Intro to Text Analysis

Sebastián Vallejo

12/5/2019

Contents

1 Outline of the workshop

1. Text as data: Brief overview of the theory behind text analysis
2. The corpus
3. Analyzing our corpora

- Features
- Dictionary-based approaches
- Estimating distance
- Topic models
- Natural language processing (NLP)
- Text networks
- Neural networks / word embeddings

4. Conclusion and parting remarks

Note: The point of this workshop is not to for you to leave an expert on text analysis, but rather for you to have a taste of what can be achieved (i.e. what substantive questions can be answered) when using text analysis techniques. The code provided can help you get started, but you will need to explore each method more in detail if you want to apply it to your research.

1.1 What will you need?

If you want to follow along in your computer, you should have spaCyR intalled. In addition to spaCyR, you should have the following packages installed:

- quanteda
- quanteda.dictionaries
- stm
- tidyverse
- ggplot2
- lme4
- lattice
- ggthemr (to prettify)
- dplyr

Text Analysis is a computer-intensive task. spaCyR and quanteda processes can consume a lot of your RAM, so take that into account when running your code.

1.2 What are my sources?

Much of the material/ideas for the workshop were taken from the following sources:

- Kenneth Benoit's website
- The spaCyR GitHub page
- Chris Bail's Course
- Elliot Ash's Course
- Will Lowe's Text Analysis workshop at IQMR
- The Internet

If you are interested in applied and/or theoretical readings on text analysis, here is a short list to get you started:

- Grimmer and Stewart (2013) - Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts **in Political Analysis**
- Lucas et al. (2015) - Computer-Assisted Text Analysis for Comparative Politics **in Political Analysis**

- Blei (2012) - Probabilistic Topic Models **in Communications of the ACM**
- **Poetics**, Volume 41, Special Issue on "Topic Models and the Cultural Sciences"
- Slapin and Proksch (2008) - A Scaling Model for Estimating Time-Series Party Positions from Texts **in AJPS**
- Welbers et al. (2017) - Text Analysis in R **in Communication Methods and Measures**

Let's start!

2 Text as data: Some theory

2.1 Why text?

1. Politicians *love* to speak.
2. Bureaucrat *love* to write.
3. Machirulos *love* to tweet.

There is text in every aspect of politics: debates on legislation, peace treaties, news reports, political manifestos, campaign speeches, social media, etc. Not only is text ubiquitous, but it is produced at the time (sometimes in real time).

2.2 Why use the help of a computer?

1. Humans are great at understanding and analyzing the content of texts. Computers are not.
2. Humans are also great at not being able to read thousands of documents in minutes, organize that text, classify that text, scale that text, and then produce pretty graphics from that text. Computers are not not that. (I stand by the double negative)

We will be learning about the latter.

Note: Text analysis is not a field, it is a tool. Think about it as a regression where the data are words instead of numbers. Thus, the fanciest of text analysis techniques is no good without a well thought out, substantive, theoretically motivated question.

2.3 Four principles of automated text analysis (from Grimmer and Stewart (2013))

1. All Quantitative Models of Language Are Wrong—But Some Are Useful.

- Data generation process for text is unknown.
- Language it too complex for computers to correctly decipher (e.g. "Time flies like an arrow. Fruit flies like a banana.").
- Since language is so context-specific, more complex models are rarely more useful for the analysis of texts.

2. Quantitative Methods Augment Humans, Not Replace Them

- You need to read the text, know the text, be the text.
- Computers organize, direct, and suggest.
- Humans read and interpret.

3. There Is No Globally Best Method for Automated Text Analysis

- Different needs require different methods.
- Even when you have the same needs, the same model might not fit the data.

4. Validate, Validate, Validate

- Outputs can be misleading (or simply wrong).
- It is incumbent upon you, the researcher, to validate the use of automated text analysis.

Now we can start with some (sample) text analysis. Note that we will be doing basic analysis (as generalizable as it gets) of text, and then use some models as examples. To see the extent of what can be done with different models, I suggest you read Grimmer and Stewart (2013).

3 The corpus (in theory)

To analyze text, we first need a corpus, a large and structured set of texts for analysis. The units (texts) of the "structured set of texts" can be anything you want them to be: a complete speech, each paragraph of the speech, or each sentence within that paragraph. The relevance of the unit will depend on your research question.

3.1 Text as data: Some specifics

Text data is a sequence of characters called **documents**. The set of documents is the **corpus**.

- Text data is **unstructured**
- There is information we want, mixed in with (A LOT) of information we do not.
- We need to separate the wheat from the chaff.

Note: All text analysis methods will produce some information. The *art* lies on the ability of the researcher to figure out what is valuable and what is not.

3.2 What counts as a document?

The unit of analysis when using text as data will depend on the question you are asking. For example:

- If you are looking at how politicians react to different types of economic crises, then each document produced after a crisis would be the unit of analysis.
- If you are looking at how politicians differ within a campaign, then you might aggregate all the texts produced by a candidates during a campaign as the unit of analysis.
- If you are looking at how politicians address different topics within a campaign, then your unit of analysis might be a section or paragraph of a manifesto.

3.3 Where can I get my hands on some juicy corpora?

1. Chris Bail curates a list of corpora already compiled and ready to use.
2. Governments produce text ALL THE TIME. It is, usually, publicly available and, depending on the country/organization, easily accessible.
3. Scrape the webs. Websites can make it difficult to scrape data with restrictive terms of use (i.e. bot-blockers or, worse, javascript). There are creative ways to get around these. (If you are interested in web-scrapping, let me know and I can help you get started with some code.)

3.4 Some final words on documents

Original corpora are rarely ready to be analyzed without some previous cleaning. You often want to get rid of hyphenations at line brakes, table of contents, indexes, etc. All of these are corpus-specific and require attention ahead of time.

- Learn how to use regular expressions (regex). The `stringr` package in R or the Python package *re* are useful tools.
- While useful, regex is tedious to learn. Check out Sanchez 2013 for a good guide.

Some data are only available as non-searchable PDFs or images. These need to be converted to text before R (or Python) can read them. I use ABBYY FineReader, which is 'expensive' but might be available at your (old/next) university library. Joe Sutherland has an open-source OCR (Optical Character Recognition).

Finally, I advise against using spell checkers. Most corpora use specialized language that would be flagged by standard spell-checkers (and I have not found one that can be automated to check text in Spanish). In most empirical contexts, we can safely assume that spelling errors (especially OCR errors) are uncorrelated with treatment assignment.

(Ask me about some of the other strengths weaknesses of using data from social media.)

4 The corpus (in practice)

We need texts for text analysis. Luckily, we have a dataset of 5640 tweets from 3885 unique users containing the word 'capitalism'. The dataset has been tinkered with and cleaned and is ready to be processed.

Before we start, let's load the required packages. Remember that spaCyR is a Python wrapper that needs to first be initialized.

```
rm(list=ls(all=TRUE))
library(tidyverse)
library(spacyr)
library(stm)
library(quanteda)
library(quanteda.dictionaries)
library(dplyr)
library(lattice)
library(ggplot2)
library(ggrepel)
library(lme4)
library(ggthemr)
ggthemr("fresh")
spacy_initialize()
```

Let's load our dataset and see how it looks.

```
load("data_capitalism.Rdata")
data_capitalism <- data_capitalism[!data_capitalism$text_clean=="",]
data_capitalism %>% glimpse()
```

```
## Observations: 5,627
## Variables: 13
## $ text          <chr> "if you're having trouble understanding capitalism j...
## $ friendsRT     <int> 637, 576, 674, 116, 2937, 7721, 11768, 3011, 2838, 1...
## $ followersRT   <int> 1126, 3407, 36590, 5080, 6088, 12074, 15690, 3277, 1...
## $ timeRT        <dbl> 18195, 18228, 18223, 18230, 18226, 18222, 18223, 182...
## $ nameauth      <chr> "shnupz", "aniceburrito", "CaseyExplosion", "HairyBo...
## $ likeRT        <int> 45195, 4349, 2850, 57, 78, 183, 34, 110, 3750, 133, ...
## $ retweetRT     <int> 7258, 590, 1031, 5, 16, 41, 16, 44, 1113, 55, 12, 90...
## $ to_membership <dbl> 3, 3, 3, 3, 6, 6, 6, 3, 6, 6, 6, 6, 6, 6, 6, 6, 9, 9...
## $ to_ind        <dbl> 82, 115, 238, 3, 11, 17, 12, 280, 282, 16, 8, 98, 98...
## $ mem_name      <chr> "POC Anti-Capitalism", "POC Anti-Capitalism", "POC A...
```

```
## $ mem_name_dummy <chr> "Anti-Capitalism", "Anti-Capitalism", "Anti-Capitali...
## $ text_clean     <chr> "if you're having trouble understanding capitalism j...
## $ date_created   <date> 2019-10-26, 2019-11-28, 2019-11-23, 2019-11-30, 201...
```

To analyze the text found in the dataset, we need to create a `corpus` object. The `corpus()` function (from the quanteda package) does just this. The main object has to be a character vector. (The `readtext()` function from the readtext package can read text-formatted files).

```r
cap_corp <- corpus(data_capitalism$text_clean,
                   docvars = data.frame(author = data_capitalism$nameauth,
                                        time = data_capitalism$timeRT,
                                        in_degree = data_capitalism$to_ind,
                                        followers = data_capitalism$followersRT,
                                        RT_count = data_capitalism$retweetRT,
                                        left_right = data_capitalism$mem_name,
                                        left_right_dum = data_capitalism$mem_name_dummy),
                   metacorpus = list(source = "Twitter",
                                     notes = "Scrapped on Nov. 30 - Dec. 3, 2019"))

summary(cap_corp,10)
```

```
## Corpus consisting of 5627 documents, showing 10 documents:
##
##    Text Types Tokens Sentences          author  time in_degree followers
##   text1    22     23         1          shnupz 18195        82      1126
##   text2    13     14         2    aniceburrito 18228       115      3407
##   text3    44     53         2  CaseyExplosion 18223       238     36590
##   text4     8      8         1 HairyBoomerDude 18230         3      5080
##   text5    36     51         4  brianjtrautman 18226        11      6088
##   text6    10     10         1        LostDiva 18222        17     12074
##   text7    39     55         3      safeagain1 18223        12     15690
##   text8    22     29         2        lvl28mage 18229       280      3277
##   text9     7      7         1     davidsirota 18226       282    153820
##  text10    17     20         1         jayasax 18222        16     18902
##  RT_count            left_right  left_right_dum
##      7258   POC Anti-Capitalism Anti-Capitalism
##       590   POC Anti-Capitalism Anti-Capitalism
##      1031   POC Anti-Capitalism Anti-Capitalism
##         5   POC Anti-Capitalism Anti-Capitalism
##        16 White Anti-Capitalism Anti-Capitalism
##        41 White Anti-Capitalism Anti-Capitalism
##        16 White Anti-Capitalism Anti-Capitalism
##        44   POC Anti-Capitalism Anti-Capitalism
##      1113 White Anti-Capitalism Anti-Capitalism
##        55 White Anti-Capitalism Anti-Capitalism
##
## Source: Twitter
## Created: Sun Dec  8 20:19:46 2019
## Notes: Scrapped on Nov. 30 - Dec. 3, 2019
```

A corpus object works similar to a normal dataset, in that each document (in our case, each tweet) is an observation that has additional covariates (i.e. *docvars*) that describe it.

```r
cap_corp[c(1:5)]
```

```
##
```

```
##
##
##
##
##                              "\"Fascism is capitalism in decay.\"\n\nI didn't even understand what tha
##
##
##
## "@davidsirota No stage - it's just capitalism. Stages aren't distinguishable with capitalism. Only i
```

You can explore a corpus as you would explore other lists (with brackets). To get all the texts you can `texts(cap_corp)`, but you don't want to do that. We can subset a corpus. Here are all the tweets from TW authorities with more than 90K followers (and less than 200 words... I know, useless, but you might appreciate the code to do it):

```
cap_subset <- corpus_subset(cap_corp,
                               followers > 90000 & ntype(cap_corp) < 200)
summary(cap_subset, 15)
```

```
## Corpus consisting of 424 documents, showing 15 documents:
##
##    Text Types Tokens Sentences          author  time in_degree followers
##   text9     7      7         1     davidsirota 18226       282    153820
##  text11    16     16         2    mikebarnicle 18225         8    133427
##  text12    25     52         4       peterdaou 18231        98    281684
##  text13    27     28         2       peterdaou 18232        98    281675
##  text17    14     16         3 RedNationRising 18229       118    170751
##  text18    24     25         1  KurtSchlichter 18232       131    211589
##  text19    33     46         3   LacyJohnsonMN 18229       258    103051
##  text20    39     48         4     ACTBrigitte 18227       284    193371
##  text25    29     33         2   BreitbartNews 18232       350   1207329
##  text27    19     19         1  BridgetPhetasy 18231       756    153876
##  text54    40     47         3     iheartmindy 18231       362    161898
##  text55    21     25         2    thecjpearson 18229        90    320951
##  text56    20     22         2      NikkiHaley 18225       294    524277
##  text60    24     25         2  SandraSentinel 18229        59    169095
##  text61    46     57         5   mikandynothem 18232       168    220387
##  RT_count            left_right  left_right_dum
##      1113 White Anti-Capitalism Anti-Capitalism
##        12 White Anti-Capitalism Anti-Capitalism
##        90 White Anti-Capitalism Anti-Capitalism
##       355 White Anti-Capitalism Anti-Capitalism
##       717            Capitalism      Capitalism
##       528            Capitalism      Capitalism
##      1352            Capitalism      Capitalism
##      2074            Capitalism      Capitalism
##      1727            Capitalism      Capitalism
##      4115            Capitalism      Capitalism
##      1093            Capitalism      Capitalism
##        61            Capitalism      Capitalism
##      2741            Capitalism      Capitalism
##        24            Capitalism      Capitalism
##       554            Capitalism      Capitalism
##
## Source: Twitter
```

```
## Created: Sun Dec  8 20:19:46 2019
## Notes: Scrapped on Nov. 30 - Dec. 3, 2019
```

We might be interested in sentences rather than complete tweets.

```
cap_sentences <- corpus_reshape(cap_corp, to = "sentences") # or "paragraphs"
summary(cap_sentences,8)
```

```
## Corpus consisting of 12172 documents, showing 8 documents:
##
##     Text Types Tokens Sentences          author  time in_degree followers
##  text1.1    22     23         1          shnupz 18195        82      1126
##  text2.1    12     13         1    aniceburrito 18228       115      3407
##  text2.2     1      1         1    aniceburrito 18228       115      3407
##  text3.1     7      8         1  CaseyExplosion 18223       238     36590
##  text3.2    39     45         1  CaseyExplosion 18223       238     36590
##  text4.1     8      8         1 HairyBoomerDude 18230         3      5080
##  text5.1     8      8         1  brianjtrautman 18226        11      6088
##  text5.2     6      6         1  brianjtrautman 18226        11      6088
##  RT_count            left_right  left_right_dum
##      7258   POC Anti-Capitalism Anti-Capitalism
##       590   POC Anti-Capitalism Anti-Capitalism
##       590   POC Anti-Capitalism Anti-Capitalism
##      1031   POC Anti-Capitalism Anti-Capitalism
##      1031   POC Anti-Capitalism Anti-Capitalism
##         5   POC Anti-Capitalism Anti-Capitalism
##        16 White Anti-Capitalism Anti-Capitalism
##        16 White Anti-Capitalism Anti-Capitalism
##
## Source: Twitter
## Created: Sun Dec  8 20:19:47 2019
## Notes: corpus_reshape.corpus(cap_corp, to = "sentences")
```

```
cap_sentences[1:5]
```

```
##
##                                                                                                                                                 "if you'
##
##
##
##
##
##
##
## "I didn't even understand what that meant a few years back, but the wealthy funnelling money to whit
```

Since these are tweets and people often tweet one sentence at a time, this conversion might be moot for our corpus. But in longer texts narrowing down the unit of analysis can be helpful, especially if we are trying to estimate topic models (more on this later).

