Webscraping, Processing, and Text Analysis Workshop

Denver McNeney

Centre for the Study of Democratic Citizenship
McGill University

denver.mcneney@mail.mcgill.ca

March 14, 2016

Download Materials

- To download slides, scripts, and example materials, visit: `https://github.com/denvermc/Text-Analyses-Workshops/`
- To download $\mathbb{R}$, visit `https://www.r-project.org/`

Webscraping: What is it?

- Automatic extraction and parsing of online information to create structured database
- Two kinds:
  - Web APIs (Application Program Interface) → Website or database creates interface for data requests that return JSON or XML files
  - Screen or Pseudo-Manual Scraping → Need to either extract from html or interact with website using bots to download materials

Webscraping: What is it?

- Automatic extraction and parsing of online information to create structured database
- Two kinds:
  - Web APIs (Application Program Interface) $\rightarrow$ Website or database creates interface for data requests that return JSON or XML files
  - Screen or Pseudo-Manual Scraping $\rightarrow$ Need to either extract from html or interact with website using bots to download materials

Webscraping: What is it?

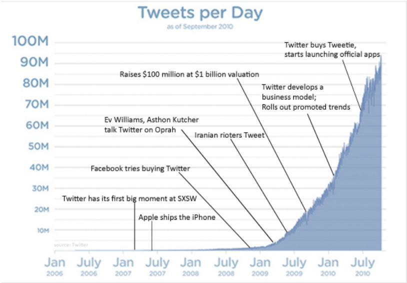
- Automatic extraction and parsing of online information to create structured database
- Two kinds:
  - Web APIs (Application Program Interface) $\rightarrow$ Website or database creates interface for data requests that return JSON or XML files
  - Screen or Pseudo-Manual Scraping $\rightarrow$ Need to either extract from html or interact with website using bots to download materials

Webscraping: What is it?

- Automatic extraction and parsing of online information to create structured database
- Two kinds:
  - Web APIs (Application Program Interface) $\rightarrow$ Website or database creates interface for data requests that return JSON or XML files
  - Screen or Pseudo-Manual Scraping $\rightarrow$ Need to either extract from html or interact with website using bots to download materials

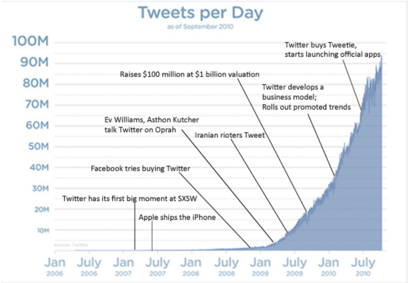
Webscraping: Why Bother?

- Amazing amount of data available online
- **But!** Data is usually unstructured and often available across a number of databases or websites
- Downloading information manually is time-consuming (perhaps impossible), boring & error-ridden



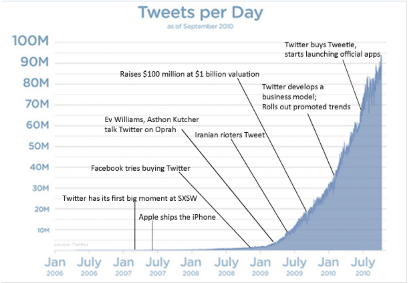
Webscraping: Why Bother?

- Amazing amount of data available online
- **But!** Data is usually unstructured and often available across a number of databases or websites
- Downloading information manually is time-consuming (perhaps impossible), boring & error-ridden

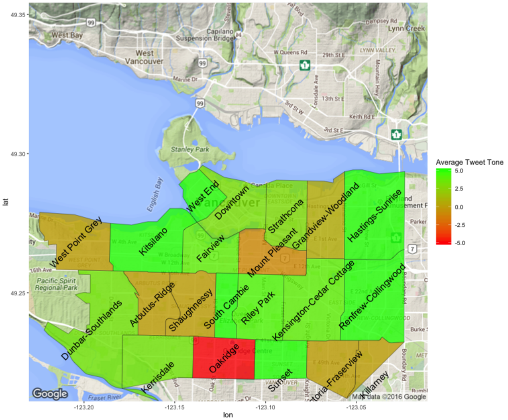


Webscraping: Why Bother?

