Very Applied Methods (VAM)
– (Very) Applied Quantitative Text Analysis –

Jan Stuckatz & Tom Paskhalis

LSE
Government & Methodology

31 January 2019

Outline

1. Basic of Quantitative Text Analysis
    - Basic Concepts + Text Descriptives
    - Document Input
    - Regular Expressions
    - *Exercise 1: Load and describe a Corpus of Documents*

Outline

1. Basic of Quantitative Text Analysis
    - Basic Concepts + Text Descriptives
    - Document Input
    - Regular Expressions
    - *Exercise 1: Load and describe a Corpus of Documents*
2. Dictionary Methods
    - Keywords-in-Context
    - Automated Dictionary Methods
    - *Exercise 2: Perform simple Dictionary Analysis*

Outline

1. Basic of Quantitative Text Analysis
    - Basic Concepts + Text Descriptives
    - Document Input
    - Regular Expressions
    - *Exercise 1: Load and describe a Corpus of Documents*
2. Dictionary Methods
    - Keywords-in-Context
    - Automated Dictionary Methods
    - *Exercise 2: Perform simple Dictionary Analysis*
3. Topic Models
    - Latent Dirichlet Allocation
    - LDA Validation
    - Structural Topic Models
    - *Exercise 3: Run and Interpret Topic Models*

Part 1: Basics of Quantitative Text Analysis

Motivation

Workflow: Quantitative Text Analysis

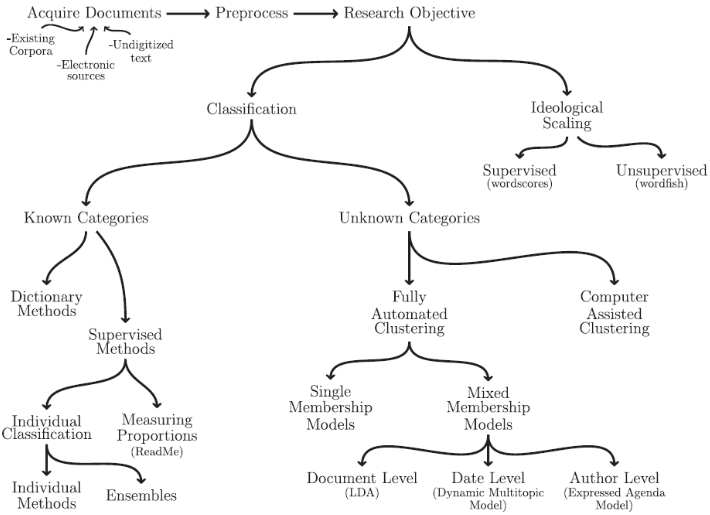


Figure 1 from Grimmer and Stewart (2013)

Workflow: Quantitative Text Analysis

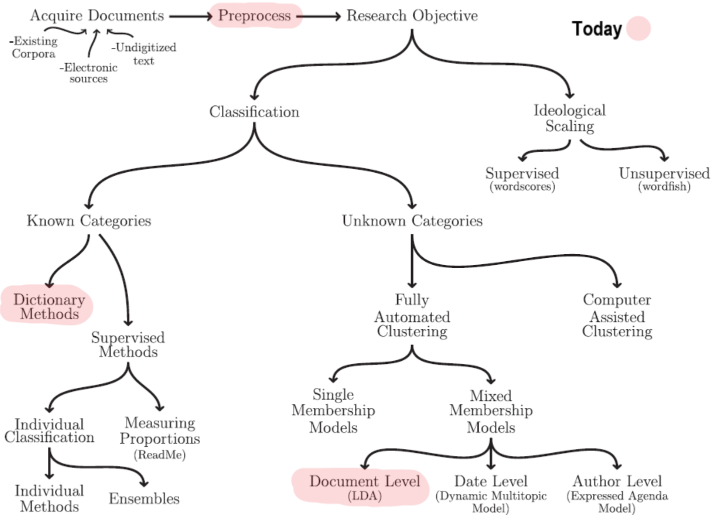


Figure 1 from Grimmer and Stewart (2013)

Some key basic concepts

(text) corpus a large and structured set of texts for analysis

Some key basic concepts

(text) corpus a large and structured set of texts for analysis

document each of the units of the corpus (e.g. a FB post)

Some key basic concepts

(text) corpus: a large and structured set of texts for analysis

document: each of the units of the corpus (e.g. a FB post)

types: for our purposes, a unique word

Some key basic concepts

(text) corpus	a large and structured set of texts for analysis
document	each of the units of the corpus (e.g. a FB post)
types	for our purposes, a unique word
tokens	any word – so token count is total words

Some key basic concepts

(text) corpus	a large and structured set of texts for analysis
document	each of the units of the corpus (e.g. a FB post)
types	for our purposes, a unique word
tokens	any word – so token count is total words
e.g.	

Some key basic concepts

(text) corpus: a large and structured set of texts for analysis

document: each of the units of the corpus (e.g. a FB post)

types: for our purposes, a unique word

tokens: any word – so token count is total words

e.g.

Doc 1: `A corpus is a set of documents.`

Some key basic concepts

(text) corpus a large and structured set of texts for analysis

document each of the units of the corpus (e.g. a FB post)

types for our purposes, a unique word

tokens any word – so token count is total words

e.g.

Doc 1 `A corpus is a set of documents.`

Doc 2 `This is the 2nd document in the corpus.`

is a corpus with 2 documents, where each document is a sentence. The first document has 6 types and 7 tokens. The second has 7 types and 8 tokens. (We ignore punctuation for now.)

Some more key basic concepts

stems words with suffixes removed (using set of rules)

Some more key basic concepts

stems words with suffixes removed (using set of rules)

lemmas canonical word form (the base form of a word that has the same meaning even when different suffixes or prefixes are attached)

Some more key basic concepts

stems words with suffixes removed (using set of rules)

lemmas canonical word form (the base form of a word that has the same meaning even when different suffixes or prefixes are attached)

be careful: lemmas $\neq$ stems! (usage depends on analysis/research design)

word	win	winning	wins	won	winner
stem	win	win	win	won	winner
lemma	win	win	win	win	win

Some more key basic concepts

stems words with suffixes removed (using set of rules)

lemmas canonical word form (the base form of a word that has the same meaning even when different suffixes or prefixes are attached)

be careful: lemmas $\neq$ stems! (usage depends on analysis/research design)

word	win	winning	wins	won	winner
stem	win	win	win	won	winner
lemma	win	win	win	win	win

stop words Words that are designated for exclusion from any analysis of a text (e.g. prepositions, uninformative verbs, pronouns,...)

