int>
## 1 black      3    10
## 2 green      2     5
## 3 red        1     6
```

Exercise

1. Use `gapminder`.
2. Group by `year`.
3. Calculate the average life expectancy and the maximum GDP per year.

Solution

```
data_gap_summarised <- gapminder %>%
  group_by(year) %>%
  summarise(life_exp_mean = mean(lifeExp),
            gdp_max = max(gdpPercap))
```

```
data_gap_summarised
```

```
## # A tibble: 12 x 3
##     year life_exp_mean gdp_max
##    <int>         <dbl>   <dbl>
##  1  1952          49.1 108382.
##  2  1957          51.5 113523.
##  3  1962          53.6  95458.
##  4  1967          55.7  80895.
##  5  1972          57.6 109348.
##  6  1977          59.6  59265.
##  7  1982          61.5  33693.
##  8  1987          63.2  31541.
##  9  1992          64.2  34933.
## 10  1997          65.0  41283.
## 11  2002          65.7  44684.
## 12  2007          67.0  49357.
```

Create new variables with mutate

```
names(gapminder)
```

```
## [1] "country"   "continent" "year"      "lifeExp"   "pop"       "gdpPercap"
```

```
gapminder <- gapminder %>%
  group_by(year) %>%
  mutate(life_exp_mean = mean(lifeExp),
         diff_country_mean = lifeExp - life_exp_mean)
summary(gapminder$diff_country_mean)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -40.561  -9.583   1.293   0.000  10.240  23.612
```

Further information

Wickham, Hadley and Garrett Grolemund (2017). R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. Sebastopol: O'Reilly.

Healy, Kieran (2019). Data Visualization: A Practical Introduction. Princeton: Princeton University Press.

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.