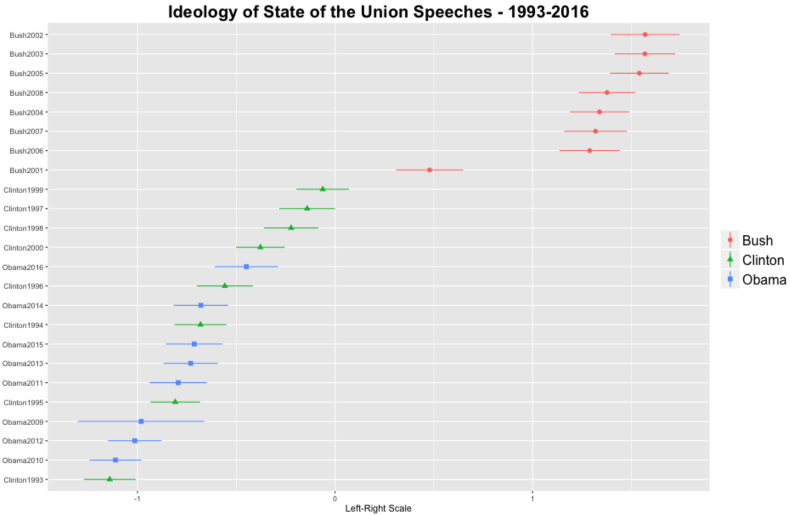
- Amazing amount of data available online
- **But!** Data is usually unstructured and often available across a number of databases or websites
- Downloading information manually is time-consuming (perhaps impossible), boring & error-ridden



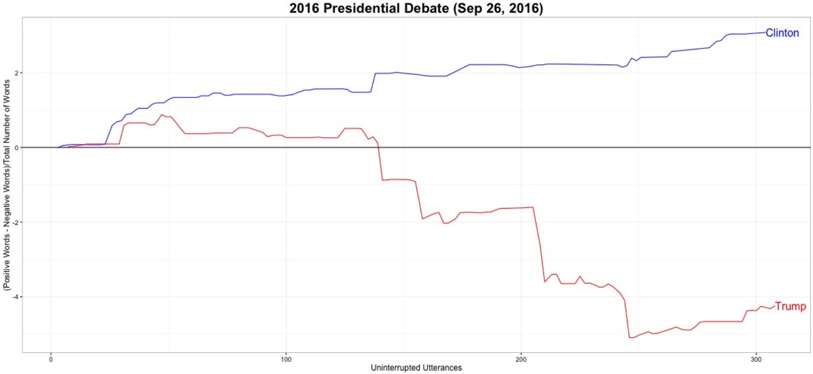
Webscraping: Why Bother?



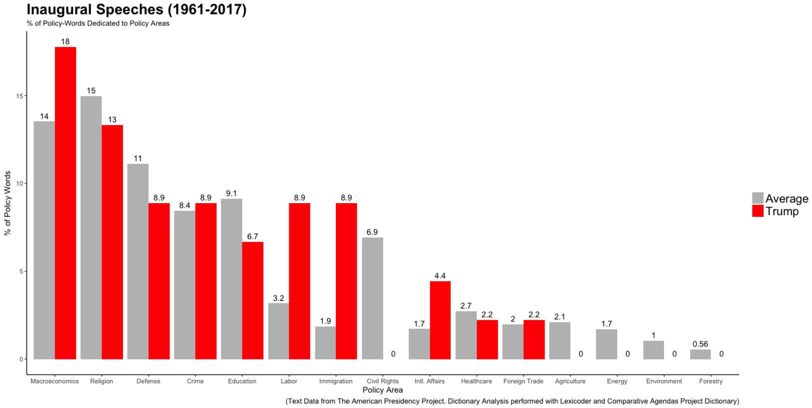
Webscraping: Why Bother?