Parts of speech

- the Penn "Treebank" is the standard scheme for tagging POS

Number	Tag	Description
1.	CC	Coordinating conjunction
2.	CD	Cardinal number
3.	DT	Determiner
4.	EX	Existential *there*
5.	FW	Foreign word
6.	IN	Preposition or subordinating conjunction
7.	JJ	Adjective
8.	JJR	Adjective, comparative
9.	JJS	Adjective, superlative
10.	LS	List item marker
11.	MD	Modal
12.	NN	Noun, singular or mass
13.	NNS	Noun, plural
14.	NNP	Proper noun, singular
15.	NNPS	Proper noun, plural
16.	PDT	Predeterminer
17.	POS	Possessive ending
18.	PRP	Personal pronoun
19.	PRP$	Possessive pronoun
20.	RB	Adverb
21.	RBR	Adverb, comparative
22.	RBS	Adverb, superlative
23.	RP	Particle
24.	SYM	Symbol
25.	TO	*to*
26.	UH	Interjection
27.	VB	Verb, base form
28.	VBD	Verb, past tense
29.	VBG	Verb, gerund or present participle
30.	VBN	Verb, past participle
31.	VBP	Verb, non-3rd person singular present
32.	VBZ	Verb, 3rd person singular present
33.	WDT	Wh-determiner
34.	WP	Wh-pronoun
35.	WP$	Possessive wh-pronoun
36.	WRB	Wh-adverb

Parts of speech

```
## first install spaCy. See instructions on spacyr GitHub page:
## https://github.com/kbenoit/spacyr
install.packages("spacyr")
library(spacyr)

spacy_initialize()
d = spacy_parse("Bob Smith gave Alice his login information.", dependency = TRUE)
d[,-c(1,2)]
```

```
token_id       token       lemma    pos head_token_id  dep_rel   entity
       1         Bob         bob  PROPN             2 compound PERSON_B
       2       Smith       smith  PROPN             3    nsubj PERSON_I
       3        gave        give   VERB             3     ROOT
       4       Alice       alice  PROPN             3   dative PERSON_B
       5         his      -PRON-    ADJ             7     poss
       6       login       login   NOUN             7 compound
       7 information information   NOUN             3     dobj
       8           .           .  PUNCT             3    punct
```

Input Textual Data

- Your best friend: `readtext()` (we will see more in the Example)
- Supports:
    - plain text (`.txt`)
    - JavaScript Object Notation (`.json`) and XML (`.xml`)
    - comma-and tab-separated files (`.csv`, `.tab`, `.tsv`)
    - Microsoft and PDF files (`.doc`, `.docx`, `.pdf`)
- Can easily import multiple files at once

```
# install and load "readtext" package
install.packages("readtext")
library(readtext)

# read all files in senate_speeches folder
speeches <- readtext(C:/directory/texts/senate_speeches/*")

# read  file (here .csv) with multiple columns
speech1 <- readtext("C:/directory/texts/senate_speeches/speech1.csv",
textfield = "Speech") # "Speech" is the text column
```

Preprocessing

- **Preprocessing** might include:

Preprocessing

- **Preprocessing** might include:
  - Tokenization

Preprocessing

- **Preprocessing** might include:
  - Tokenization
  - Lowercasing + Stemming

Preprocessing

- **Preprocessing** might include:
  - Tokenization
  - Lowercasing + Stemming
  - Removing of Stopwords

Preprocessing

- **Preprocessing** might include:
  - Tokenization
  - Lowercasing + Stemming
  - Removing of Stopwords
- **In Quanteda**: can all be done with `quanteda::dfm()`

Preprocessing

- **Preprocessing** might include:
  - Tokenization
  - Lowercasing + Stemming
  - Removing of Stopwords
- **In Quanteda**: can all be done with `quanteda::dfm()`

```
text <-  c(d1 = "An example of preprocessing techniques",
           d2 = "An additional example",
           d3 = "A third example")
dtm <- dfm(text,                              ## input text
           tolower = TRUE, stem = TRUE,       ## set lowercasing and stemming to TRUE
           remove = stopwords("english"))     ## provide the stopwords for deletion
dtm
```

```
Document-feature matrix of: 3 documents, 5 features (53.3\% sparse).
3 x 5 sparse Matrix of class "dfmSparse"
         features
docs        exampl preprocess techniqu   addit  third

  d1             1          1        1       0      0
  d2             1          0        0       1      0
  d3             1          0        0       0      1
```

from Welbers, Van Atteveldt and Benoit (2017, p.253)

Filtering and Weighting

- As mentioned, **not all words are equally informative** for analysis:

Filtering and Weighting

- As mentioned, **not all words are equally informative** for analysis:
    - Word Removal (stopwords)

Filtering and Weighting

- As mentioned, **not all words are equally informative** for analysis:
  - Word Removal (stopwords)
  - Weighting of Words + Document Frequency Selection

Filtering and Weighting

- As mentioned, **not all words are equally informative** for analysis:
  - Word Removal (stopwords)
  - Weighting of Words + Document Frequency Selection
- **In Quanteda**:

Filtering and Weighting

- As mentioned, **not all words are equally informative** for analysis:
  - Word Removal (stopwords)
  - Weighting of Words + Document Frequency Selection
- **In Quanteda**:
  - `tf()`, `docfreq()`, `(tfidf)`

Filtering and Weighting

- As mentioned, **not all words are equally informative** for analysis:
  - Word Removal (stopwords)
  - Weighting of Words + Document Frequency Selection
- **In Quanteda**:
  - `tf()`, `docfreq()`, `(tfidf)`
  - `dfm_weight()`