The reshape function can divide texts into "sentences" and "paragraphs". `corpus_reshape()` uses punctuation marks (e.g. "\\n", "\n") to determine cuts.

5 Pre-processing the corpus

As previously mentioned, our corpora have the information we want, and a lot of information we do not. Uninformative data add noise and reduce the precision of resulting estimates (and are computationally costly). We aim to have a "bag-of-words", or to convert a corpus D to a matrix X. In the "bag-of-words" representation, a row of X is just the frequency distribution over words in the document corresponding to that row.

Before we do that, we will get rid of all the unwanted information. First, we turn all words to lower-case and get rid of all punctuation and numbers. The `tokens()` command will do this and separate all the texts into tokens.

```
cap_toks <- tokens(cap_corp,
                   remove_numbers = T,
  remove_punct = T,
  remove_separators = TRUE,
  remove_twitter = T,
  remove_hyphens = T,
  remove_url = T)
```

Tokens (`ntoken`) is a fancy name for "word". These contain all the information we need to run our models. Yet, there is still a lot of noise.

```
cap_coll <- textstat_collocations(cap_toks)
head(cap_coll, 20)
```

```
##            collocation count count_nested length   lambda        z
## 1        capitalism is   789            0      2 2.362610 54.56040
## 2        of capitalism   686            0      2 1.818068 41.17951
## 3          free market    88            0      2 6.805751 40.79497
## 4              this is   285            0      2 2.898294 40.58134
## 5                to be   264            0      2 2.877196 37.78918
## 6       climate change    75            0      2 8.099856 37.64941
## 7               if you   145            0      2 3.750860 37.62064
## 8              we have   130            0      2 3.619659 35.53517
## 9        working class    57            0      2 6.626296 35.23928
## 10              in the   450            0      2 1.912162 34.93206
## 11          late stage    57            0      2 8.194708 34.43979
## 12             we need    92            0      2 4.611996 34.13644
## 13    under capitalism   196            0      2 3.900645 33.38231
## 14        mark ruffalo    78            0      2 9.382803 32.39438
## 15                is a   362            0      2 1.881855 32.06737
## 16 economic revolution    47            0      2 6.941818 32.06536
## 17          people who    89            0      2 3.863746 31.64427
## 18          those who    59            0      2 5.106866 31.18187
## 19           should be    65            0      2 4.770880 30.16049
## 20            has been    57            0      2 4.918624 29.77172
```

(Quick detour: Collocations bundle together a set number of words -also known as ngrams- that appear next to each other. The default is 2, but we can set it at any length we want.)

Some words are "useless". Let's get rid of the *stopwords*. (For a take on when stopwords are informative, check Pennebaker (2011).)

```
cap_toks_stop <- tokens_remove(cap_toks,
                               stopwords(language = "en"),
                               padding = F)
```

```r
cap_toks_stop <- tokens_remove(cap_toks_stop, "amp")
cap_toks_stop <- tokens_remove(cap_toks_stop, "capitalism")
cap_coll <- textstat_collocations(cap_toks_stop)
head(cap_coll, 20)
```

```
##             collocation count count_nested length   lambda        z
## 1           free market    88            0      2 6.176197 36.84064
## 2        climate change    75            0      2 7.462913 34.54893
## 3         working class    57            0      2 5.979366 31.75400
## 4            late stage    57            0      2 7.748096 31.36852
## 5          mark ruffalo    78            0      2 8.744120 30.13495
## 6   economic revolution    47            0      2 6.304274 29.00872
## 7            killing us    52            0      2 5.755766 25.98845
## 8   ethical consumption    35            0      2 8.636356 25.37620
## 9      means production    26            0      2 6.504765 23.97048
## 10        ruffalo calls    26            0      2 7.036844 23.82651
## 11       bernie sanders    30            0      2 8.042982 23.30022
## 12       calls economic    31            0      2 6.596260 23.21391
## 13         social media    27            0      2 5.845772 23.12489
## 14      economic system    39            0      2 4.323610 22.97572
## 15        everyone else    24            0      2 5.697881 22.19599
## 16       climate crisis    24            0      2 5.528306 21.88571
## 17        worth million    18            0      2 6.113556 20.61271
## 18          gift buying    16            0      2 7.764324 20.54672
## 19      white supremacy    37            0      2 8.370088 20.29066
## 20            years ago    25            0      2 8.068831 20.26880
```

Finally, we might want to stem our tokens.

```r
cap_toks_stem <- tokens_wordstem(cap_toks_stop)
cap_coll <- textstat_collocations(cap_toks_stem)
head(cap_coll, 20)
```

```
##         collocation count count_nested length    lambda        z
## 1       free market   110            0      2  6.018092 40.29298
## 2      climat chang    74            0      2  7.027756 35.06895
## 3        late stage    57            0      2  7.704935 31.43510
## 4      mark ruffalo    86            0      2  8.721140 31.33799
## 5     econom revolut    47            0      2  6.072575 28.60968
## 6           kill us    58            0      2  4.635673 27.88024
## 7        work class    59            0      2  4.353081 26.88487
## 8     ethic consumpt    34            0      2  8.028629 25.64903
## 9      berni sander    30            0      2  7.928019 23.15127
## 10      call econom    34            0      2  4.597643 22.79844
## 11    econom system    40            0      2  4.051867 22.18342
## 12      everyon els    24            0      2  5.605248 21.96526
## 13     climat crisi    24            0      2  5.536514 21.90668
## 14       rule class    25            0      2  5.482741 21.65451
## 15     ruffalo call    26            0      2  5.105934 21.54630
## 16     mean product    26            0      2  4.889848 21.19389
## 17 white supremaci    37            0      2  8.224788 20.93996
## 18        net worth    22            0      2  8.152715 20.46240
## 19 leecamp sensand    24            0      2 10.059114 19.97046
## 20     black friday    55            0      2  8.993914 19.66205
```

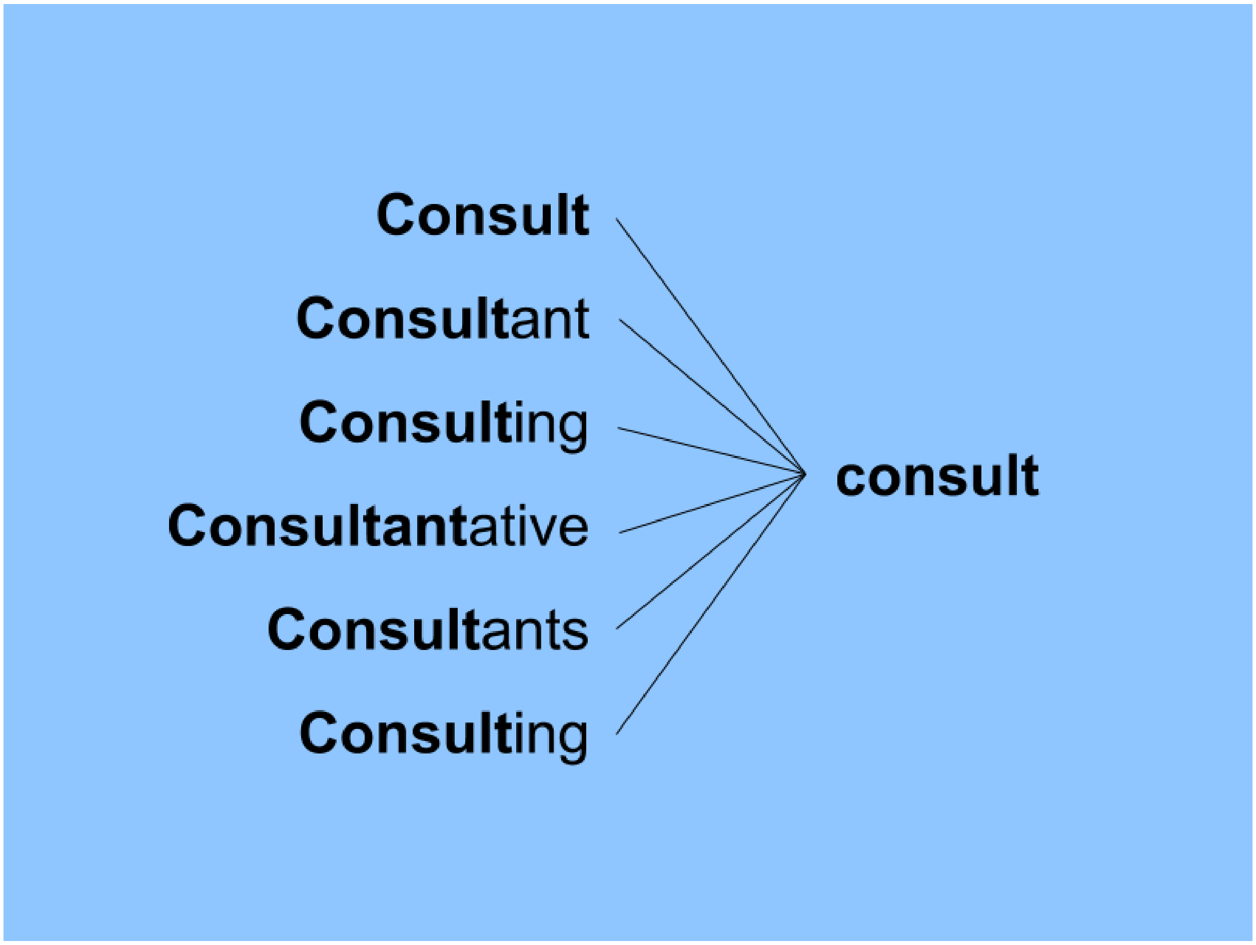


Figure 1: Example of stemming

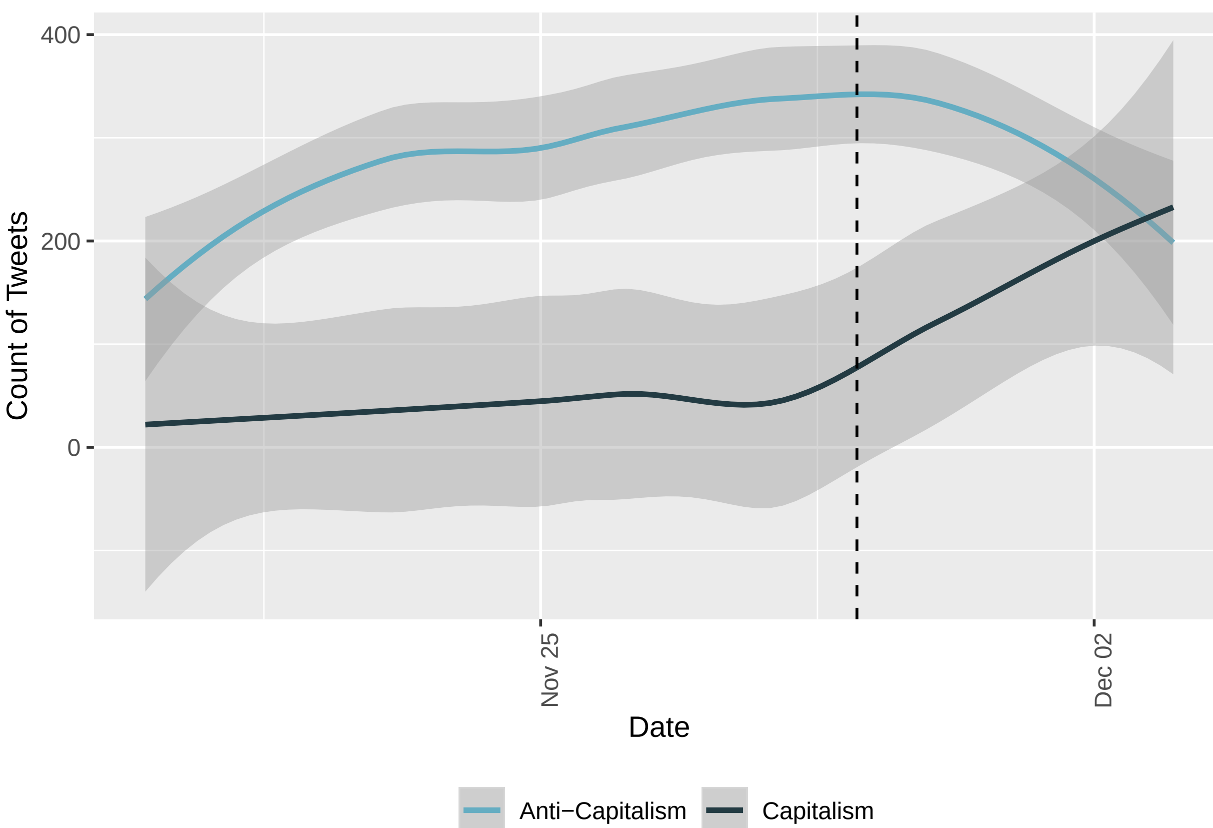
5.1 Before the "bag-of-words"

We have *just* pre-processed the data, but the number of documents (e.g. tweets) and the (pre-processed) length of these documents already provide an interesting set of variables for analysis.