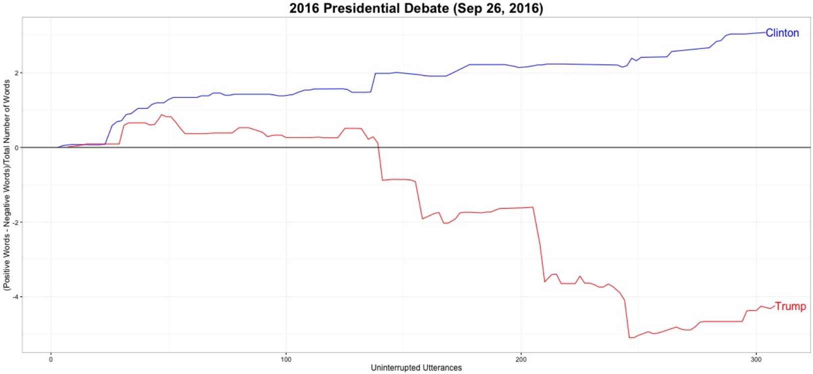
Webscraping: Why Bother?



Webscraping: Why Bother?



Webscraping: Why Bother?



Scraping Twitter with $\mathbb{R}$

Twitter provides resources (APIs) for downloading Tweets:

- RESTful API:
  - Information about users and their *existing* tweets (static)
  - i.e. All tweets by Donald Trump, list of followers and friends, etc.
- Streaming API:
  - "Stream" of Tweets as they become available
  - i.e. Keyword-specific tweets, Geo-tagged tweets
  - Issues:
    - Unless using "Firehose" method ($$$) can only collect random 1% of tweets
    - Can only go back 2 weeks

Scraping Twitter with ℝ

Twitter provides resources (APIs) for downloading Tweets:

- RESTful API:
  - Information about users and their *existing* tweets (static)
  - i.e. All tweets by Donald Trump, list of followers and friends, etc.
- Streaming API:
  - "Stream" of Tweets as they become available
  - i.e. Keyword-specific tweets, Geo-tagged tweets
  - Issues:
    - Unless using "Firehose" method ($$) can only collect random 1% of tweets
    - Can only go back 2 weeks

Scraping Twitter with $\mathbb{R}$

Twitter provides resources (APIs) for downloading Tweets:

- RESTful API:
  - Information about users and their *existing* tweets (static)
  - i.e. All tweets by Donald Trump, list of followers and friends, etc.
- Streaming API:
  - "Stream" of Tweets as they become available
  - i.e. Keyword-specific tweets, Geo-tagged tweets
  - Issues:
    - Unless using "Firehose" method ($$$) can only collect random 1% of tweets
    - Can only go back 2 weeks

Scraping Twitter with $\mathbb{R}$

Twitter provides resources (APIs) for downloading Tweets:

- RESTful API:
  - Information about users and their *existing* tweets (static)
  - i.e. All tweets by Donald Trump, list of followers and friends, etc.
- Streaming API:
  - "Stream" of Tweets as they become available
  - i.e. Keyword-specific tweets, Geo-tagged tweets
  - Issues:
    - Unless using "Firehose" method ($$$) can only collect random 1% of tweets
    - Can only go back 2 weeks

Scraping Twitter with $\mathbb{R}$

Twitter provides resources (APIs) for downloading Tweets:

- RESTful API:
  - Information about users and their *existing* tweets (static)
  - i.e. All tweets by Donald Trump, list of followers and friends, etc.
- Streaming API:
  - "Stream" of Tweets as they become available
  - i.e. Keyword-specific tweets, Geo-tagged tweets
  - Issues:
    - Unless using "Firehose" method ($$$) can only collect random 1% of tweets
    - Can only go back 2 weeks

Scraping Twitter with $\mathbb{R}$

Twitter provides resources (APIs) for downloading Tweets:

- RESTful API:
  - Information about users and their *existing* tweets (static)
  - i.e. All tweets by Donald Trump, list of followers and friends, etc.
- Streaming API:
  - "Stream" of Tweets as they become available
  - i.e. Keyword-specific tweets, Geo-tagged tweets
  - Issues:
    - Unless using "Firehose" method ($$$) can only collect random 1% of tweets
    - Can only go back 2 weeks

Scraping Twitter with $\mathbb{R}$

Twitter provides resources (APIs) for downloading Tweets:

- RESTful API:
  - Information about users and their *existing* tweets (static)
  - i.e. All tweets by Donald Trump, list of followers and friends, etc.
- Streaming API:
  - "Stream" of Tweets as they become available
  - i.e. Keyword-specific tweets, Geo-tagged tweets
  - Issues:
    - Unless using "Firehose" method ($$$) can only collect random 1% of tweets
    - Can only go back 2 weeks

Scraping Twitter with $\mathbb{R}$

Twitter provides resources (APIs) for downloading Tweets:

- RESTful API:
  - Information about users and their *existing* tweets (static)
  - i.e. All tweets by Donald Trump, list of followers and friends, etc.
- Streaming API:
  - "Stream" of Tweets as they become available
  - i.e. Keyword-specific tweets, Geo-tagged tweets
  - Issues:
    - Unless using "Firehose" method ($$$) can only collect random 1% of tweets
    - Can only go back 2 weeks

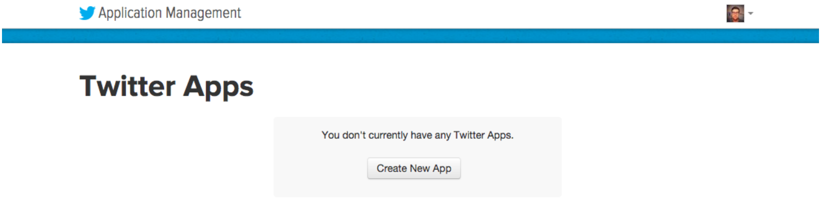
Scraping Twitter with $\mathbb{R}$

Twitter provides resources (APIs) for downloading Tweets:

- RESTful API:
  - Information about users and their *existing* tweets (static)
  - i.e. All tweets by Donald Trump, list of followers and friends, etc.
- Streaming API:
  - "Stream" of Tweets as they become available
  - i.e. Keyword-specific tweets, Geo-tagged tweets
  - Issues:
    - Unless using "Firehose" method ($$$) can only collect random 1% of tweets
    - Can only go back 2 weeks

Accessing Twitter's API

- Sign in to your twitter account and go to `https://apps.twitter.com/`

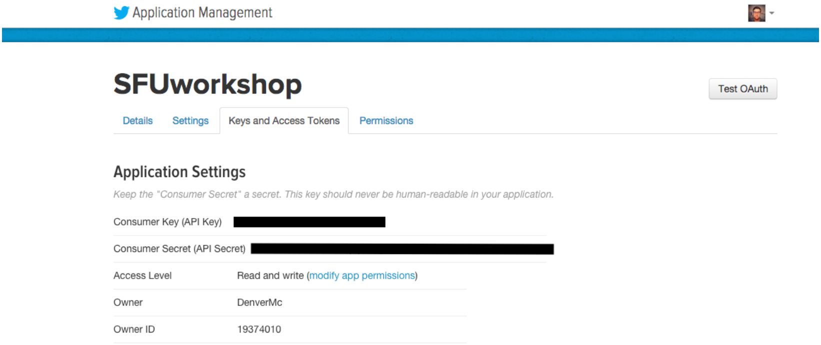


Accessing Twitter's API

No Twitter Account?
Username: sfu_workshop
Password: SFUworkshop1

Accessing Twitter's API

- Get your credentials: consumer key, consumer secret, access token, and access token secret