Filtering and Weighting

- As mentioned, **not all words are equally informative** for analysis:
  - Word Removal (stopwords)
  - Weighting of Words + Document Frequency Selection
- **In Quanteda**:
  - `tf()`, `docfreq()`, `(tfidf)`
  - `dfm_weight()`

```
doc_freq <- docfreq(dtm)          ## document frequency per term (column)
dtm <- dtm[, doc_freq >= 2]       ## select terms with doc_freq >= 2
dtm <- dfm_weight(dtm, "tfidf")   ## weight the features using tf-idf
head(dtm)
```

```
Document-feature matrix of: 5 documents, 524 features (46.6% sparse).
(showing first 5 documents and first 6 features)
                 features
docs              fellow-citizen   senat     hous     repres         :      among
  2uhqjJE?.csv.1      0.2218487 0.39794  0.79588  0.4436975 0.2218487  0.09691001
  2uhqjJE?.csv.2      0.0000000 0.00000  0.00000  0.0000000 0.2218487  0.00000000
  2uhqjJE?.csv.3      0.6655462 0.39794  1.19382  0.6655462 0.0000000  0.38764005
  2uhqjJE?.csv.4      0.4436975 0.00000  0.00000  0.2218487 0.2218487  0.09691001
  2uhqjJE?.csv.5      0.0000000 0.00000  0.00000  0.0000000 0.0000000  0.67837009
```

from Welbers, Van Atteveldt and Benoit (2017, p.254)

Preprocessing + Document-Feature-Matrix

From words to numbers:

1. Preprocess text:

   "A corpus is a set of documents."

   "This is the second document in the corpus."

Preprocessing + Document-Feature-Matrix

From words to numbers:

1. Preprocess text: lowercase,

“a corpus is a set of documents.”

“this is the second document in the corpus.”

Preprocessing + Document-Feature-Matrix

From words to numbers:

1. Preprocess text: lowercase, remove stopwords and punctuation,

“~~a~~ corpus ~~is~~ ~~a~~ set ~~of~~ documents~~.~~”

“~~this~~ ~~is~~ ~~the~~ second document ~~in~~ ~~the~~ corpus~~.~~”

Preprocessing + Document-Feature-Matrix

From words to numbers:

1. Preprocess text: lowercase, remove stopwords and punctuation, stem,

   "corpus set document~~s~~"

   "second document corpus"

Preprocessing + Document-Feature-Matrix

From words to numbers:

1. Preprocess text: lowercase, remove stopwords and punctuation, stem, tokenize into unigrams and bigrams (bag-of-words assumption)

   [corpus, set, document, corpus_set, set_document]

   [second, document, corpus, second_document, document_corpus]

Preprocessing + Document-Feature-Matrix

From words to numbers:

1. Preprocess text: lowercase, remove stopwords and punctuation, stem, tokenize into unigrams and bigrams (bag-of-words assumption)

   [corpus, set, document, corpus_set, set_document]

   [second, document, corpus, second_document, document_corpus]

2. Document-feature matrix:
   - **W**: matrix of N documents by M unique n-grams
   - w_{im}= number of times m-th n-gram appears in i-th document.

	corpus	set	document	corpus set	...	M n-grams
Document 1	1	1	1	1	...	
Document 2	1	0	1	0	...	
...						
Document n	0	1	1	0	...	

Regular Expressions: R Basics

- **RegEx** is a sequence of characters that define a search pattern, to find or replace character patterns. In R **Some useful RegEx** are:

Regular Expressions: R Basics

- **RegEx** is a sequence of characters that define a search pattern, to find or replace character patterns. In R **Some useful RegEx** are:
  - character classes: “`[a-z]`”, “`[A-Z]`”, “`[0-9]`”,

Regular Expressions: R Basics

- **RegEx** is a sequence of characters that define a search pattern, to find or replace character patterns. In R **Some useful RegEx** are:
  - character classes: “`[a-z]`”, “`[A-Z]`”, “`[0-9]`”,
  - special meta-characters:

Regular Expressions: R Basics

- **RegEx** is a sequence of characters that define a search pattern, to find or replace character patterns. In R **Some useful RegEx** are:
  - character classes: “`[a-z]`”, “`[A-Z]`”, “`[0-9]`”,
  - special meta-characters:
    - new line in text: “`\n`”
    - tab in text: “`\t`”

Regular Expressions: R Basics

- **RegEx** is a sequence of characters that define a search pattern, to find or replace character patterns. In R **Some useful RegEx** are:
  - character classes: "`[a-z]`", "`[A-Z]`", "`[0-9]`",
  - special meta-characters:
    - new line in text: "`\n`"
    - tab in text: "`\t`"
  - Anchors:

Regular Expressions: R Basics

- **RegEx** is a sequence of characters that define a search pattern, to find or replace character patterns. In R **Some useful RegEx** are:
    - character classes: "`[a-z]`", "`[A-Z]`", "`[0-9]`",
    - special meta-characters:
        - new line in text: "`\n`"
        - tab in text: "`\t`"
    - Anchors:
        - start of string: "`^`"
        - end of string: "`$`"

Regular Expressions: R Basics

- **RegEx** is a sequence of characters that define a search pattern, to find or replace character patterns. In R **Some useful RegEx** are:
  - character classes: "`[a-z]`", "`[A-Z]`", "`[0-9]`",
  - special meta-characters:
    - new line in text: "`\n`"
    - tab in text: "`\t`"
  - Anchors:
    - start of string: "`^`"
    - end of string: "`$`"
  - Quantifiers:

Regular Expressions: R Basics

- **RegEx** is a sequence of characters that define a search pattern, to find or replace character patterns. In R **Some useful RegEx** are:
  - character classes: “`[a-z]`”, “`[A-Z]`”, “`[0-9]`”,
  - special meta-characters:
    - new line in text: “`\n`”
    - tab in text: “`\t`”
  - Anchors:
    - start of string: “`^`”
    - end of string: “`$`”
  - Quantifiers:
    - match preceding pattern zero or once: “`*`”
    - match preceding pattern once or more: “`+`”
    - match preceding pattern *n* times: “`{n}`”
    - match preceding pattern betwen *n* and *m* times: “`{n,m}`”

Regular Expressions: R Basics

- **RegEx** is a sequence of characters that define a search pattern, to find or replace character patterns. In R **Some useful RegEx** are:
  - character classes: "`[a-z]`", "`[A-Z]`", "`[0-9]`",
  - special meta-characters:
    - new line in text: "`\n`"
    - tab in text: "`\t`"
  - Anchors:
    - start of string: "`^`"
    - end of string: "`$`"
  - Quantifiers:
    - match preceding pattern zero or once: "`*`"
    - match preceding pattern once or more: "`+`"
    - match preceding pattern *n* times: "`{n}`"
    - match preceding pattern betwen *n* and *m* times: "`{n,m}`"
- These can be easily combined with keywords in dictionaries