For example: - How do major capitalist events affect the production of tweets from capitalists and anti-capitalists? - What is the relation between in-degree and effort and quality of tweets?

```
# How do major capitalist events affect the production of tweets from capitalists and anti-capitalists?

data_capitalism <- data_capitalism %>%
  group_by(date_created,mem_name_dummy) %>%
   mutate(count_tweets = n())

data_capitalism_res <- data_capitalism[!duplicated(data_capitalism[c(11,13)]),]
data_capitalism_res <- data_capitalism_res[data_capitalism_res$date_created > "2019-11-19",]

ggplot(data_capitalism_res, aes(x=date_created , y = count_tweets , color = mem_name_dummy)) +
         stat_smooth() +
    scale_color_discrete(name = "") +
  labs(x = "Date", y = "Count of Tweets") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1),
        legend.position="bottom") +
  geom_vline(xintercept = as.Date("2019-11-29"), color = "black", linetype = "dashed")
```

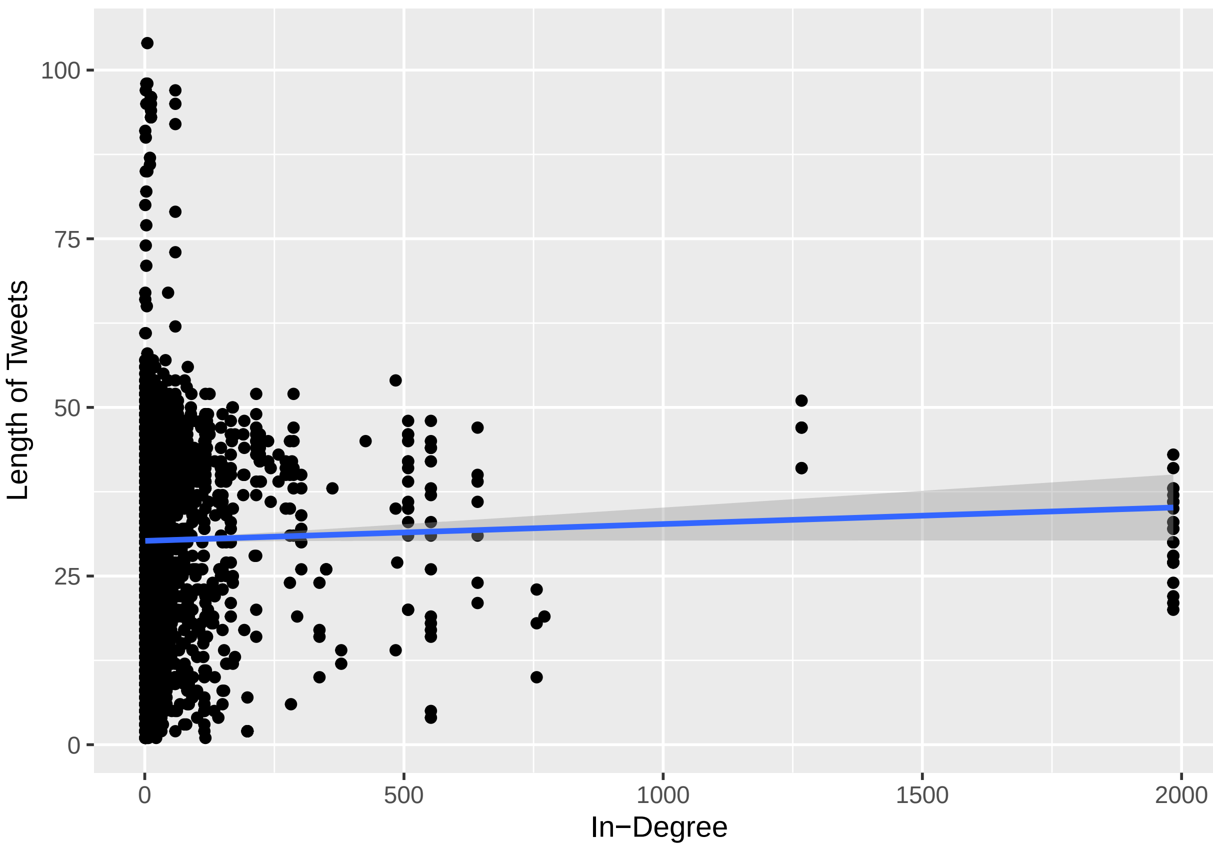
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



the Data I-1.bb

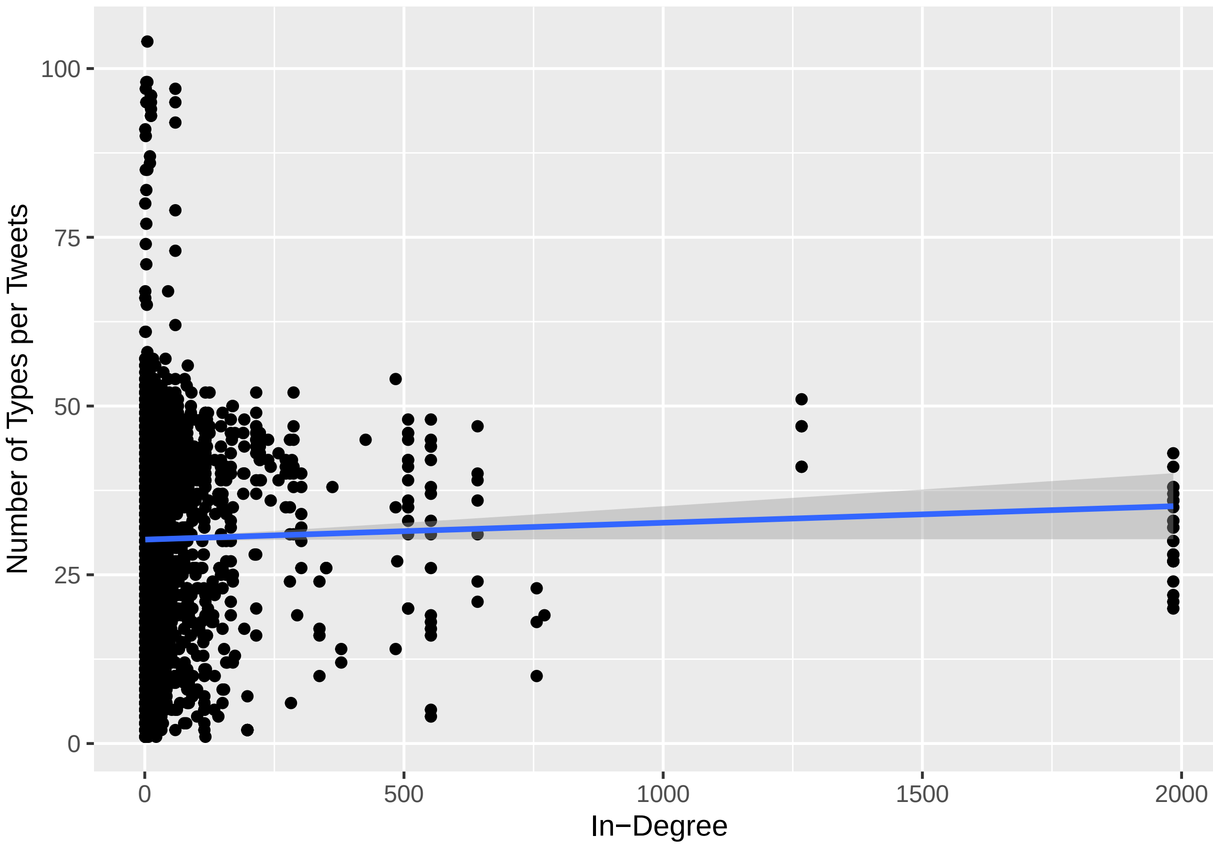
```
# What is the relation between in-degree and effort of tweets?
```

```r
data_capitalism$length_tweet <- ntoken(char_tolower(data_capitalism$text_clean),
                                        remove_punct = TRUE)

data_capitalism$length_type <- ntoken(char_tolower(data_capitalism$text_clean),
                                       remove_punct = TRUE) #repeated words are cut..

ggplot(data_capitalism, aes(x=to_ind , y =  length_tweet)) +
         geom_point() +
     geom_smooth(method = "lm", se = T)+
  labs(x = "In-Degree", y = "Length of Tweets")
```



the Data I-2.bb

```r
ggplot(data_capitalism, aes(x=to_ind , y =  length_type)) +
         geom_point() +
     geom_smooth(method = "lm", se = T)+
  labs(x = "In-Degree", y = "Number of Types per Tweets")
```



the Data I-3.bb

6 Analyzing our corpus: Keywords in context and collocations

Finally, we are ready to analyze our corpus. We will start by the basic: keywords in context.

6.1 Keywords in context

Say we are interested in the way capitalist and anti-capitalist describe **capitalism**. Let's see how they talk about **capitalism** using the "keyword-in-context" function:

```
cap_talk <- kwic(cap_toks_stop, "capital*", window=10) # 10 words before and after the word in question
head(cap_talk, 20)
```

```
##
##    [text35, 3]
##   [text44, 20]
##    [text64, 3]
##   [text64, 16]
##   [text69, 17]
##   [text101, 1]
##  [text102, 18]
##  [text111, 12]
##   [text125, 7]
##  [text131, 24]
##   [text136, 3]
##  [text136, 14]
```

```
## [text137, 18]
##  [text150, 1]
##  [text151, 1]
##  [text159, 6]
##  [text160, 1]
##  [text164, 2]
## [text168, 17]
## [text171, 11]
##
##                                                                      know Fascism
##    services socialism reality whatever programs creates always constantly threat ruling
##                                                                        Every time
##                calls population control matter dress rank pseudo science used obscure
##               financial weight stop Luthor Corp various drug lords parents sycophants
##
##               Finance rules benefit elite wealthy leave masses behind struggling wake
## systematically designed extremely powerful counter revolutionary force many ways global
##                         Unpopular opinion think qualities fetishize entrepreneurship
##                         coup attempt Panama weeks later US invaded overthrew gov CIA
##                                                                   Arguing defence
##                  ongoing effort suck blood poor giving junk return argument privilege
##                        leaguer Rhodes Scholar gone work anywhere world chose go work
##
##
##                                                      Lech Walesa Harvard deplore US
##
##                                                                             Among
##                swerves accelerates retreats depending routes open tolls lower one get
##   wrote biggest difference Bernie Sanders Elizabeth Warren views constitutes corruption
##
## | Capitalism's |
## |  capitalist  |
## |   Capital    |
## | capitalism's |
## |  capitalist  |
## | Capitalism's |
## | Capitalism's |
## | capitalism's |
## |  capitalist  |
## | Capitalism's |
## | capitalism's |
## | capitalism's |
## | capitalism's |
## | Capitalism's |
## | Capitalism's |
## | capitalism's |
## | Capitalism's |
## | capitalism's |
## |   capital    |
## |  capitalist  |
##
## life support system decay activate stomp dissidents alternatives recoils back
## class
## approaches crisis calls population control matter dress rank pseudo science
```

```
##  role ecocide set scientific precedent genocide
##  values Comrade Wayne enough sabatoge efforts thwart x
##  Failures Ignited Worldwide Protests
##  inability answer social economic needs people
##  crown jewel really need stop looking liberatory art constantly disappointed
##  success behaviors prop rape culture example saw book business section
##  Invisible Army
##  ongoing effort suck blood poor giving junk return argument privilege
##  bidding Poor people deserve nice things value money just like
##  despicable mercenaries McKinsey co call sell call zealot
##  Housing Crisis Socialist Alternative Democrat Party forces America sees calling
##  Failures Ignited Worldwide Protests TheRealNews
##  global decline course decline good empire Self congratulatory delusions
##  self destruction US economy's future depends increasingly quality quantity educated
##  basic injustices small minority employers can hire fire majority employees
##  profitable destination faster One woman's corruption another woman's
##  economies Let know think
```

If we put together all these words in one whole text, we can see how they are related in that context.

```
cap_talk_context <- paste(cap_talk$pre, cap_talk$post, collapse = " ")
cap_talk_coll <- textstat_collocations(cap_talk_context, size = 3)
head(cap_talk_coll, 20)
```

```
##                   collocation count count_nested length   lambda         z
## 1      production means means     2            0      3 3.344771 1.3068288
## 2       economic needs people     2            0      3 3.278764 1.2789896
## 3             know people pay     2            0      3 2.825768 1.0993908
## 4    capitalist also actually     2            0      3 2.329066 1.0192045
## 5            capital can make     2            0      3 2.170820 0.9495177
## 6          needs instead just     2            0      3 2.359568 0.9151489
## 7               paid work get     2            0      3 2.343121 0.9094750
## 8  ignited protests worldwide     6            0      3 2.211441 0.8701963
## 9       labor capitalist mode     3            0      3 1.743161 0.7611189
## 10            rich labor rich     3            0      3 1.946246 0.7581333
## 11 ignited worldwide protests     8            0      3 1.794087 0.7034010
## 12       anti capitalist also     2            0      3 1.289242 0.6807563
## 13          labor nature work     2            0      3 1.540523 0.6678149
## 14      social economic needs     2            0      3 1.456625 0.6348281
## 15  instead just accumulation     2            0      3 1.620961 0.6242451
## 16          nature work social     2            0      3 1.339011 0.5800195
## 17   democracy pro democratic     2            0      3 1.378495 0.5324733
## 18       labor consuming make     3            0      3 1.358459 0.5208067
## 19        society poor people     2            0      3 1.038841 0.4635130
## 20            hey know people     2            0      3 1.197946 0.4550775
```

As a reference, the λ score is a measure of the times K specific consecutive tokens happen (in our second example, $K = 3$), given all the K consecutive tokens possibilities.

7 Analyzing a corpus: Features

The features of a corpus can give us clues about the characteristics of our text. At this point, we are already treating our documents as "bags-of-words". Having tokenized our corpus means that we are interested in each word, how often they appear, in conjunction with what other words, etc. We are going to creaate a

Document-Feature Matrix (*dfm*) object. I will going into detail of what is a *dfm*. For now, just think of it a dataset where each row if the count of words of each document.

```
dfm_toks <- dfm(cap_toks_stop)
```

Note that my *dfm* object contains all the document level variables I added to my corpus at the beginning.

7.1 (10) Most frequently used words

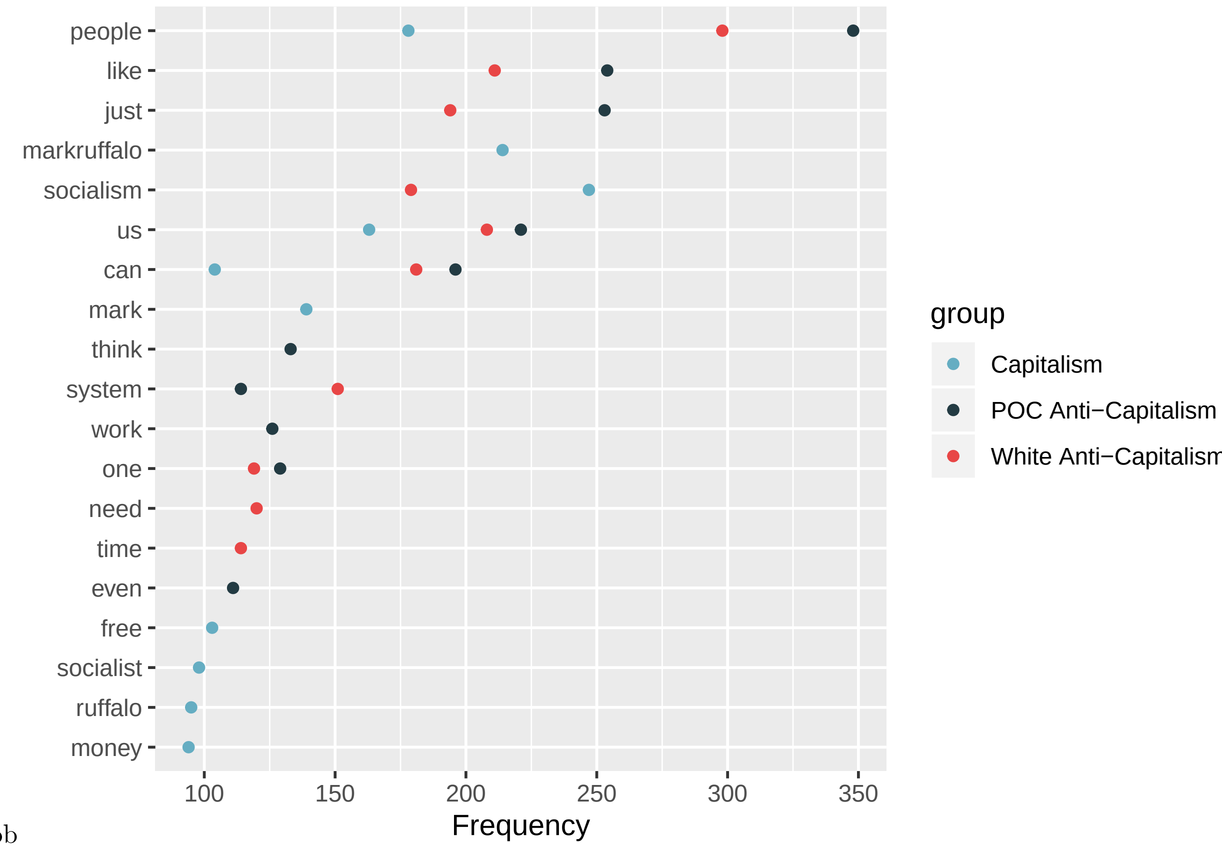
Simply:

```
cap_freq <- textstat_frequency(dfm_toks, n = 5, groups = "left_right")
head(cap_freq, 15)
```

```
##        feature frequency rank docfreq                group
## 1    socialism       247    1     217           Capitalism
## 2  markruffalo       214    2     206           Capitalism
## 3       people       178    3     156           Capitalism
## 4           us       163    4     138           Capitalism
## 5         mark       139    5     128           Capitalism
## 6       people       348    1     305   POC Anti-Capitalism
## 7         like       254    2     223   POC Anti-Capitalism
## 8         just       253    3     238   POC Anti-Capitalism
## 9           us       221    4     185   POC Anti-Capitalism
## 10         can       196    5     186   POC Anti-Capitalism
## 11      people       298    1     255 White Anti-Capitalism
## 12        like       211    2     182 White Anti-Capitalism
## 13          us       208    3     182 White Anti-Capitalism
## 14        just       194    4     178 White Anti-Capitalism
## 15         can       181    5     163 White Anti-Capitalism
```