Working with Twitter APIs in $\mathbb{R}$

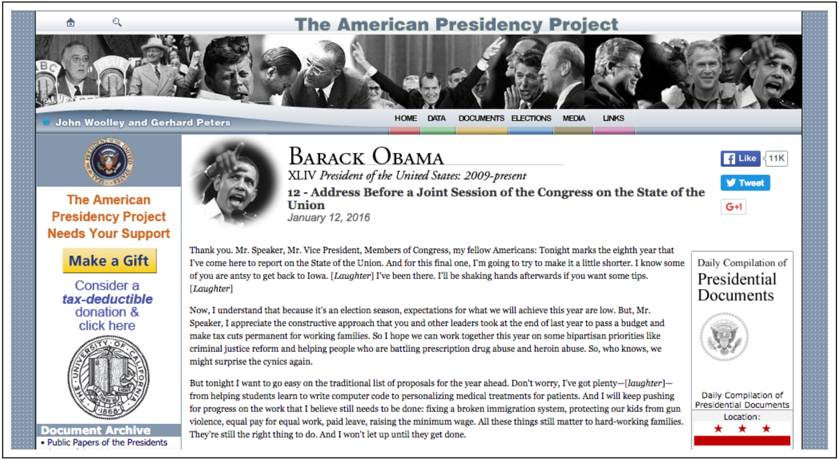
Open “Twitter.R”

```
getwd()
setwd("~/Desktop/Dropbox/Text Analysis
   Workshop/TextAnalysisWorkshop/1 - Twitter
    Scraping")
PackagesToInstall <- c("streamR","ROAuth","
   twitteR","ggplot2","devtools",
                         "RCurl", "wordcloud",
                             "tm")
install.packages(PackagesToInstall, repos =
   "http://cran.r-project.org")
```

Inspecting Website

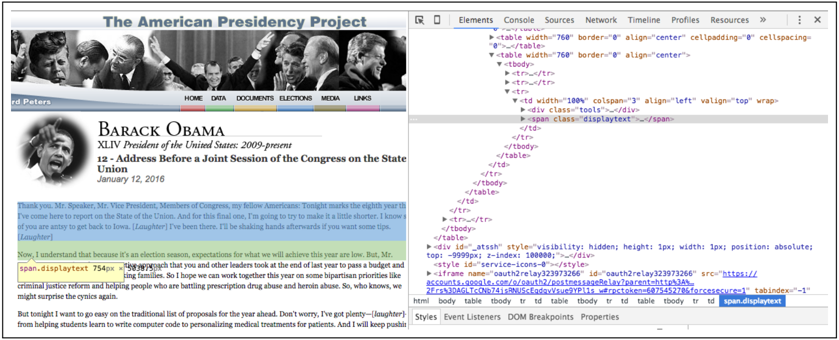
Obama's 2016 SOTU

www.presidency.ucsb.edu/ws/index.php?pid=111174



Inspecting Website

Inspecting Website



$\mathbb{R}$ Loops

Open "Webscraper.R"

```
for (item in seq_along(urls)){
  print(urls[item])
  data$text[[item]] <- html_nodes(read_html(
      urls[item]),
                          selector_name) %>%
  html_text()
}
```

Approaches to Automated Text Analysis

- Bag of Words
  - Does not take words' "context" into account
  - Either create Document-Term Matrix and sum word scores to look for similarities or classification schemes in documents or use simple dictionaries to count pre-classified words
  - *Lexicoder* and *Lexicoder Sentiment Dictionary* are examples
- Natural Language Processing
  - *Attempt* to take sentence structure and context into account
  - To have machine understand grammar, need to tag parts of speech (called entities)
  - Software suite from *Stanford NLP Group* an example

Approaches to Automated Text Analysis

- Bag of Words
  - Does not take words' "context" into account
  - Either create Document-Term Matrix and sum word scores to look for similarities or classification schemes in documents or use simple dictionaries to count pre-classified words
  - *Lexicoder* and *Lexicoder Sentiment Dictionary* are examples
- Natural Language Processing
  - *Attempt* to take sentence structure and context into account
  - To have machine understand grammar, need to tag parts of speech (called entities)
  - Software suite from *Stanford NLP Group* an example

Approaches to Automated Text Analysis

- Bag of Words
  - Does not take words' "context" into account
  - Either create Document-Term Matrix and sum word scores to look for similarities or classification schemes in documents or use simple dictionaries to count pre-classified words
  - *Lexicoder* and *Lexicoder Sentiment Dictionary* are examples
- Natural Language Processing
  - *Attempt* to take sentence structure and context into account
  - To have machine understand grammar, need to tag parts of speech (called entities)
  - Software suite from *Stanford NLP Group* an example