Regular Expressions: R Basics

- **RegEx** is a sequence of characters that define a search pattern, to find or replace character patterns. In R **Some useful RegEx** are:
  - character classes: "`[a-z]`", "`[A-Z]`", "`[0-9]`",
  - special meta-characters:
    - new line in text: "`\n`"
    - tab in text: "`\t`"
  - Anchors:
    - start of string: "`^`"
    - end of string: "`$`"
  - Quantifiers:
    - match preceding pattern zero or once: "`*`"
    - match preceding pattern once or more: "`+`"
    - match preceding pattern *n* times: "`{n}`"
    - match preceding pattern betwen *n* and *m* times: "`{n,m}`"
- These can be easily combined with keywords in dictionaries
- There are many more (Link), and they can get quite complicated!

Regular Expressions: R Functions

- in **R**, there are a number of useful RegEx commands:

Regular Expressions: R Functions

- in **R**, there are a number of useful RegEx commands:
- All take `pattern` to match on, and `string` (data) to be matched

Regular Expressions: R Functions

- in **R**, there are a number of useful RegEx commands:
- All take `pattern` to match on, and `string` (data) to be matched
  - Detect patterns: `grep()`, `grepl()`

Regular Expressions: R Functions

- in **R**, there are a number of useful RegEx commands:
- All take `pattern` to match on, and `string` (data) to be matched
  - Detect patterns: `grep()`, `grepl()`
  - Locate patterns: `regexpr()`, `gregexpr()`

Regular Expressions: R Functions

- in **R**, there are a number of useful RegEx commands:
- All take `pattern` to match on, and `string` (data) to be matched
  - Detect patterns: `grep()`, `grepl()`
  - Locate patterns: `regexpr()`, `gregexpr()`
  - Replace patterns: `sub()`, `gsub()`

Regular Expressions: R Functions

- in **R**, there are a number of useful RegEx commands:
- All take `pattern` to match on, and `string` (data) to be matched
  - Detect patterns: `grep()`, `grepl()`
  - Locate patterns: `regexpr()`, `gregexpr()`
  - Replace patterns: `sub()`, `gsub()`
  - Extract patterns: `regmatches()` + `gregexpr()`

Regular Expressions: R Functions

- in **R**, there are a number of useful RegEx commands:
- All take `pattern` to match on, and `string` (data) to be matched
  - Detect patterns: `grep()`, `grepl()`
  - Locate patterns: `regexpr()`, `gregexpr()`
  - Replace patterns: `sub()`, `gsub()`
  - Extract patterns: `regmatches()` + `gregexpr()`
  - Split strings by pattern: `strsplit()`

Regular Expressions: R Functions

- in **R**, there are a number of useful RegEx commands:
- All take `pattern` to match on, and `string` (data) to be matched
  - Detect patterns: `grep()`, `grepl()`
  - Locate patterns: `regexpr()`, `gregexpr()`
  - Replace patterns: `sub()`, `gsub()`
  - Extract patterns: `regmatches()` + `gregexpr()`
  - Split strings by pattern: `strsplit()`
  - Substring by position: `substr()`

Regular Expressions: R Functions

- in **R**, there are a number of useful RegEx commands:
- All take `pattern` to match on, and `string` (data) to be matched
  - Detect patterns: `grep()`, `grepl()`
  - Locate patterns: `regexpr()`, `gregexpr()`
  - Replace patterns: `sub()`, `gsub()`
  - Extract patterns: `regmatches()` + `gregexpr()`
  - Split strings by pattern: `strsplit()`
  - Substring by position: `substr()`
- Include options:

Regular Expressions: R Functions

- in **R**, there are a number of useful RegEx commands:
- All take `pattern` to match on, and `string` (data) to be matched
  - Detect patterns: `grep()`, `grepl()`
  - Locate patterns: `regexpr()`, `gregexpr()`
  - Replace patterns: `sub()`, `gsub()`
  - Extract patterns: `regmatches()` + `gregexpr()`
  - Split strings by pattern: `strsplit()`
  - Substring by position: `substr()`
- Include options:
  - `fixed`: match pattern as is or use RegEx

Regular Expressions: R Functions

- in **R**, there are a number of useful RegEx commands:
- All take `pattern` to match on, and `string` (data) to be matched
    - Detect patterns: `grep()`, `grepl()`
    - Locate patterns: `regexpr()`, `gregexpr()`
    - Replace patterns: `sub()`, `gsub()`
    - Extract patterns: `regmatches()` + `gregexpr()`
    - Split strings by pattern: `strsplit()`
    - Substring by position: `substr()`
- Include options:
    - `fixed`: match pattern as is or use RegEx
    - `ignore.case`: ignore lower/upper case

Exercise 1: Load Data and Simple Text Manipulation

- **Data: UK Withdrawal Agreement from the European Union**

Hint 1 keep regex101.com and one RegEx cheatsheet open while doing this exercise.

Hint 2 you will need `gsub()` and `stringr::str_extract_all()`. Check out their help files.

Regular Expression Resources

- Useful websites to test regular expressions:
  - regexr.com
  - regex101.com
- Regular Expression Cheatsheets
  - good cheatsheet: Link
  - alternative: Link
- Introductions to RegEx in R
  - General String Manipulation Intro by Gaston Sanchez
  - General RegEx Intro: Link
  - RegEx in R using `stringr` package: Link
  - RegEx using base R functions: Link

Part 2: Dictionary Methods

Dictionary Methods

- Between quantitative and qualitative text analysis

Dictionary Methods

- Between quantitative and qualitative text analysis
- “Qualitative”: identification of concepts of interest and associated keys/categories

Dictionary Methods

- Between quantitative and qualitative text analysis
- “Qualitative”: identification of concepts of interest and associated keys/categories
- Dictionary construction involves a lot of contextual interpretation and qualitative judgment

Dictionary Methods

- Between quantitative and qualitative text analysis
- “Qualitative”: identification of concepts of interest and associated keys/categories
- Dictionary construction involves a lot of contextual interpretation and qualitative judgment
- “Quantitative’: very high reliability/reproducibility of analysis/procedure