Or a plot of the most frequently used words by group:

```
dfm_toks %>%
  textstat_frequency(n = 10,groups = "left_right") %>%
  ggplot(aes(x = reorder(feature, frequency), y = frequency, color = group)) +
  geom_point() +
  coord_flip() +
  labs(x = NULL, y = "Frequency")
```



Text Frequency-1.bb

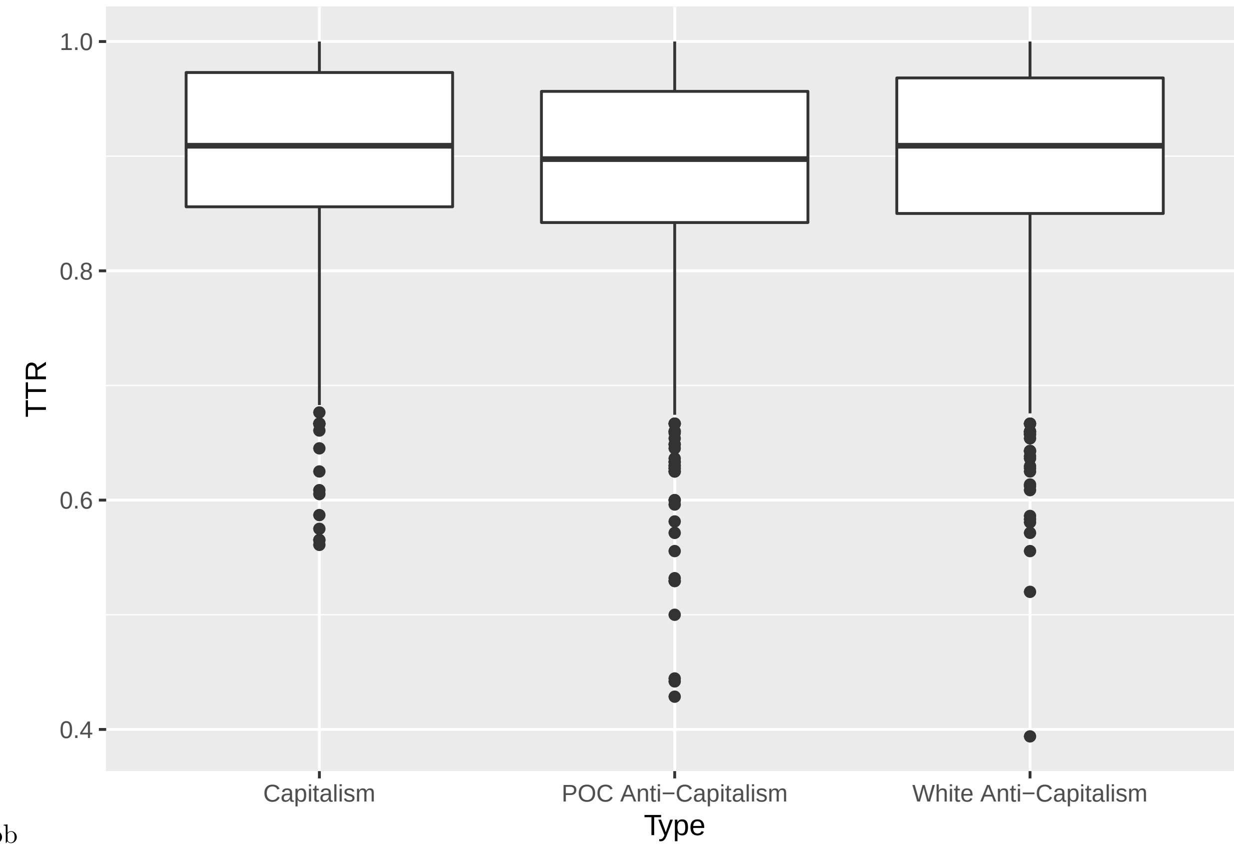
7.2 Lexical diversity

We might be interested in the breadth and variety of vocabulary used in a document. *Lexical diversity* is widely believed to be an important parameter to rate a document in terms of textual richness and effectiveness. Knowing Marxists, we might expect them to use more complex language than capitalists.

```
dfm_toks_2 <- dfm(cap_toks)

cap_lexdiv <- textstat_lexdiv(dfm_toks_2,measure = "TTR")
tail(cap_lexdiv, 5) # Does not tell us much, but...
```

```
##      document       TTR
## 5623 text5623 0.9473684
## 5624 text5624 0.8571429
## 5625 text5625 0.9729730
## 5626 text5626 0.7804878
## 5627 text5627 0.9743590
```

```
to_plot <- cbind.data.frame(data_capitalism, cap_lexdiv)

ggplot(to_plot, aes(x= factor(mem_name), y=TTR)) +
  geom_boxplot() +
    labs(x = "Type")
```



Diversity-1.bb

There are many measures of lexical diversity, each measuring something slightly different. `textstat_lexdiv()` includes many and you can check which adapts best to your needs here.

7.3 TF-IDF

Looking at simple frequencies might hide some important document features. As in everything in the social sciences, we can always complicate it a bit more. Enter TF-IDF: "Term-frequency / Inverse-document-frequency". TF-IDF weighting up-weights relatively rare words that do not appear in all documents. Using term frequency and inverse document frequency allows us to find words that are characteristic for one document within a collection of documents.

```r
head(dfm_toks[, 5:15])
```

```
## Document-feature matrix of: 6 documents, 11 features (83.3% sparse).
## 6 x 11 sparse Matrix of class "dfm"
##        features
## docs    websites use hate cronyism poster really pushes top beautiful fascism
##   text1        1   1    1        0      0      0      0   0         0       0
##   text2        0   0    0        1      1      1      1   1         1       0
##   text3        0   0    0        0      0      0      0   0         0       1
##   text4        0   0    0        0      0      0      0   0         0       0
##   text5        0   0    0        0      0      0      0   0         0       0
##   text6        0   0    0        0      0      0      0   0         0       0
##        features
## docs    decay
##   text1     0
##   text2     0
```

```
##   text3     1
##   text4     0
##   text5     0
##   text6     0
```

```r
head(dfm_tfidf(dfm_toks)[, 5:15])
```

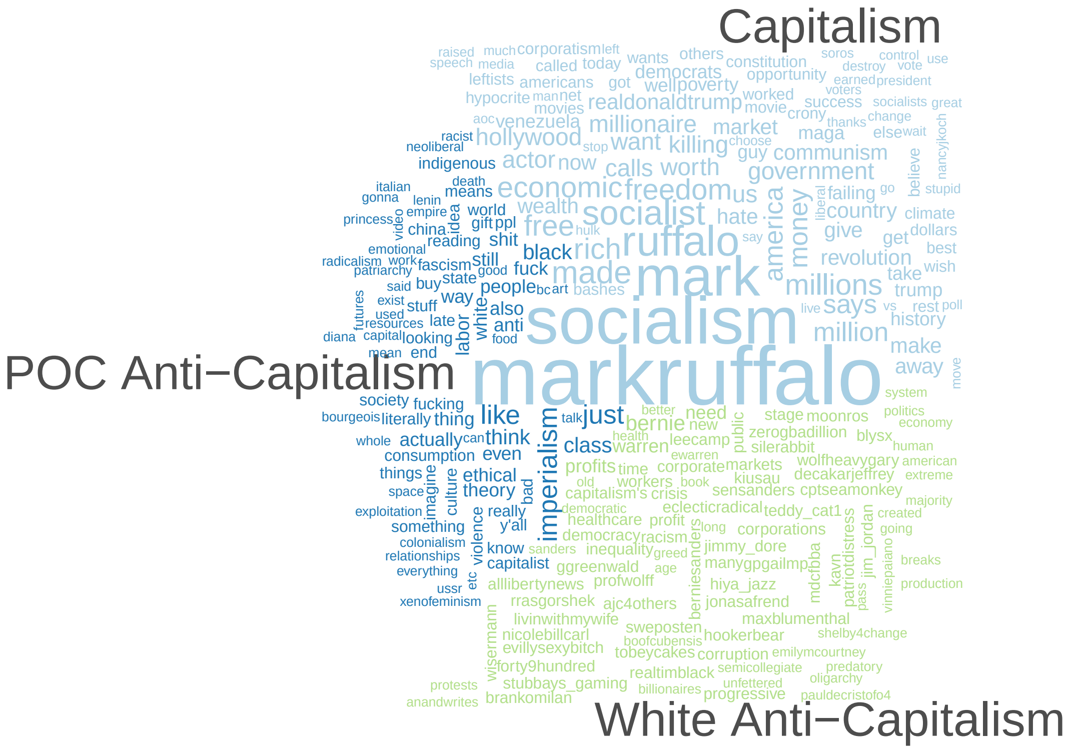
```
## Document-feature matrix of: 6 documents, 11 features (83.3% sparse).
## 6 x 11 sparse Matrix of class "dfm"
##        features
## docs     websites      use     hate cronyism   poster   really   pushes      top
##   text1 3.449247 1.869463 1.701059 0        0        0        0        0
##   text2 0        0        0        2.231763 2.495004 1.568433 3.273156 2.258915
##   text3 0        0        0        0        0        0        0        0
##   text4 0        0        0        0        0        0        0        0
##   text5 0        0        0        0        0        0        0        0
##   text6 0        0        0        0        0        0        0        0
##        features
## docs     beautiful  fascism    decay
##   text1  0        0        0
##   text2  2.847187 0        0
##   text3  0        1.858182 2.708884
##   text4  0        0        0
##   text5  0        0        0
##   text6  0        0        0
```

If we are building a dictionary, for example, we might want to include words with high TF-IDF values. Another way to think about TF-IDF is in terms of predictive power. Words that are common to all documents do not have any predictive power and receive a TD-IDF value of 0. Words that appear, but only in relatively few document, have greater predictive power and receive a TD-IDF > 0.

7.4 Wordclouds

Wordclouds are silly, but people seem to **love** them. Begrudgingly, I include the code:

```r
# In order to group the wordcloud I will create a new dfm with only one group:
dfm_toks_wordcloud <- dfm(cap_toks_stop,
                          groups = "left_right")

# comparison = T divides the word cloud into groups:
textplot_wordcloud(dfm_toks_wordcloud, comparison = T, max_words = 300)
```



8 Analyzing a corpus: Dictionary-based approaches

Dictionaries help us connect qualitative (concepts) and quantitative information extracted from text. Constructing a dictionary requires contextual interpretations. The **key** in a dictionary for text analysis (more of a thesaurus) is associated with non-exclusive terms (**values**):

- WC = wc, toilet, restroom, loo, bathroom
- vote = poll, suffrage, franchise, ballot, vote

Formally, there are three major categories:

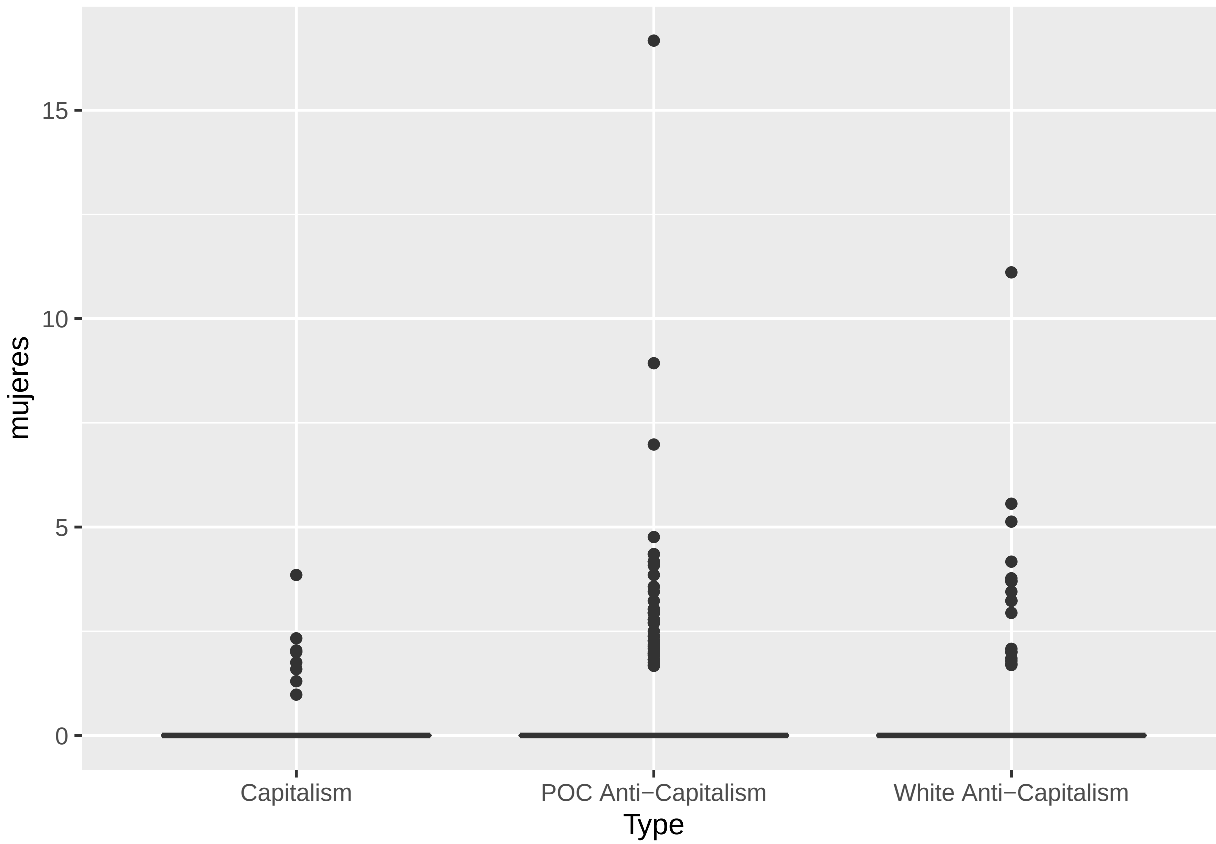
- Corpus-specific. For example, Pearson and Dancey (2011) use a dictionary category "women," which includes mentions of woman, women, woman's, women's, girl, girl's, girls, girls', female, female's, females, females', servicewoman, and servicewomen.
- General (e.g. LIWC)
- Sentiment Analysis

You can create your dictionary (using the `dictionary()` function), or you can use well-known dictionaries like: General Inquirer (Stone et al. 1996), Regressive Imagery Dictionary (Martindale, 1975, 1990), Linguistic Inquiry and Word Count, Laver and Garry (2000) to distinguish policy domains, Lexicoder Sentiment Dictionary (Young and Soroka, 2012). All dictionaries have drawbacks and may or may not adequately capture what you want them to capture. Remember: validate, validate, validate.