Approaches to Automated Text Analysis

- Bag of Words
  - Does not take words' "context" into account
  - Either create Document-Term Matrix and sum word scores to look for similarities or classification schemes in documents or use simple dictionaries to count pre-classified words
  - *Lexicoder* and *Lexicoder Sentiment Dictionary* are examples
- Natural Language Processing
  - *Attempt* to take sentence structure and context into account
  - To have machine understand grammar, need to tag parts of speech (called entities)
  - Software suite from *Stanford NLP Group* an example

Approaches to Automated Text Analysis

- Bag of Words
  - Does not take words' "context" into account
  - Either create Document-Term Matrix and sum word scores to look for similarities or classification schemes in documents or use simple dictionaries to count pre-classified words
  - *Lexicoder* and *Lexicoder Sentiment Dictionary* are examples
- Natural Language Processing
  - *Attempt* to take sentence structure and context into account
  - To have machine understand grammar, need to tag parts of speech (called entities)
  - Software suite from *Stanford NLP Group* an example

Approaches to Automated Text Analysis

- Bag of Words
  - Does not take words' "context" into account
  - Either create Document-Term Matrix and sum word scores to look for similarities or classification schemes in documents or use simple dictionaries to count pre-classified words
  - *Lexicoder* and *Lexicoder Sentiment Dictionary* are examples
- Natural Language Processing
  - *Attempt* to take sentence structure and context into account
  - To have machine understand grammar, need to tag parts of speech (called entities)
  - Software suite from *Stanford NLP Group* an example

Approaches to Automated Text Analysis

- Bag of Words
  - Does not take words' "context" into account
  - Either create Document-Term Matrix and sum word scores to look for similarities or classification schemes in documents or use simple dictionaries to count pre-classified words
  - *Lexicoder* and *Lexicoder Sentiment Dictionary* are examples
- Natural Language Processing
  - *Attempt* to take sentence structure and context into account
  - To have machine understand grammar, need to tag parts of speech (called entities)
  - Software suite from *Stanford NLP Group* an example

Approaches to Automated Text Analysis

- Bag of Words
  - Does not take words' "context" into account
  - Either create Document-Term Matrix and sum word scores to look for similarities or classification schemes in documents or use simple dictionaries to count pre-classified words
  - *Lexicoder* and *Lexicoder Sentiment Dictionary* are examples
- Natural Language Processing
  - *Attempt* to take sentence structure and context into account
  - To have machine understand grammar, need to tag parts of speech (called entities)
  - Software suite from *Stanford NLP Group* an example

Automated Sentiment Analysis using *Lexicoder*

- Bag of Words, dictionary-based approach to sentiment analysis
- *Lexicoder Sentiment Dictionary* has coded 4,500 words as either **positive** or **negative**
- Tone scores for each unit of analysis usually expressed as:

$$\frac{\#PositiveWords - \#NegativeWords}{Total\#Words} \times 100$$

Automated Sentiment Analysis using *Lexicoder*

- Bag of Words, dictionary-based approach to sentiment analysis
- *Lexicoder Sentiment Dictionary* has coded 4,500 words as either **positive** or **negative**
- Tone scores for each unit of analysis usually expressed as:

$$\frac{\#PositiveWords - \#NegativeWords}{Total\#Words} \times 100$$

Automated Sentiment Analysis using *Lexicoder*

- Bag of Words, dictionary-based approach to sentiment analysis
- *Lexicoder Sentiment Dictionary* has coded 4,500 words as either **positive** or **negative**
- Tone scores for each unit of analysis usually expressed as:

$$\frac{\#PositiveWords - \#NegativeWords}{Total\#Words} \times 100$$

Automated Sentiment Analysis using *Lexicoder*

Open “Webscraper.R”

```
dir.create("corpus")
L <- length(data$text)
for (i in 1:L) {
  t <- data[i,"text"]
  write.table(t, paste("corpus/",i,".txt",
     sep=""), sep="\t",col.names=F,row.names
     =F, quote = F)
}
```

Policy Positions from Text using *Wordfish*

- Assumption that politics exists on (uni-dimensional) policy continuum
- *Latent* policy positions can be estimated based on word occurrences
- Basic data structure is frequency matrix (we'll call it W)