Rationale for dictionaries

- Rather than count words that occur, pre-define words associated with specific meanings

Rationale for dictionaries

- Rather than count words that occur, pre-define words associated with specific meanings
- Two components:

  key the label for the equivalence class for the concept or canonical term

  values (multiple) terms or patterns that are declared equivalent occurences of the key class
- Frequently involves lemmatization: better than stemming

Rationale for dictionaries

- Rather than count words that occur, pre-define words associated with specific meanings
- Two components:

  key the label for the equivalence class for the concept or canonical term

  values (multiple) terms or patterns that are declared equivalent occurences of the key class
- Frequently involves lemmatization: better than stemming

Example: Welbers et al. 2017

```
myDict <- dictionary(list(terror = c("terror*"),
                          economy = c("job*", "business*", "econom*")))
dict_dtm <- dfm_lookup(dtm, myDict, nomatch = "_unmatched")
tail(dict_dtm)
```

```
Document-feature matrix of: 58 documents, 3 features (37.4% sparse).
(showing last 6 documents and last 3 features)
                     features
docs                   terror    economy      _unmatched
   1997-Clinton             2          3            1125
   2001-Bush                0          2             782
   2005-Bush                0          1            1040
   2009-Obama               1          7            1165
   2013-Obama               0          6            1030
   2017-Trump               1          5             709
```

from Welbers, Van Atteveldt and Benoit (2017, p.255)

Validate Search Terms: Keywords-in-Context (KWIC)

Note the differences between `glob`, `regex`, and `fixed`

```
head(kwic(data_corpus_inaugural, "secure*", window = 3, valuetype = "glob"))
#>
#>       [1797-Adams, 479]  welfare, and | secure   | the blessings of
#>      [1797-Adams, 1513]  nations, and | secured  | immortal glory with
#>  [1805-Jefferson, 2368]    , and shall | secure   | to you the
#>     [1817-Monroe, 1755] cherished. To | secure   | us against these
#>     [1817-Monroe, 1815] defense as to | secure   | our cities and
#>     [1817-Monroe, 3012]      I can to | secure   | economy and fidelity
head(kwic(data_corpus_inaugural, "secur", window = 3, valuetype = "regex"))
#>
#>  [1789-Washington, 1497] government for the | security | of their union
#>        [1797-Adams, 479]       welfare, and |  secure  | the blessings of
#>       [1797-Adams, 1513]       nations, and | secured  | immortal glory with
#>   [1805-Jefferson, 2368]        , and shall |  secure  | to you the
#>      [1813-Madison, 321]       seas and the | security | of an important
#>      [1817-Monroe, 1610]      may form some | security | against these dangers
head(kwic(data_corpus_inaugural, "security", window = 3, valuetype = "fixed"))
#>
#>  [1789-Washington, 1497] government for the | security |
#>      [1813-Madison, 321]       seas and the | security |
#>      [1817-Monroe, 1610]      may form some | security |
#>      [1817-Monroe, 3430]           and as a | security |
#>       [1825-Adams, 1371]      that the best | security |
#>       [1825-Adams, 1443]    that the firmest | security |
```

Building a Dictionary: What to Consider

- The ideal content analysis dictionary *associates all and only the relevant words* to each category in a perfectly valid scheme
- Three key issues:

Validity	Is the dictionary's category scheme valid?
Recall	Does this dictionary identify *all* my content?
Precision	Does it identify *only* my content?

- There exist more automated/data-driven ways to build dictionaries/keywords (King, Lam and Roberts, 2017).

How to build a dictionary

1. **Identify "extreme texts" with "known" positions. Examples**:
   - Speeches by populist vs mainstream politicians (for populism dictionary)
   - Facebook comments to news about natural catastrophes vs football victories (for sentiment dictionary)
   - Subreddits for white nationalist groups vs regular politics (for racist rhetoric)
2. Search for differentially occurring words using word frequencies
3. Examine these words in context to check their precision and recall
4. Use regular expressions to see whether stemming or wildcarding is required

Example: Populism Dictionary

APPENDIX B
DICTIONARY OF THE COMPUTER-BASED CONTENT ANALYSIS

	NL	UK	GE	IT
Core	elit*	elit*	elit*	elit*
	consensus*	consensus*	konsens*	consens*
	ondemocratisch*	undemocratic*	undemokratisch*	antidemocratic*
	ondemokratisch*			
	referend*	referend*	referend*	referend*
	corrupt*	corrupt*	korrupt*	corrot*
	propagand*	propagand*	propagand*	propagand*
	politici*	politici*	politiker*	politici*
	bedrog	*deceit*	täusch*	ingann*
	bedrieg	*deceiv*	betrüg*	
			betrug*	
	verraa	*betray*	*verrat*	tradi*
	verrad			
	schaam*	shame*	scham*	vergogn*
			schäm*	
	schand*	scandal*	skandal*	scandal*
	waarheid*	truth*	wahrheit*	verità
	oneerlijk*	dishonest*	unfair*	disonest*
			unehrlich*	
Context	establishm*	establishm*	establishm*	partitocrazia
	heersend*	ruling*	*herrsch*	
	capitul*			
	kapitul*			
	kaste*			
	leugen*		lüge*	menzogn*
	lieg*			mentir*

from Rooduijn and Pauwels (2011)

Hierarchical Dictionaries

- Dictionaries can include **hierarchies of keywords** for further systematization
- Example: Manifesto Project
- In R, these are implemented using the familiar `list()` function and `quanteda:dictionary()`

```
dict <- quanteda::dictionary(
list(trade = list(general=c("trade*", "tariff*", "import*", "export*"),
china=c("china", "dumping", "steel", "aluminum", "cheat"),
institutions= c("trade agreement*", "wto", "nafta") )
```

Dictionaries: Pro's and Con's

+ very flexible and easy to construct
+ use of expert knowledge
+ great for corpus exploration
- highly specific to context
- some words with multiple meanings
- some limits to use in other multiple languages

Exercise 2

- Data: US Senate Speeches
- Application: Dictionary Approaches

Hint 1 For dictionaries, be aware that `glob` (wildcards like $*$) is not the same as `regex`.

Hint 2 For handling/reshaping corpora and dfm's, make extensive use of the `dplyr` package.