8.1 Corpus-specific dictionary

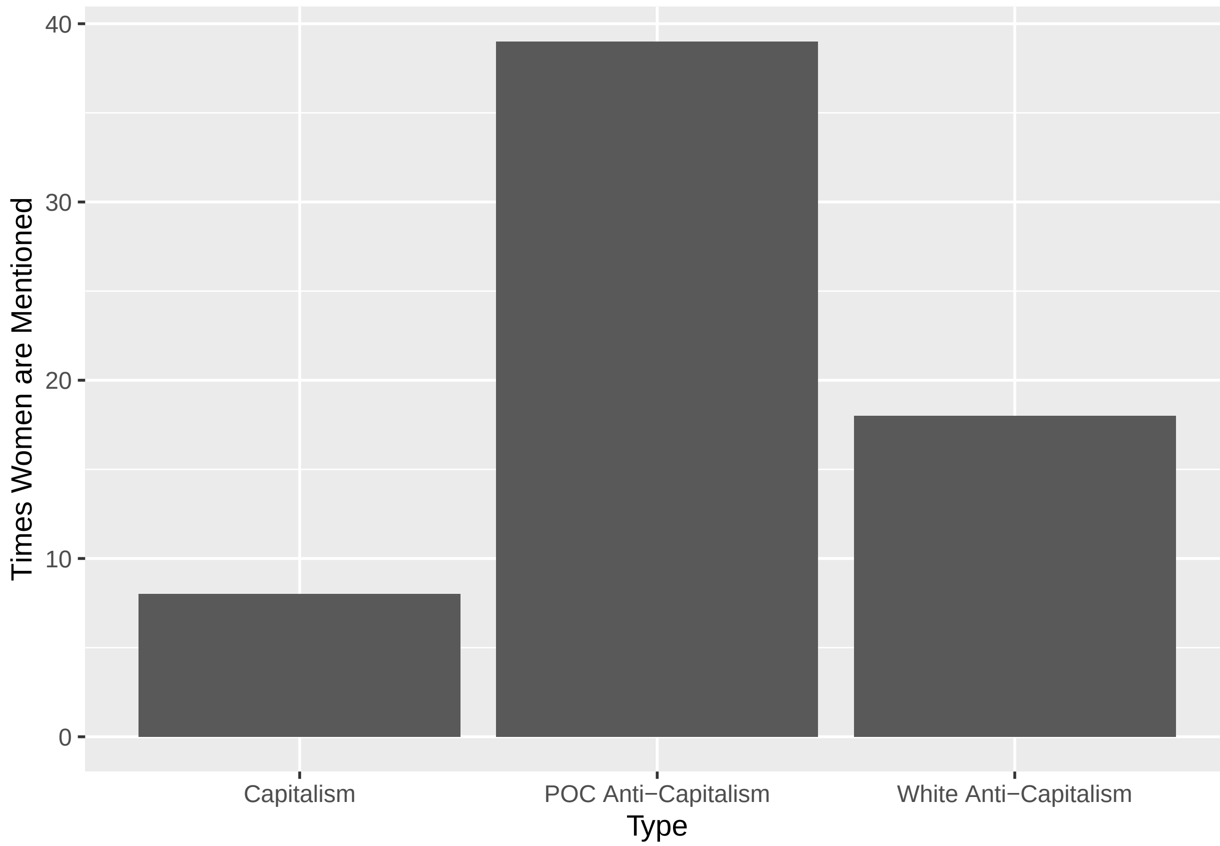
Let's apply the dictionary used by Pearson and Dancey (2011) to our corpus and see if there is any gender element to the way each side addresses capitalism.

```r
dict_women <- dictionary(list(mujeres = c("woman", "women", "girl*", "female*")))

cap_women <- liwcalike(data_capitalism$text_clean,
                              dictionary = dict_women)

to_plot <-  cbind.data.frame(cap_women,data_capitalism)
to_plot$women_dum <- 0
to_plot$women_dum[to_plot$mujeres>0] <- 1

ggplot(to_plot, aes(x=mem_name, y = mujeres)) +
    geom_boxplot() +
    labs(x = "Type")
```



I-1.bb

```r
ggplot(to_plot, aes(x=mem_name, y = women_dum)) +
    geom_bar(stat = "identity") +
    labs(x = "Type", y = "Times Women are Mentioned" )
```



I-2.bb

"The patriarchy is effing vast…"

8.2 LIWC and Sentiment Analysis

Let's do some simple "sentiment analysis" using two dictionaries: GI and NRC.

```
cap_sentimentNRC <- liwcalike(cap_corp,
                              dictionary = data_dictionary_NRC)

cap_sentimentGI <- liwcalike(cap_corp,
                             dictionary = data_dictionary_geninqposneg)

head(cap_sentimentNRC, 15)
```

```
##    docname Segment WC       WPS Sixltr   Dic anger anticipation disgust  fear
## 1    text1       1 23 23.000000  21.74 30.43  4.35         0.00    4.35  4.35
## 2    text2       2 14  5.500000  21.43 35.71  0.00         7.14    0.00  0.00
## 3    text3       3 53 22.500000  16.98 37.74  1.89         3.77    1.89  3.77
## 4    text4       4  8  7.000000  12.50  0.00  0.00         0.00    0.00  0.00
## 5    text5       5 51  9.250000  27.45 41.18  9.80         0.00    5.88  5.88
## 6    text6       6 10  9.000000  30.00  0.00  0.00         0.00    0.00  0.00
## 7    text7       7 55 14.333333  20.00  7.27  0.00         0.00    0.00  0.00
## 8    text8       8 29 12.000000  34.48 55.17 10.34         0.00   13.79  6.90
## 9    text9       9  7  6.000000  14.29  0.00  0.00         0.00    0.00  0.00
## 10  text10      10 20 19.000000  15.00  5.00  0.00         0.00    0.00  0.00
## 11  text11      11 16  7.000000  18.75 25.00  0.00         6.25    0.00  0.00
## 12  text12      12 52  9.000000  23.08 19.23  1.92         0.00    0.00  3.85
```

```
## 13  text13      13 28 12.500000  25.00 25.00  0.00         0.00    0.00  7.14
## 14  text14      14 19  5.333333  31.58 68.42 10.53         5.26    5.26 10.53
## 15  text15      15 48 12.000000  33.33 20.83  0.00         0.00    2.08  4.17
##     joy negative positive sadness surprise trust AllPunc Period Comma Colon
## 1  0.00     4.35     4.35    4.35     0.00  4.35    4.35   0.00  0.00  0.00
## 2  7.14     0.00    14.29    0.00     0.00  7.14   21.43   7.14  0.00  0.00
## 3  5.66     5.66     5.66    3.77     1.89  3.77   20.75   3.77  3.77  0.00
## 4  0.00     0.00     0.00    0.00     0.00  0.00   12.50   0.00 12.50  0.00
## 5  0.00    13.73     0.00    5.88     0.00  0.00   33.33   7.84  7.84  5.88
## 6  0.00     0.00     0.00    0.00     0.00  0.00   10.00   0.00 10.00  0.00
## 7  1.82     0.00     5.45    0.00     0.00  0.00   30.91   5.45  3.64  0.00
## 8  0.00    13.79     0.00   10.34     0.00  0.00   17.24   6.90 10.34  0.00
## 9  0.00     0.00     0.00    0.00     0.00  0.00   14.29   0.00  0.00  0.00
## 10 0.00     0.00     5.00    0.00     0.00  0.00    5.00   0.00  5.00  0.00
## 11 0.00     6.25     6.25    6.25     0.00  0.00   18.75   6.25  0.00  0.00
## 12 0.00     5.77     3.85    3.85     0.00  0.00   25.00   5.77  0.00  0.00
## 13 0.00     3.57     7.14    0.00     0.00  7.14   14.29   3.57  0.00  7.14
## 14 5.26    10.53     5.26    5.26     5.26  5.26   15.79  15.79  0.00  0.00
## 15 2.08     4.17     4.17    2.08     0.00  2.08   31.25  10.42  6.25  0.00
##    SemiC QMark Exclam Dash Quote Apostro Parenth OtherP
## 1   0.00  0.00   0.00 0.00  4.35    4.35       0   4.35
## 2   0.00  0.00   0.00 0.00 14.29    0.00       0  21.43
## 3   0.00  0.00   0.00 0.00 13.21    5.66       0  20.75
## 4   0.00  0.00   0.00 0.00  0.00    0.00       0  12.50
## 5   3.92  0.00   0.00 1.96  3.92    3.92       0  31.37
## 6   0.00  0.00   0.00 0.00  0.00    0.00       0  10.00
## 7   3.64  0.00   1.82 0.00  3.64    0.00       0  30.91
## 8   0.00  0.00   0.00 0.00  0.00    0.00       0  17.24
## 9   0.00 14.29   0.00 0.00  0.00    0.00       0  14.29
## 10  0.00  0.00   0.00 0.00  0.00    0.00       0   5.00
## 11  0.00  0.00   0.00 0.00  6.25    6.25       0  18.75
## 12  0.00  1.92   0.00 0.00  1.92    1.92       0  17.31
## 13  0.00  0.00   0.00 0.00  0.00    0.00       0  14.29
## 14  0.00  0.00   0.00 0.00  0.00    0.00       0  15.79
## 15  2.08  0.00   0.00 0.00  4.17    0.00       0  31.25
```

```
head(cap_sentimentGI, 15)
```

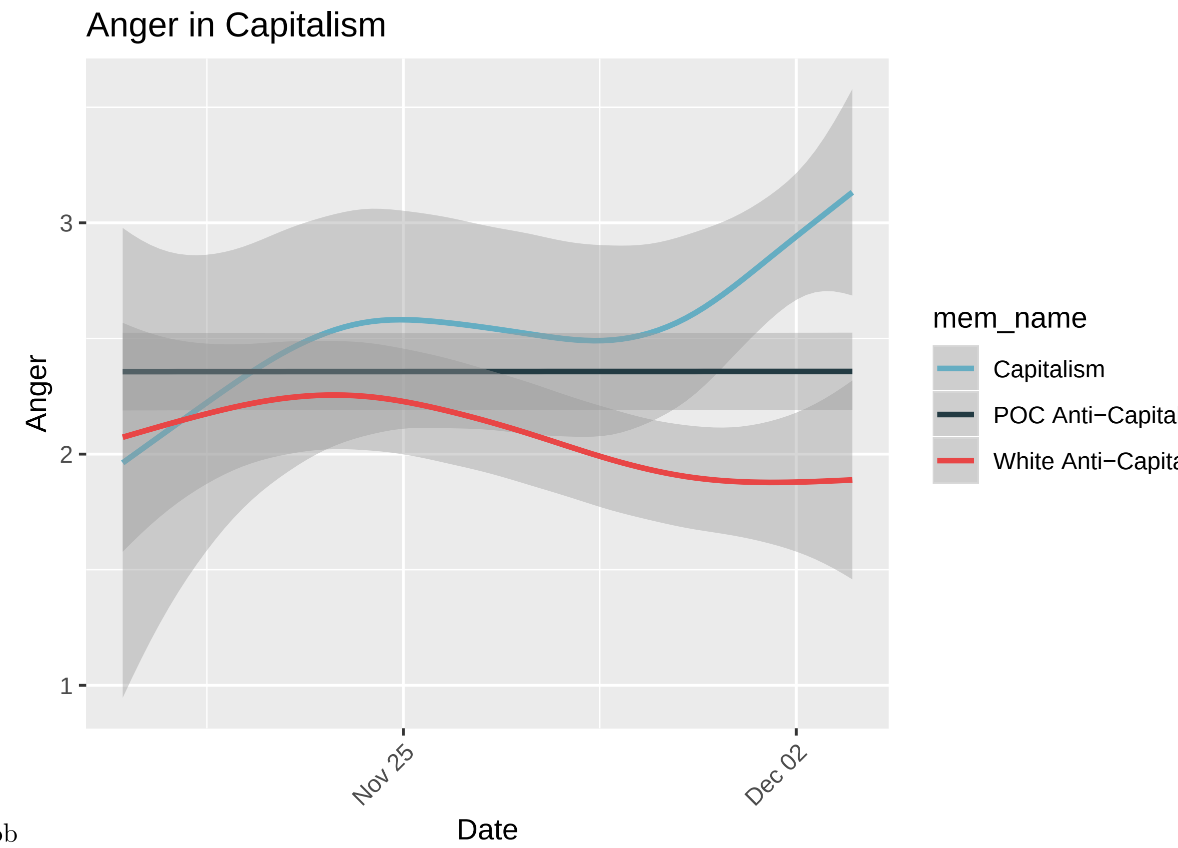
```
##    docname Segment WC       WPS Sixltr   Dic positive negative AllPunc Period
## 1    text1       1 23 23.000000  21.74 13.04     4.35     8.70    4.35   0.00
## 2    text2       2 14  5.500000  21.43  0.00     0.00     0.00   21.43   7.14
## 3    text3       3 53 22.500000  16.98 15.09     7.55     7.55   20.75   3.77
## 4    text4       4  8  7.000000  12.50 12.50     0.00    12.50   12.50   0.00
## 5    text5       5 51  9.250000  27.45 13.73     1.96    11.76   33.33   7.84
## 6    text6       6 10  9.000000  30.00 10.00    10.00     0.00   10.00   0.00
## 7    text7       7 55 14.333333  20.00  7.27     7.27     0.00   30.91   5.45
## 8    text8       8 29 12.000000  34.48 10.34     3.45     6.90   17.24   6.90
## 9    text9       9  7  6.000000  14.29  0.00     0.00     0.00   14.29   0.00
## 10  text10      10 20 19.000000  15.00 30.00    15.00    15.00    5.00   0.00
## 11  text11      11 16  7.000000  18.75  6.25     0.00     6.25   18.75   6.25
## 12  text12      12 52  9.000000  23.08 13.46     5.77     7.69   25.00   5.77
## 13  text13      13 28 12.500000  25.00  3.57     3.57     0.00   14.29   3.57
## 14  text14      14 19  5.333333  31.58  5.26     5.26     0.00   15.79  15.79
## 15  text15      15 48 12.000000  33.33  6.25     2.08     4.17   31.25  10.42
##    Comma Colon SemiC QMark Exclam Dash Quote Apostro Parenth OtherP
```

```
## 1   0.00  0.00  0.00  0.00   0.00 0.00  4.35   4.35     0   4.35
## 2   0.00  0.00  0.00  0.00   0.00 0.00 14.29   0.00     0  21.43
## 3   3.77  0.00  0.00  0.00   0.00 0.00 13.21   5.66     0  20.75
## 4  12.50  0.00  0.00  0.00   0.00 0.00  0.00   0.00     0  12.50
## 5   7.84  5.88  3.92  0.00   0.00 1.96  3.92   3.92     0  31.37
## 6  10.00  0.00  0.00  0.00   0.00 0.00  0.00   0.00     0  10.00
## 7   3.64  0.00  3.64  0.00   1.82 0.00  3.64   0.00     0  30.91
## 8  10.34  0.00  0.00  0.00   0.00 0.00  0.00   0.00     0  17.24
## 9   0.00  0.00  0.00 14.29   0.00 0.00  0.00   0.00     0  14.29
## 10  5.00  0.00  0.00  0.00   0.00 0.00  0.00   0.00     0   5.00
## 11  0.00  0.00  0.00  0.00   0.00 0.00  6.25   6.25     0  18.75
## 12  0.00  0.00  0.00  1.92   0.00 0.00  1.92   1.92     0  17.31
## 13  0.00  7.14  0.00  0.00   0.00 0.00  0.00   0.00     0  14.29
## 14  0.00  0.00  0.00  0.00   0.00 0.00  0.00   0.00     0  15.79
## 15  6.25  0.00  2.08  0.00   0.00 0.00  4.17   0.00     0  31.25
```

Both produce dataframe objects that can later be manipulated to conduct more in-depth analysis. Let's see how "anger" and "joy" language varies across groups and time using the NRC dictionary.