  N Documents $\times V$ Words

  W_{ij} is number of times j appears in i
- Can use W_{ij} to (indirectly) estimate actors' policy positions (θ_i)

Policy Positions from Text using *Wordfish*

- Assumption that politics exists on (uni-dimensional) policy continuum
- *Latent* policy positions can be estimated based on word occurrences
- Basic data structure is frequency matrix (we'll call it W)

  N Documents $\times V$ Words

  W_{ij} is number of times j appears in i
- Can use W_{ij} to (indirectly) estimate actors' policy positions (θ_i)

Policy Positions from Text using *Wordfish*

- Assumption that politics exists on (uni-dimensional) policy continuum
- *Latent* policy positions can be estimated based on word occurrences
- Basic data structure is frequency matrix (we'll call it W)

  N Documents $\times V$ Words

  W_{ij} is number of times j appears in i
- Can use W_{ij} to (indirectly) estimate actors' policy positions (θ_i)

Policy Positions from Text using *Wordfish*

- Assumption that politics exists on (uni-dimensional) policy continuum
- *Latent* policy positions can be estimated based on word occurrences
- Basic data structure is frequency matrix (we'll call it W)

  N Documents $\times V$ Words

  W_{ij} is number of times j appears in i
- Can use W_{ij} to (indirectly) estimate actors' policy positions (θ_i)

Policy Positions from Text using *Wordfish*

- Assumption that politics exists on (uni-dimensional) policy continuum
- *Latent* policy positions can be estimated based on word occurrences
- Basic data structure is frequency matrix (we'll call it W)

  N Documents $\times V$ Words

  W_{ij} is number of times j appears in i
- Can use W_{ij} to (indirectly) estimate actors' policy positions (θ_i)

Policy Positions from Text using *Wordfish*

- Assumption that politics exists on (uni-dimensional) policy continuum
- *Latent* policy positions can be estimated based on word occurrences
- Basic data structure is frequency matrix (we'll call it W)

  N Documents $\times V$ Words

  W_{ij} is number of times j appears in i
- Can use W_{ij} to (indirectly) estimate actors' policy positions (θ_i)

Policy Positions from Text using *Wordfish*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
abroad	0	7	4	3	1	0	2	1	0	1	4	0	2	3	2	2	0	1	
access	2	1	0	2	2	0	1	4	4	0	1	1	1	0	0	2	0	2	
achieve	0	3	0	2	2	2	2	1	0	3	1	0	2	2	4	2	0	0	
achievement	0	0	0	1	1	1	1	2	1	0	0	0	1	1	2	2	0	1	
act	6	5	5	1	5	6	9	9	5	7	4	15	5	10	5	5	5	8	
action	0	2	3	0	13	2	2	3	2	4	2	0	1	3	2	0	2	2	
actions	1	0	0	0	0	3	0	1	0	0	1	2	2	0	0	2	0	1	
address	1	0	1	3	0	2	4	2	1	0	2	1	0	1	0	1	0	2	
administration	1	3	5	5	0	0	0	0	1	0	0	2	1	0	0	4	2	5	
advance	0	1	0	0	3	3	2	1	0	0	0	0	4	2	1	1	0	1	
afford	1	0	1	2	1	2	0	0	0	0	0	2	1	2	2	1	0	7	
affordable	1	1	3	0	1	1	4	9	0	1	1	1	2	2	3	1	0	3	
afghanistan	0	0	0	0	0	0	0	0	0	13	3	5	3	2	4	4	0	3	
africa	0	1	1	0	1	1	2	2	0	2	7	0	0	0	2	0	0	0	
agenda	0	1	1	0	0	2	1	1	1	0	0	2	1	2	0	0	0	0	
ago	3	3	9	3	2	3	7	5	3	1	2	1	6	1	2	3	0	4	
agree	2	1	4	6	3	0	1	2	3	0	0	0	1	0	0	1	2	2	
agreement	2	0	1	3	2	2	1	3	0	0	1	0	0	0	0	10	0	1	
ahead	1	3	2	2	2	5	2	2	1	2	2	1	1	0	3	4	0	2	
air	0	1	1	4	0	1	2	2	0	1	2	0	1	0	1	0	0	0	