Hint 3 You are handling BIG data, so calculations can take some time. Subset the data with the `subset()` or `dplyr:filter()` if you want quickly check whether your code works.

Part 3: Topic Models

Topic Models

- Topic Models are algorithms to discover the 'main themes' in unstructured text corpora

Topic Models

- Topic Models are algorithms to discover the ‘main themes’ in unstructured text corpora
- Require no prior information, dictionary, or input by researcher

Topic Models

- Topic Models are algorithms to discover the 'main themes' in unstructured text corpora
- Require no prior information, dictionary, or input by researcher
- only input $= K$, the number of topics to be discovered

Topic Models

- Topic Models are algorithms to discover the 'main themes' in unstructured text corpora
- Require no prior information, dictionary, or input by researcher
- only input $= K$, the number of topics to be discovered
- *Mixed-Membership Model*: documents belong to multiple topics, and topic distributions vary over documents

Latent Dirichlet Allocation

- The LDA model is a Bayesian mixture model for discrete data

Latent Dirichlet Allocation

- The LDA model is a Bayesian mixture model for discrete data
- LDA provides a generative model that describes how the documents in a dataset were created

Latent Dirichlet Allocation

- The LDA model is a Bayesian mixture model for discrete data
- LDA provides a generative model that describes how the documents in a dataset were created
- Each of the K *topics* is a distribution over a fixed vocabulary

Latent Dirichlet Allocation

- The LDA model is a Bayesian mixture model for discrete data
- LDA provides a generative model that describes how the documents in a dataset were created
- Each of the K *topics* is a distribution over a fixed vocabulary
- Each document is a collection of words, generated according to a multinomial distribution, one for each of K topics

Latent Dirichlet Allocation

- The LDA model is a Bayesian mixture model for discrete data
- LDA provides a generative model that describes how the documents in a dataset were created
- Each of the K *topics* is a distribution over a fixed vocabulary
- Each document is a collection of words, generated according to a multinomial distribution, one for each of K topics
- Inference consists of estimating a posterior distribution from a joint distribution based on the probability model from a combination of what is observed (words in documents) and what is hidden (topic and word parameters)

Latent Dirichlet Allocation

Key parameters:

1. θ = matrix of dimensions N documents by K topics where θ_{ik} corresponds to the probability that document i belongs to topic k; i.e. assuming $K = 5$:

	T1	T2	T3	T4	T5
Document 1	0.15	0.15	0.05	0.10	0.55
Document 2	0.80	0.02	0.02	0.10	0.06
...					
Document N	0.01	0.01	0.96	0.01	0.01

Latent Dirichlet Allocation

Key parameters:

1. θ = matrix of dimensions N documents by K topics where θ_{ik} corresponds to the probability that document i belongs to topic k; i.e. assuming $K = 5$:

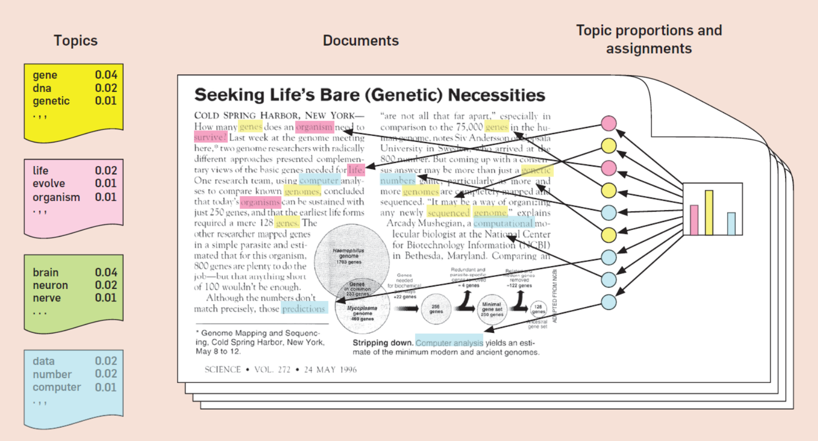
	T1	T2	T3	T4	T5
Document 1	0.15	0.15	0.05	0.10	0.55
Document 2	0.80	0.02	0.02	0.10	0.06
...					
Document N	0.01	0.01	0.96	0.01	0.01

2. β = matrix of dimensions K topics by M words where β_{km} corresponds to the probability that word m belongs to topic k; i.e. assuming $M = 6$:

	W1	W2	W3	W4	W5	W6
Topic 1	0.40	0.05	0.05	0.10	0.10	0.30
Topic 2	0.10	0.10	0.10	0.50	0.10	0.10
...						
Topic k	0.05	0.60	0.10	0.05	0.10	0.10

Example 1: Topics in *Science* Articles

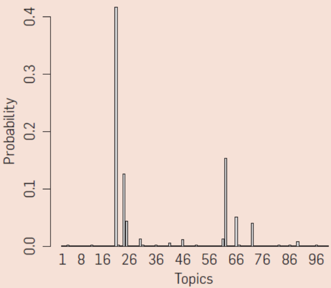
Figure 1. The intuitions behind latent Dirichlet allocation. We assume that some number of "topics," which are distributions over words, exist for the whole collection (far left). Each document is assumed to be generated as follows. First choose a distribution over the topics (the histogram at right); 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.



from Blei (2012), Figure 1

Example 1: Topics in *Science* Articles

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

from Blei (2012), Figure 2

Validation

Validate the Topics (Quinn et al., 2010)

- **Semantic validity**: do the topics identify coherent document groups that are meaningfully related?

Validation

Validate the Topics (Quinn et al., 2010)

- **Semantic validity**: do the topics identify coherent document groups that are meaningfully related?
- **Convergent/discriminant construct validity** do the topics match from existing measures? Do they differ where they should?

Validation

Validate the Topics (Quinn et al., 2010)

- **Semantic validity**: do the topics identify coherent document groups that are meaningfully related?
- **Convergent/discriminant construct validity** do the topics match from existing measures? Do they differ where they should?
- **Predictive validity** is the topic variation in line with real-world events?