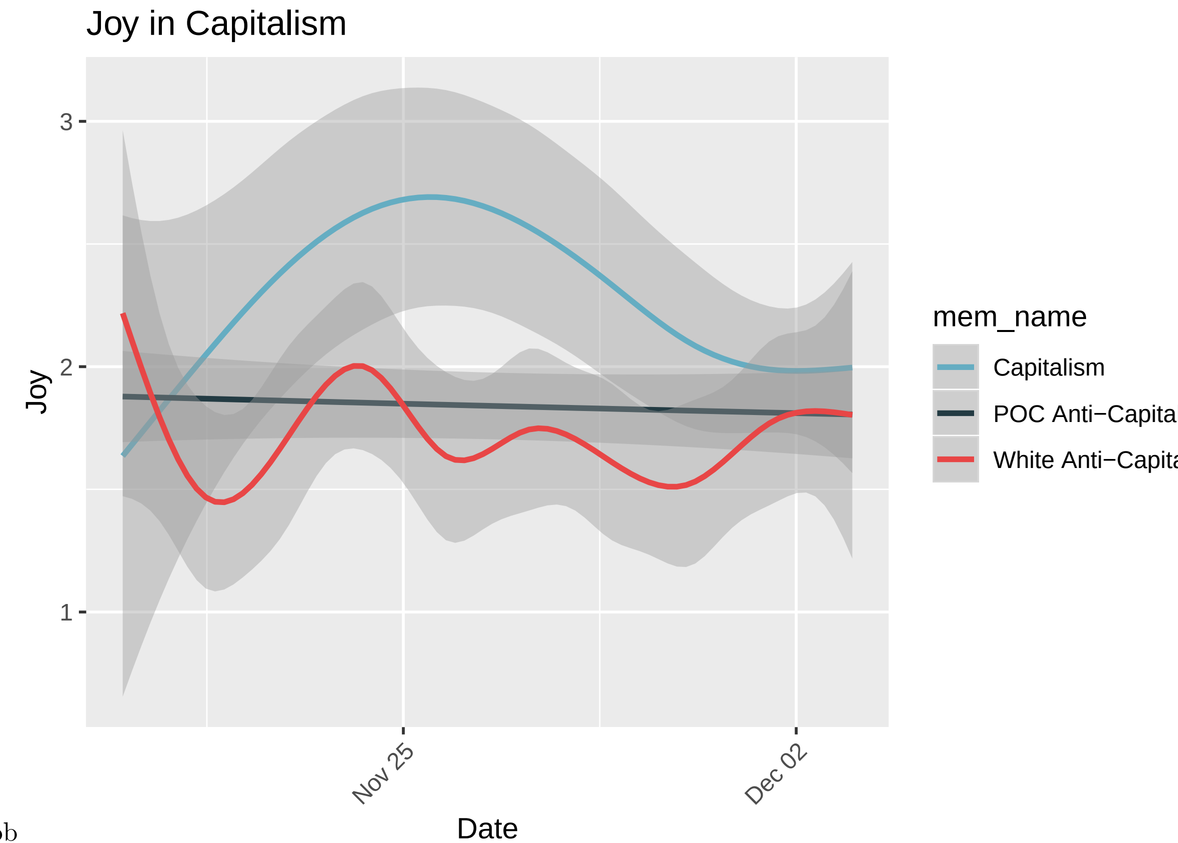
```r
cap_sentiment_df <- cbind.data.frame(cap_sentimentNRC,data_capitalism)
cap_sentiment_df <- cap_sentiment_df[cap_sentiment_df$date_created > "2019-11-19",]

ggplot(cap_sentiment_df, aes(x=date_created, y=anger, color = mem_name))+
  stat_smooth() +
  labs(title="Anger in Capitalism", x="Date", y = "Anger", las=2)+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```r
ggplot(cap_sentiment_df, aes(x=date_created, y=joy, color = mem_name))+
  stat_smooth() +
  labs(title="Joy in Capitalism", x="Date", y = "Joy", las=2)+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Sentiment Analysis-2.bb

One of the limitations of sentiment analysis is the variation we get on similar sentiments when using different dictionaries. For example, analyzing *polarity* in our data (positive language minus negative language), we obtain different results from GI and NRC.

```r
polarity_NRC <- cap_sentimentNRC$positive -
cap_sentimentNRC$negative

polarity_GI <- cap_sentimentGI$positive -
cap_sentimentGI$negative

cor(polarity_NRC,polarity_GI)
```

```
## [1] 0.5070789
```

As a general recommendation, it is always good practice to explore the dictionary we are going to use before actually using it. This is particularly important when working with text in another language. Most dictionaries (commercial or otherwise) that have been evaluated and validated, have been tested and evaluated in English. If using another language, it might be a good idea to check how words are classified. Another common practice is to use Google Translate API or Microsoft Translator API on the original corpus. This can be costly depending on the size of the corpus and the results, like with any other text-analysis tool, need to be constantly validated (Lucas et al. 2015).

Table 1: Matrix D

	W_1	W_2	W_3	W_m
D_1	3	4	0	1
D_2	0	3	0	4
D_3	1	2	2	0
D_m	1	0	2	3

Figure 2: Matrix D

9 Analyzing a corpus: Topic models and document distance

Topics models were primarily developed to summarize unstructured text, use words within documents to infer their topic, and as a form of dimension reduction. It allows social scientists to use topics as a form of measurement, as we are often interested in how observed covariates drive trends in language. For example,

- Political attention patterns in Senate floor speeches (Quinn et al. 2010)
- Attention senators allocate to press releases (Grimmer 2010).
- Climate change "skepticism" in reports and communications by think tanks and interest groups (Bousaills and Coan 2016).

Topic models are "unsupervised" methods. As such, they require a great deal of human supervision, especially when it comes to validating the results.

9.1 Document-term Matrix

Topic models are a broad class of Bayesian generative models that encode problem-specific structure into an estimation of categories (Grimmer and Stewart 2013; Blei et al. 2010). Statistically, a topic is a probability mass function over words. The idea of topic models is that each document exhibits a topic in some proportion: each document is a distribution over topics, and each topic is a distribution over words.

We can take our corpus and turn it into a matrix that reflects this concept. We call it a document-term matrix (*dtm*).

- A corpus of n documents D_1, D_2, D_3 ... D_n
- Vocabulary of m words W_1, W_2, W_3 ... W_m
- The value of i, j cell gives the frequency count of word W_j in document D_i

Latent Dirichlet Allocation (LDA) converts the *dtm* into two lower-dimension matrices, M_1 and M_2:

- M_1 is a NxK document-topic matrix
- M_2 is a KxM topic-term matrix

Table 2: Matrix M_1

	K_1	K_2	K_3	K_p
D_1	1	0	0	1
D_2	1	1	0	0
D_3	1	0	1	1
D_m	1	0	0	0

Figure 3: Matrix M1

Table 3: Matrix M_2

	W_1	W_2	W_3	W_m
K_1	1	0	1	1
K_2	0	1	0	0
K_3	1	1	1	0
K_p	1	0	0	0

Figure 4: Matrix M2

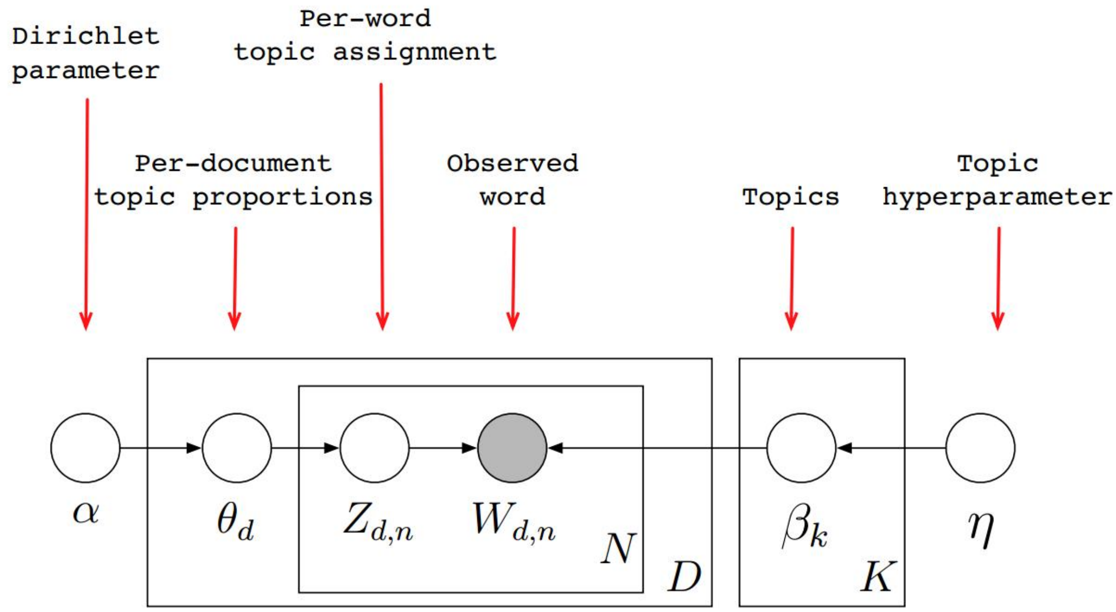


Figure 5: Plate notation of Latent Dirichlet Allocation

LDA estimates the distribution over words for each topic, and the proportion of a topic in each document. You can read about the math behind the two-step process in Grimmer and Stewart (2013).

- α –> document-topic density: higher α means documents contain more topics, lower α means a document contains fewer topics.
- β –> topic-word density. higher β means topics have more words, lower β means topics have fewer words.
- Number of topics –> this is specified in advance, or can be chosen to optimize model fit (we will get back to this point). The "statistically optimal" topic count is usually too high for the topics to be interpretable or useful.

9.2 Structural Topic Models (STM)

To apply (and visualize) LDA, we are going to be using an extension of LDA known as Structural Topic Models (STM).

$$STM = LDA + Metadata$$

STM provides two ways to include contextual information to "guide" the estimation of the model. First, topic prevalence can vary by metadata (e.g. Republicans talk about military issues more than Democrats). Second, topic content can vary by metadata (e.g. Republicans talk about military issues differently from Democrats).

We can run STM using the `stm` package. The `stm` package includes the complete workflow (i.e. from raw text to figures), and if you are planning to use it in the future I highly encourage you to check this and this. `stm()` takes our *dfm* and produces topics. If we do not specify any prevalence terms, then it will estimate an LDA. Since this is a Bayesian approach, it is recommended you set a seed value for future replication. We also need to set K number of topics. How many topics are the right number of topics? There is no good number. Too many pre-specified topics and the categories might be meaningless. Too few, and you might be piling together two or more topics. Note that changes to a) the number of topics, b) the prevalence term, c) the omitted words, d) the seed value, can (greatly) change the outcome. Here is where validation becomes crucial (for a review see Wilkerson and Casas 2017).

Using our dataset, I will use stm to estimate the topics surrounding "capitalism" on Twitter. As my prevalence term, I add the position of each authority. I set my number of topics at 10 (but with a corpus this big I should probably set it at ~30 and work my way up from there).

```
dfm_toks_stem <- dfm(cap_toks_stem)

# After trimming, some documents were left with no tokens so I will eliminate those before running my m
dfm_toks_sub <- dfm_subset(dfm_toks_stem, ntoken(dfm_toks_stem) > 0)

cap_TM <- stm(dfm_toks_sub, K = 10, seed = 1984,
              prevalence = ~left_right,
              init.type = "Spectral")
```

The nice thing about the `stm()` function is that it allows us to see in "real-time" what is going on within the black box. We can summarize the process in the following way (this is similar to a collapsed Gibbs sampling, which the `stm()` function sort of uses):

1. Go through each document, and randomly assign each word in the document to one of the topics $t \in k$.
2. Notice that this random assignment already gives both topic representations of all the documents and word distributions of all the topics (albeit not very good ones).
3. So to improve on them, for each document W do the following: 3.1 Go through each word w in W 3.1.1 And for each topic t, compute two things: 3.1.1.1 $p(t|W)$ = the proportion of words in document W that are currently assigned to topic t, and 3.1.1.2 $p(w|t)$ = the proportion of assignments to topic t over all documents that come from this word w.

   Reassign w a new topic, where we choose topic t with probability $p(t|W) * p(w|t)$. It is worth noting that according to our generative model, this is essentially the probability that topic t generated word w, so it makes sense that we resample the current word's topic with this probability. (Also, I'm glossing over a couple of things here, in particular the use of priors/pseudocounts in these probabilities.)

   3.1.1.3 In other words, in this step, we're assuming that all topic assignments except for the current word in question are correct, and then updating the assignment of the current word using our model of how documents are generated.
4. After repeating the previous step a large number of times, you'll eventually reach a roughly steady state where your assignments are pretty good. So use these assignments to estimate the topic mixtures of each document (by counting the proportion of words assigned to each topic within that document) and the words associated with each topic (by counting the proportion of words assigned to each topic overall).

(This explanation was taken from here). Let's explore the topics produced:

```
labelTopics(cap_TM)
```

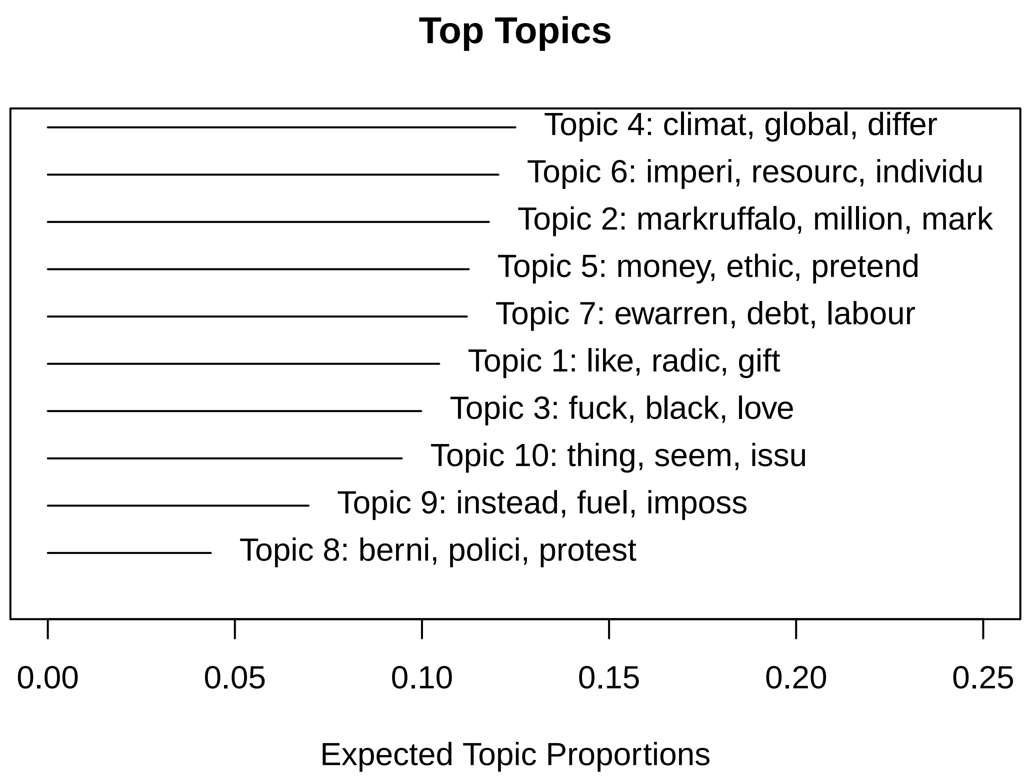
```
## Topic 1 Top Words:
##       Highest Prob: peopl, like, good, end, never, look, realli
##       FREX: like, radic, gift, art, vegan, digit, empire
##       Lift: 80s, aral, chichisup, coffin, dutch, etcetera, feet
##       Score: like, peopl, gift, look, good, diana, princess
## Topic 2 Top Words:
##       Highest Prob: social, markruffalo, rich, million, kill, mark, socialist
##       FREX: markruffalo, million, mark, ruffalo, worth, millionair, hollywood
##       Lift: deepstat, 30m, breitbartnew, bridgetphetasi, clue, cronyism, darkwatersmovi
```