Showing 1 to 21 of 592 entries

Policy Positions from Text using *Wordfish*

Open "Webscraper.R"

```
doc.corpus = Corpus(VectorSource(data$text))
doc.corpus <- tm_map(doc.corpus, content_
   transformer(tolower), mc.cores=1)
doc.corpus <- tm_map(doc.corpus,
   removeNumbers, mc.cores=1)
doc.corpus <- tm_map(doc.corpus, removeWords
   , stopwords("SMART"), mc.cores=1)
doc.corpus <- tm_map(doc.corpus, removeWords
   , stopwords("english"), mc.cores=1)
doc.corpus <- tm_map(doc.corpus,
   removePunctuation, mc.cores=1)
```

What is Machine Learning?

a general inductive process automatically builds a classifier by learning, from a set of preclassified documents, the characteristics of the categories... The advantages of this approach over the knowledge engineering approach (consisting in the manual definition of a classifier by domain experts) are a very good effectiveness, considerable savings in terms of expert manpower, and straightforward portability to different domains.

Supervised Learning Workflow

- Manually code subset of data
- Train multiple algorithms on subset of manually coded data
- Test accuracy of algorithms on subset of manually coded data *not* used for training
- Examine model statistics:
  - Precision → How often a case the algorithm predicts as belonging to a class actually belongs to that class
  - Recall → Proportion of units in a class the algorithm correctly assigns to that class
  - F-Score → Weighted average of precision and recall

Supervised Learning Workflow

- Manually code subset of data
- Train multiple algorithms on subset of manually coded data
- Test accuracy of algorithms on subset of manually coded data *not* used for training
- Examine model statistics:
  - Precision → How often a case the algorithm predicts as belonging to a class actually belongs to that class
  - Recall → Proportion of units in a class the algorithm correctly assigns to that class
  - F-Score → Weighted average of precision and recall

Supervised Learning Workflow

- Manually code subset of data
- Train multiple algorithms on subset of manually coded data
- Test accuracy of algorithms on subset of manually coded data *not* used for training
- Examine model statistics:
  - Precision → How often a case the algorithm predicts as belonging to a class actually belongs to that class
  - Recall → Proportion of units in a class the algorithm correctly assigns to that class
  - F-Score → Weighted average of precision and recall

Supervised Learning Workflow

- Manually code subset of data
- Train multiple algorithms on subset of manually coded data
- Test accuracy of algorithms on subset of manually coded data *not* used for training
- Examine model statistics:
  - Precision → How often a case the algorithm predicts as belonging to a class actually belongs to that class
  - Recall → Proportion of units in a class the algorithm correctly assigns to that class
  - F-Score → Weighted average of precision and recall

Supervised Learning Workflow

- Manually code subset of data
- Train multiple algorithms on subset of manually coded data
- Test accuracy of algorithms on subset of manually coded data *not* used for training
- Examine model statistics:
  - Precision → How often a case the algorithm predicts as belonging to a class actually belongs to that class
  - Recall → Proportion of units in a class the algorithm correctly assigns to that class
  - F-Score → Weighted average of precision and recall

Supervised Learning Workflow

- Manually code subset of data
- Train multiple algorithms on subset of manually coded data
- Test accuracy of algorithms on subset of manually coded data *not* used for training
- Examine model statistics:
  - Precision $\rightarrow$ How often a case the algorithm predicts as belonging to a class actually belongs to that class
  - Recall $\rightarrow$ Proportion of units in a class the algorithm correctly assigns to that class
  - F-Score $\rightarrow$ Weighted average of precision and recall

Supervised Learning Workflow

- Manually code subset of data
- Train multiple algorithms on subset of manually coded data
- Test accuracy of algorithms on subset of manually coded data *not* used for training
- Examine model statistics:
  - Precision $\rightarrow$ How often a case the algorithm predicts as belonging to a class actually belongs to that class
  - Recall $\rightarrow$ Proportion of units in a class the algorithm correctly assigns to that class
  - F-Score $\rightarrow$ Weighted average of precision and recall