Validation

Validate the Topics (Quinn et al., 2010)

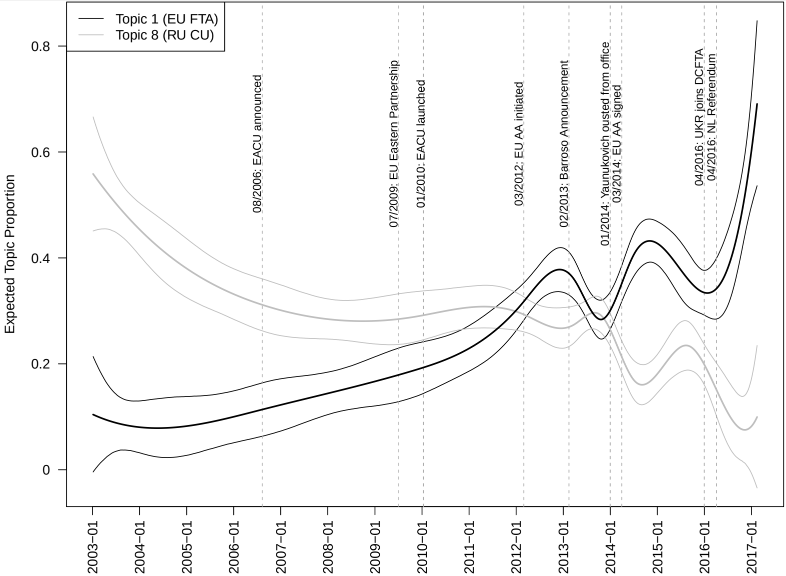
- **Semantic validity**: do the topics identify coherent document groups that are meaningfully related?
- **Convergent/discriminant construct validity** do the topics match from existing measures? Do they differ where they should?
- **Predictive validity** is the topic variation in line with real-world events?
- **Hypothesis validity** can the topic variation be used to test hypotheses?

Example 2: Trade Policy Topics in Business News

- approx. 2200 English business news on Ukrainian trade relations
- goal: extract topics about 'EU-Ukrainian Free Trade Agreement' and 'Russian-Ukrainian Customs Union'
- Fit (structural) Topic Model using $K = 10$
- Some topic examples:

Topic	**Words**
EU-UKR Free Trade	europeanparlia, easternpartnership, poroshenko, euukrain, summit, petroporoshenko, ratifi, associationagr
RUS-UKR Customs U.	freetrad, zone, customsunion, commoneconom, wto, kuchma, join, tradeorgan, ces
RUS Energy	tymoshenko, gazprom, gas, russianga, gastransit, bcm, billioncub, pipelin, naturalga, cubic
UKR IMF	respond, default, western, imf, sberbank, gdp, money, loan, crisi, currenc

Example 2: Trade Policy Topics in Business News



Choose K, the Number of Topics

- Most difficult question with regards to topic models

Choose K, the Number of Topics

- Most difficult question with regards to topic models
- Measures of Model Fit are available:

Choose K, the Number of Topics

- Most difficult question with regards to topic models
- Measures of Model Fit are available:
  - can compute a likelihood for "held-out" data

Choose K, the Number of Topics

- Most difficult question with regards to topic models
- Measures of Model Fit are available:
  - can compute a likelihood for “held-out” data
  - can compute perplexity:

$$\text{perplexity}(w) = \exp\left\{ -\frac{\sum_{d=1}^{M} \log p(w_d)}{\sum_{d=1}^{M} N_d} \right\}$$

Choose K, the Number of Topics

- Most difficult question with regards to topic models
- Measures of Model Fit are available:
  - can compute a likelihood for “held-out” data
  - can compute perplexity:

$$\text{perplexity}(w) = \exp\left\{ -\frac{\sum_{d=1}^{M} \log p(w_d)}{\sum_{d=1}^{M} N_d} \right\}$$

  - lower perplexity score indicates better performance

Choose K, the Number of Topics

- Most difficult question with regards to topic models
- Measures of Model Fit are available:
  - can compute a likelihood for “held-out” data
  - can compute perplexity:

$$\text{perplexity}(w) = \exp\left\{ -\frac{\sum_{d=1}^{M} \log p(w_d)}{\sum_{d=1}^{M} N_d} \right\}$$

  - lower perplexity score indicates better performance
  - implemented in `topicmodels::perplexity()` to use with LDA object

Choose K, the Number of Topics

- Most difficult question with regards to topic models
- Measures of Model Fit are available:
  - can compute a likelihood for “held-out” data
  - can compute perplexity:

$$\text{perplexity}(w) = \exp\left\{-\frac{\sum_{d=1}^{M} \log p(w_d)}{\sum_{d=1}^{M} N_d}\right\}$$

  - lower perplexity score indicates better performance
  - implemented in `topicmodels::perplexity()` to use with LDA object
- Problem: often there is a negative relationship between the substantive information of topics and the best-fitting model according to above measures

Choose K, the Number of Topics

- Most difficult question with regards to topic models
- Measures of Model Fit are available:
  - can compute a likelihood for “held-out” data
  - can compute perplexity:

$$\text{perplexity}(w) = \exp\left\{-\frac{\sum_{d=1}^{M} \log p(w_d)}{\sum_{d=1}^{M} N_d}\right\}$$

  - lower perplexity score indicates better performance
  - implemented in `topicmodels::perplexity()` to use with LDA object
- Problem: often there is a negative relationship between the substantive information of topics and the best-fitting model according to above measures
- Grimmer & Stewart propose to ‘choose K based on substantive fit’

Extensions of LDA

Varieties of Topic Models

- Structural topic model (Roberts et al, 2014, AJPS)

Extensions of LDA

Varieties of Topic Models

- Structural topic model (Roberts et al, 2014, AJPS)
- Dynamic topic model (Blei and Lafferty, 2006, ICML; Quinn et al, 2010, AJPS)

Extensions of LDA

Varieties of Topic Models

- Structural topic model (Roberts et al, 2014, AJPS)
- Dynamic topic model (Blei and Lafferty, 2006, ICML; Quinn et al, 2010, AJPS)
- Hierarchical topic model (Griffiths and Tenembaun, 2004, NIPS; Grimmer, 2010, PA)

Extensions of LDA

Varieties of Topic Models

- Structural topic model (Roberts et al, 2014, AJPS)
- Dynamic topic model (Blei and Lafferty, 2006, ICML; Quinn et al, 2010, AJPS)
- Hierarchical topic model (Griffiths and Tenembaun, 2004, NIPS; Grimmer, 2010, PA)

Why?