```
##       Score: markruffalo, mark, ruffalo, million, kill, worth, social
## Topic 3 Top Words:
##       Highest Prob: just, say, fuck, know, problem, much, american
##       FREX: fuck, black, love, leftist, friday, u, sex
##       Lift: asshol, bent, blk, bourgeois, chart, custom, dick
##       Score: fuck, just, hate, black, friday, love, problem
## Topic 4 Top Words:
##       Highest Prob: social, market, free, state, anti, climat, chang
##       FREX: climat, global, differ, u.s, competit, challeng, threat
##       Lift:   , 150th, 17th, 1990s, 1a2a, 4human, 4sale
##       Score: market, state, climat, free, social, global, stage
## Topic 5 Top Words:
##       Highest Prob: get, money, go, now, give, everyon, busi
##       FREX: money, ethic, pretend, speak, disgust, get, sleep
##       Lift: agreed, artworkte, barackobama, ben, bigot, blatant, brianstelt
##       Score: get, money, go, give, ethic, everyon, bad
## Topic 6 Top Words:
##       Highest Prob: system, need, human, world, profit, live, exploit
##       FREX: imperi, resourc, individu, relationship, growth, longer, altern
##       Lift: agricultur, anarchist_black, arab, autonomi, bake, bolivian, cathol
##       Score: system, human, profit, resourc, exploit, product, natur
## Topic 7 Top Words:
##       Highest Prob: work, can, us, worker, time, class, better
##       FREX: ewarren, debt, labour, insur, hour, corrupt, care
##       Lift: catastrophe, davi, ewarren, insecur, mitig, patient, unafford
##       Score: us, work, can, corrupt, pay, warren, poor
## Topic 8 Top Words:
##       Highest Prob: freedom, berni, public, polici, protest, capit, failur
##       FREX: berni, polici, protest, failur, sander, sensand, leecamp
##       Lift: pete, _michelangelo__, ( ,   ,   , 11warrior, 13_moth
##       Score: zerogbadillion, jimmy_dor, ajc4other, alllibertynew, blysx, cptseamonkey, decakarjeffre
## Topic 9 Top Words:
##       Highest Prob: make, think, right, even, labor, part, instead
##       FREX: instead, fuel, imposs, consequ, content, vision, cheap
##       Lift:   ,   ,   ,   , analyz, beckert, behaviour
##       Score: make, think, labor, even, right, instead, el
## Topic 10 Top Words:
##       Highest Prob: thing, way, also, creat, power, societi, great
##       FREX: thing, seem, issu, anandwrit, hierarchi, essay, racism
##       Lift: acceleration, ambit, assert, blackston, calebmaupin, copyright, demis
##       Score: thing, also, creat, racism, societi, way, power
```

FREX weights words by their overall frequency and how exclusive they are to the topic. Lift weights words by dividing by their frequency in other topics, therefore giving higher weight to words that appear less frequently in other topics. Similar to lift, score divides the log frequency of the word in the topic by the log frequency of the word in other topics (Roberts et al. 2013). Bischof and Airoldi (2012) show the value of using FREX over the other measures.

You can use the `plot()` function to show the topics.

```r
plot(cap_TM, type = "summary", labeltype = "frex") # or prob, lift score
```



Topics-1.bb

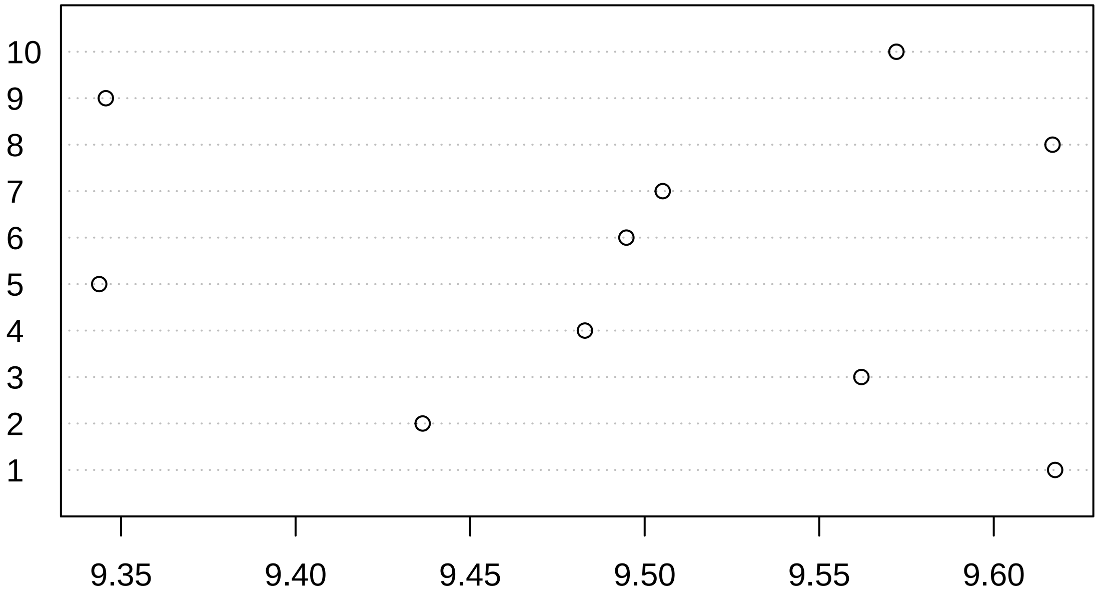
If you want to see a sample of a specific topic:

```
findThoughts(cap_TM, texts = texts(cap_corp)[docnames(dfm_toks_sub)], topics = 4)
```

```
##
##  Topic 4:
##       "Neoliberalism is the 20th-century resurgence of 19th-century ideas associated with laissez-fa
##
## Not that vague Nate
##      @LuckyHeronSay @LibDems The LibD`s,Tories &amp; anybody who supports capitalism &amp; the cuts
## Only full blooded socialism,the truth--&amp; an IMPLEMENTED socialist alt. will suffice 2 beat them.
## Blairites have IMPLEMENTED Tory cuts etc.
## Lab could be in trouble in their areas.
##      @PHPatriot_1 @zoothorn69 The connection between #Christianity, #capitalism, our #Constitution,
```

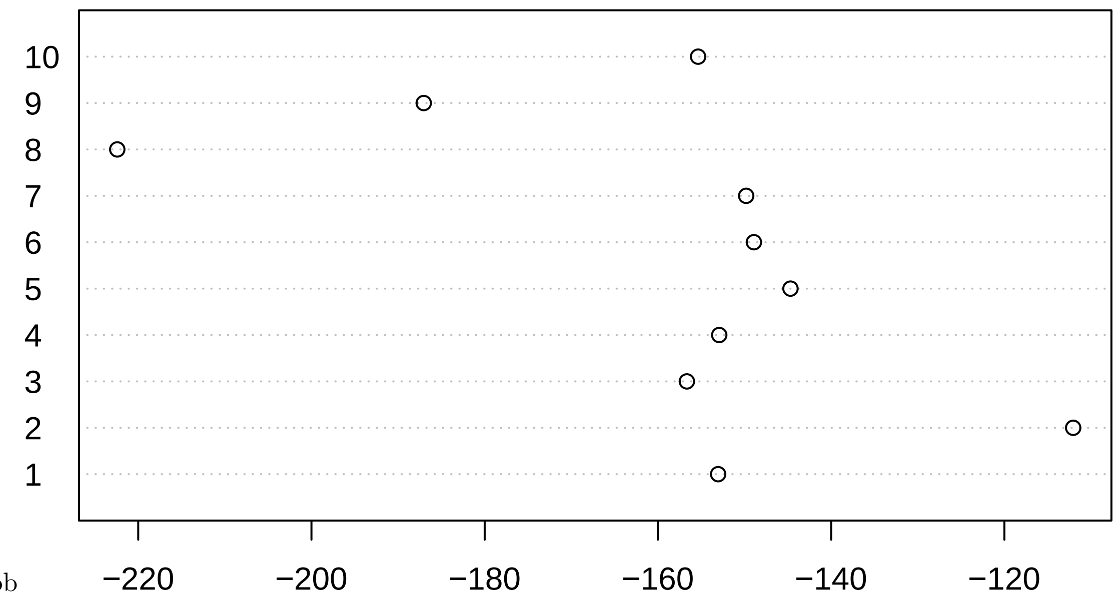
We can (should/must) run some diagnostics. There are two qualities that were are looking for in our model: semantic coherence and exclusivity. Exclusivity is base on the FREX labeling metrix. Semantic coherence is a creterion developed by Mimno et al. (2011) and it maximizes when the most probable words in a given topic frequently co-occur together. Mimno et al. (2011) show that the metric correlates well with human judgement of topic quality. Yet, it is fairly easy to obtain high semantic coherence so it is important to see it in tandem with exclusivity. Let's see how exclusive are the words in each topic:

```
dotchart(exclusivity(cap_TM), labels = 1:10)
```



We can also see the semantic coherence of our topics –words a topic generates should co-occur often in the same document–:

```
dotchart(semanticCoherence(cap_TM,dfm_toks_sub), labels = 1:10)
```

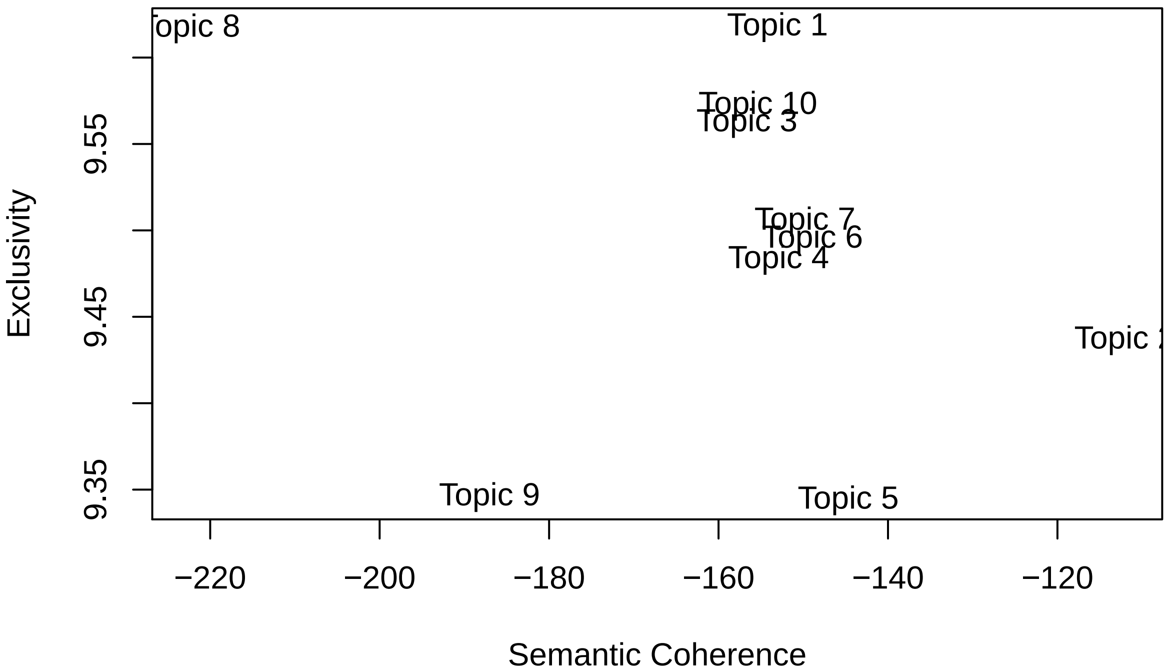


Coherence-1.bb

We can also see the overall quality of our topic model:

```
topicQuality(cap_TM,dfm_toks_sub)
```

```
## [1] -153.0452 -112.0533 -156.6526 -152.9175 -144.6928 -148.9168 -149.8053
## [8] -222.4208 -187.0373 -155.3585
## [1] 9.617573 9.436403 9.562088 9.482883 9.343743 9.494762 9.505163 9.616828
## [9] 9.345646 9.572110
```



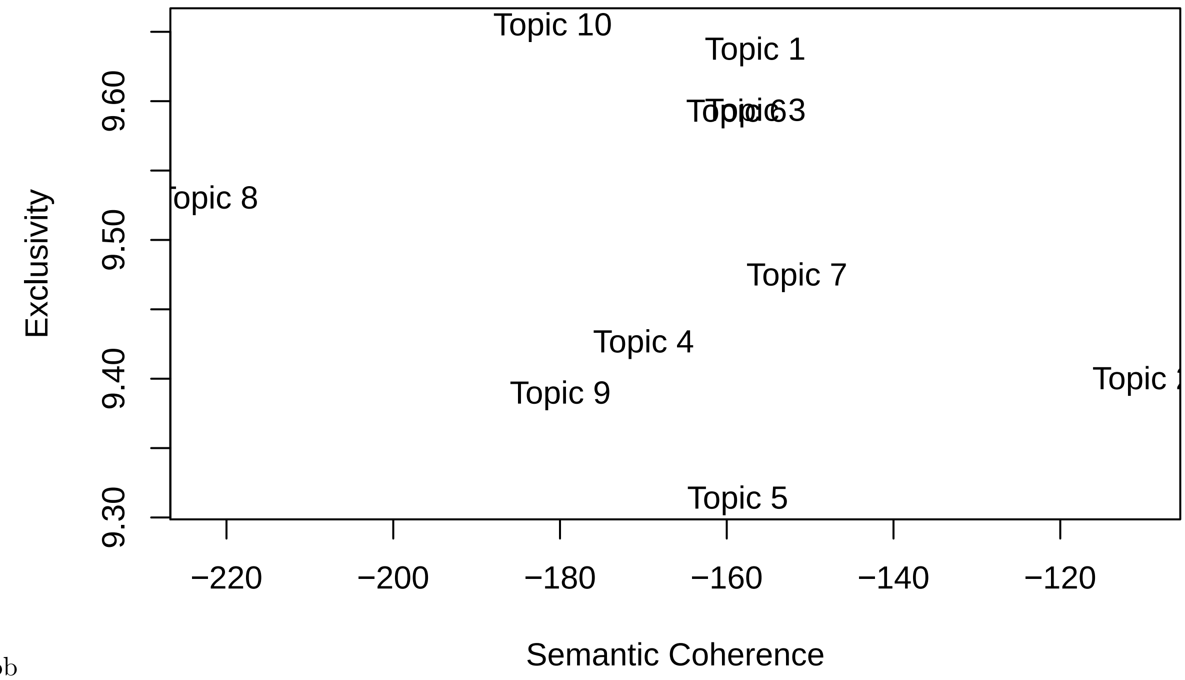
On their own, both metrics are not really useful (what do those numbers even mean?). They are useful when we are looking for the "optimal" number of topics.