Extensions of LDA

Varieties of Topic Models

- Structural topic model (Roberts et al, 2014, AJPS)
- Dynamic topic model (Blei and Lafferty, 2006, ICML; Quinn et al, 2010, AJPS)
- Hierarchical topic model (Griffiths and Tenembaun, 2004, NIPS; Grimmer, 2010, PA)

Why?

- Substantive reasons: incorporate specific elements of DGP into estimation

Extensions of LDA

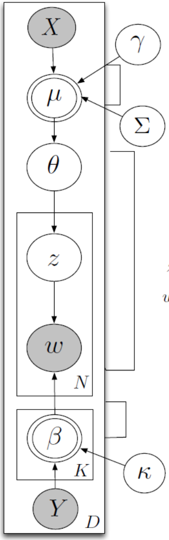
Varieties of Topic Models

- Structural topic model (Roberts et al, 2014, AJPS)
- Dynamic topic model (Blei and Lafferty, 2006, ICML; Quinn et al, 2010, AJPS)
- Hierarchical topic model (Griffiths and Tenembaun, 2004, NIPS; Grimmer, 2010, PA)

Why?

- Substantive reasons: incorporate specific elements of DGP into estimation
- Statistical reasons: structure (by document-covariates) can lead to better topics.

Structural topic model



Topic Prevalence:

$$\begin{aligned}\mu_{d,k} &= X_d \gamma_k \\ \gamma_k &\sim \mathcal{N}(0, \sigma_k^2) \\ \sigma_k^2 &\sim \text{Gamma}(s^\gamma, r^\gamma)\end{aligned}$$

Language Model:

$$\begin{aligned}\theta_d &\sim \text{LogisticNormal}(\mu_d, \Sigma) \\ z_{d,n} &\sim \text{Mult}(\theta_d) \\ w_{d,n} &\sim \text{Mult}(\beta_d^{k=z_{d,n}})\end{aligned}$$

Topical Content:

$$\begin{aligned}\beta_{d,v}^k &\propto \exp(m_v + \kappa_v^{\cdot,k} + \kappa_v^{y,\cdot} + \kappa_v^{y,k}) \\ \kappa_v^{y,k} &\sim \text{Laplace}(0, \tau_v^{y,k}) \\ \tau_v^{y,k} &\sim \text{Gamma}(s^\kappa, r^\kappa)\end{aligned}$$

- **Prevalence**: Prior on the mixture over topics is now document-specific, and can be a function of covariates (documents with similar covariates will tend to be about the same topics)
- **Content**: distribution over words is now document-specific and can be a function of covariates (documents with similar covariates will tend to use similar words to refer to the same topic)

Exercise 3

- Data: US Senate Speeches
- Application: LDA, STM

Hint 1 Make sure to plot the topics along meaningful dimensions of the data. This is a great way to connect topics to your theories/research design.

Hint 2 For `stm` models: make sure to choose between `topical prevalence` and `topical content` depending on what is most appropriate for your research question.

Further Reading

- **Basics of Quantitative Text Analysis**
  - Welbers, Van Atteveldt and Benoit (2017)
  - Grimmer and Stewart (2013)
  - Krippendorff (2004)
  - Denny and Spirling (2018)
- **Dictionary Methods**
  - Laver and Garry (2000)
  - Rooduijn and Pauwels (2011)
  - Lowe et al. (2011)
  - Seale, Ziebland and Charteris-Black (2006)
- **Topic Models**
  - LDA: Blei (2012), more technical: Blei, Ng and Jordan (2003)
  - STM: Roberts et al. (2014); Roberts, Stewart and Tingley (2015) and overview of research using STM

Thank You!

Jan: j.stuckatz@lse.ac.uk

Tom: t.g.paskhalis@lse.ac.uk

References I

Blei, David M. 2012. "Probabilistic topic models." *Communications of the ACM* 55(4):77–84.

Blei, David M., Andrew Y. Ng and Michael I. Jordan. 2003. "Latent Dirichlet Allocation." *Journal of Machine Learning Research* 3(4-5):993–1022.

Denny, Matthew J and Arthur Spirling. 2018. "Preprocessing." *Working Paper* pp. 1–49.

Grimmer, Justin and Brandon M. Stewart. 2013. "Text as data: The promise and pitfalls of automatic content analysis methods for political texts." *Political Analysis* 21(3):267–297.

King, Gary, Patrick Lam and Margaret E. Roberts. 2017. "Computer-Assisted Keyword and Document Set Discovery from Unstructured Text." *American Journal of Political Science* 61(4):971–988.

Krippendorff, Klaus. 2004. *Content Analysis. An introduction to Its Methodology*. 2 ed. Thousand Oaks: Sage Publications.

Laver, Michael and John Garry. 2000. "Estimating Policy Positions from Political Texts." *American Journal of Political Science* 44(3):619.

Lowe, Will, Kenneth Benoit, Mikhaylov Slava and Michael Laver. 2011. "Scaling policy preferences from coded political texts." *Legislative Studies Quarterly* 36(1):123–155.

References II

Quinn, Kevin M, Burt L Monroe, Michael Colaresi, Michael H Crespin and Dragomir R Radev. 2010. "How to Analyze Political Attention with Minimal Assumptions and Costs." *American Journal of Political Science* 54(1):209–228.

Roberts, Margaret E., Brandon M. Stewart, Dustin H. Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Kushner Gadarian, Bethany Albertson and David G. Rand. 2014. "Structural topic models for open-ended survey responses." *American Journal of Political Science* 58(4):1064–1082.

Roberts, Margaret E., Brandon M. Stewart and Dustin Tingley. 2015. "stm: R Package for Structural Topic Models." *Journal of Statistical Software* VV(II).

Rooduijn, Matthijs and Teun Pauwels. 2011. "Measuring Populism: Comparing Two Methods of Content Analysis." *West European Politics* 34(6):1272–1283.

Seale, Clive, Sue Ziebland and Jonathan Charteris-Black. 2006. "Gender, cancer experience and internet use: A comparative keyword analysis of interviews and online cancer support groups." *Social Science and Medicine* 62(10):2577–2590.

Welbers, Kasper, Wouter Van Atteveldt and Kenneth Benoit. 2017. "Text Analysis in R." *Communication Methods and Measures* 11(4):245–265.