```
cap_TM_0 <- manyTopics(dfm_toks_sub,
                       prevalence = ~ left_right,
                       K = c(10,15,20), runs=2,
                       max.em.its = 50,
                       init.type = "Spectral") # It takes around 250 iterations for the model to conver
```

We can now compare the perfomance of each model based on their semantic coherence and exclusivity:

```
k_10 <- cap_TM_0$out[[1]] # k_10 is an stm object which can be explored and used like any other topic m
k_15 <- cap_TM_0$out[[2]]
k_20 <- cap_TM_0$out[[3]]

# I will just graph the 'quality' of each model:
topicQuality(k_10,dfm_toks_sub)
```

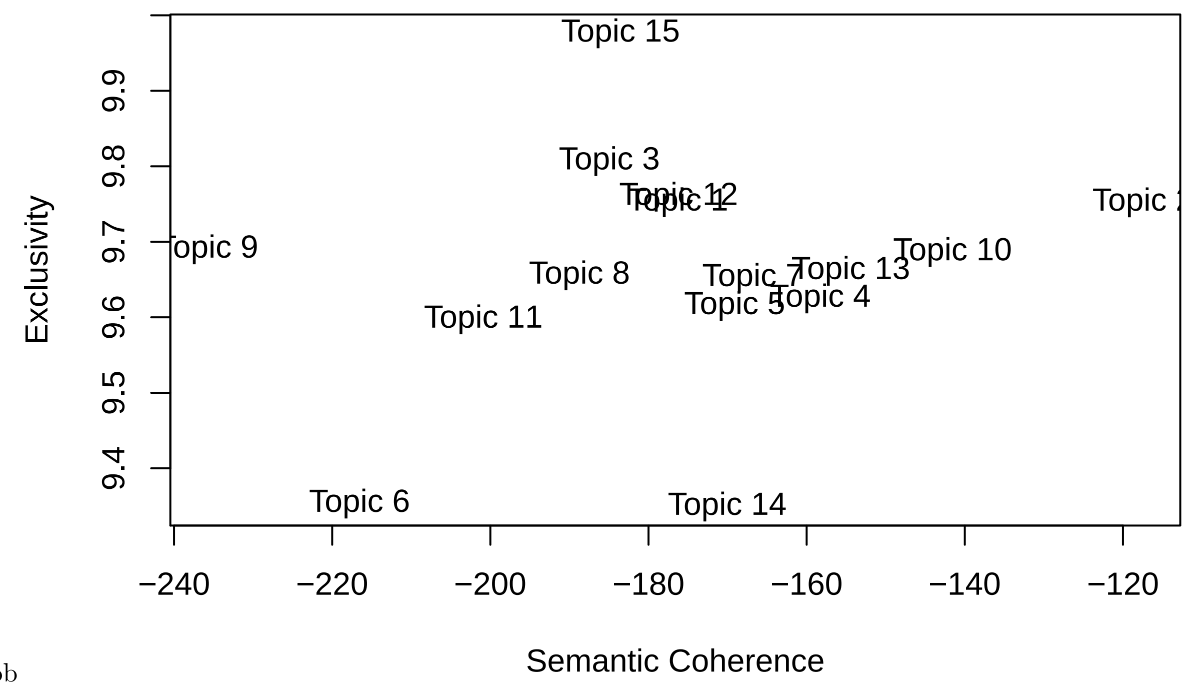
```
##  [1] -156.5871 -110.0929 -156.5661 -169.9799 -158.6913 -158.8270 -151.5906
##  [8] -222.2478 -179.9702 -180.8846
##  [1] 9.635785 9.398349 9.591756 9.424861 9.312281 9.591102 9.473077 9.528538
##  [9] 9.387953 9.653311
```



by K-1.bb

```
topicQuality(k_15,dfm_toks_sub)
```

```
##  [1] -176.3001 -117.4834 -184.9537 -158.2778 -169.1092 -216.5443 -166.8171
##  [8] -188.7352 -235.7356 -141.5557 -200.8848 -176.1947 -154.4283 -170.0446
## [15] -183.5428
##  [1] 9.752277 9.752145 9.806760 9.624878 9.615049 9.353169 9.652194 9.655456
##  [9] 9.689886 9.686355 9.597133 9.759681 9.661525 9.349216 9.976027
```

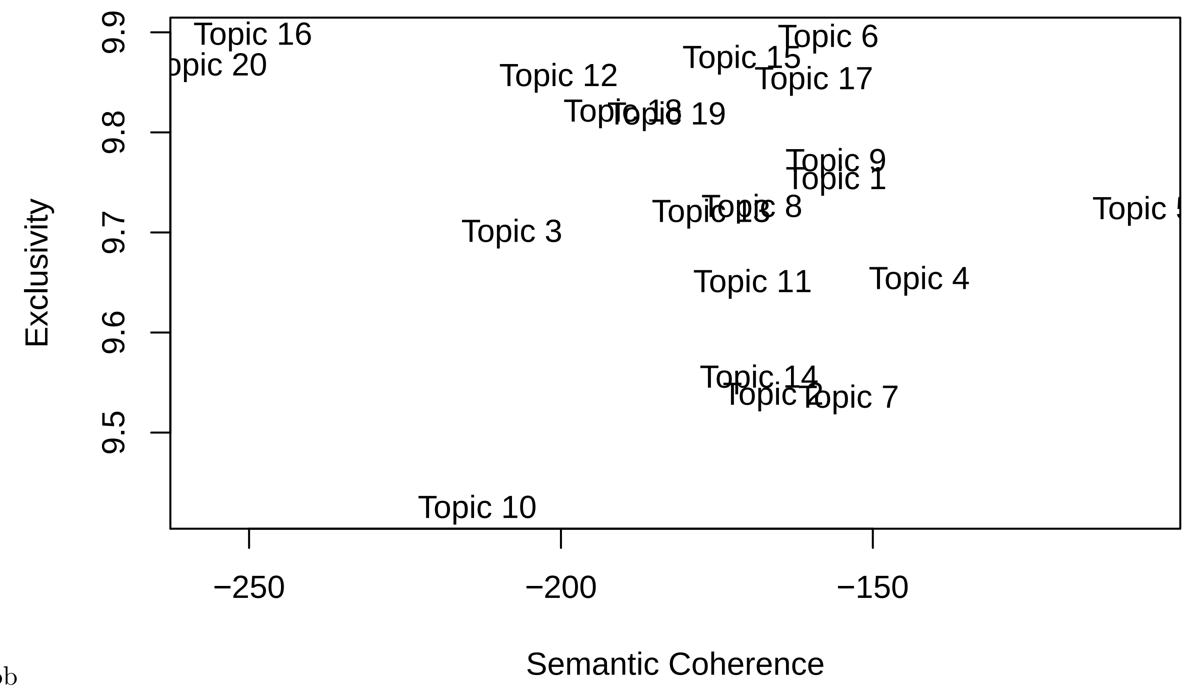


by K-2.bb

```
topicQuality(k_20,dfm_toks_sub)
```

```
##  [1] -155.8810 -165.9696 -207.8772 -142.5555 -106.7070 -157.1547 -153.8948
##  [8] -169.3887 -155.9115 -213.4163 -169.2706 -200.3757 -175.9227 -168.2318
```

```
## [15] -171.0044 -249.4002 -159.4595 -190.0719 -183.0467 -256.6249
##  [1] 9.751484 9.536017 9.698744 9.651605 9.721448 9.893554 9.533039 9.723660
##  [9] 9.769429 9.422858 9.648541 9.854435 9.718662 9.553076 9.872522 9.895744
## [17] 9.851317 9.818935 9.816059 9.865039
```



by K-3.bb

Maybe we have some theory about the difference in topic prevalence across sides (or across parties). We can see the topic proportions in our topic model object:

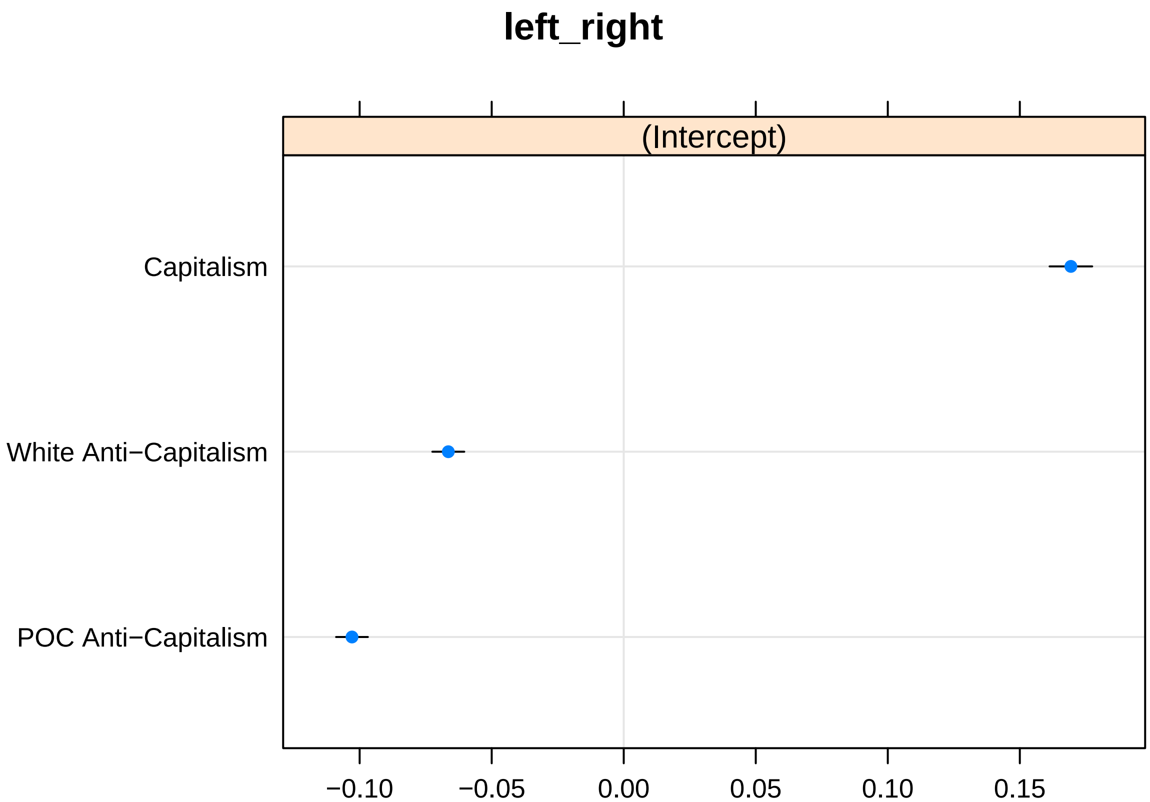
```
head(cap_TM$theta)
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## [1,] 0.07542875 0.02024515 0.49208606 0.07776768 0.09624133 0.06542261
## [2,] 0.15249491 0.18153916 0.08170145 0.03862463 0.30355151 0.05864056
## [3,] 0.16350298 0.13006020 0.07387521 0.07534854 0.12696537 0.10668717
## [4,] 0.31965533 0.03596207 0.14965279 0.06928192 0.09203642 0.09834901
## [5,] 0.02665616 0.01267243 0.03409487 0.50179732 0.01986142 0.23534687
## [6,] 0.02036968 0.01553662 0.01370014 0.04234740 0.01344656 0.02801444
##            [,7]        [,8]       [,9]      [,10]
## [1,] 0.06764590 0.006605026 0.05305056 0.04550695
## [2,] 0.09278333 0.007145652 0.03322783 0.05029096
## [3,] 0.16127230 0.009931679 0.06931750 0.08303905
## [4,] 0.08599592 0.009958856 0.05526251 0.08384517
## [5,] 0.05048596 0.015786213 0.01677402 0.08652474
## [6,] 0.03254694 0.797484935 0.01129259 0.02526068
```

What about connecting this info to our dfm and see if there are differences in the proportion topic 6 is addressed by each side.

```
cap_prev <- data.frame(topic2 = cap_TM$theta[,2], docvars(dfm_toks_sub))
lmer_topic2 <- lmer(topic2 ~ (1 | left_right), data = cap_prev)
dotplot(ranef(lmer_topic2, condVar = TRUE))
```

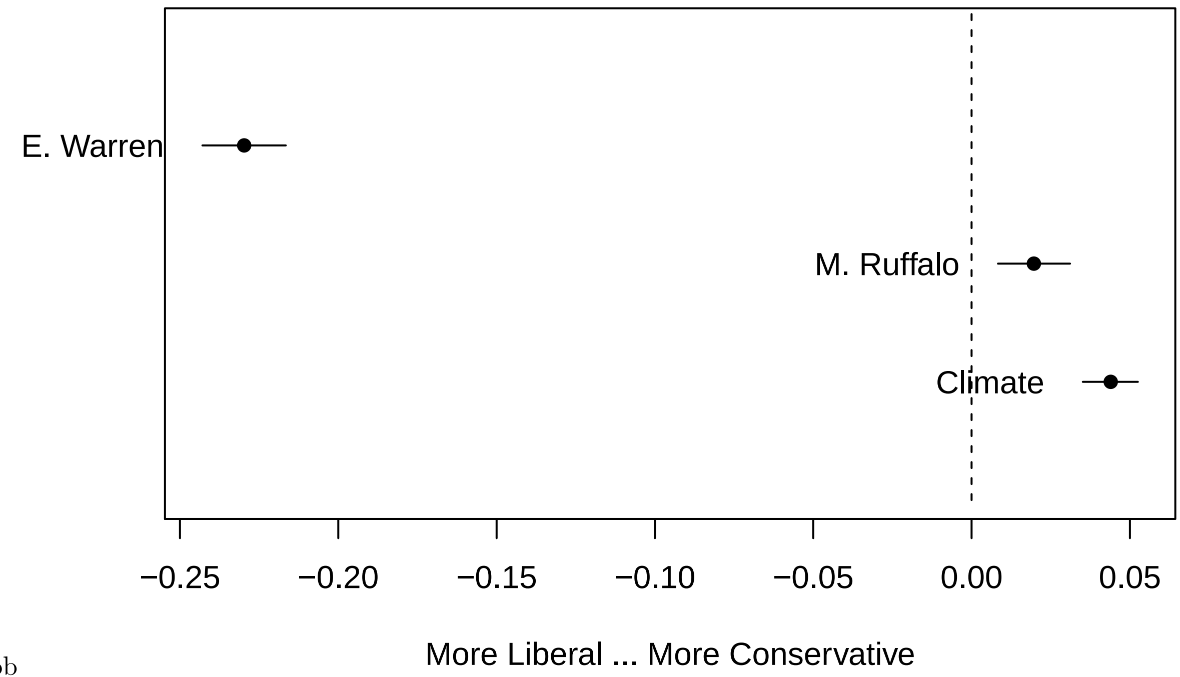
```
## $left_right
```



Topics-1.bb

Makes sense: anti-capitalist bunch together and away from capitalists. We can do something similar with the `stm` function directly. We just need to specify the functional form and add the document variables.

```
cap_topics <- estimateEffect(c(2,4,7) ~ left_right_dum, cap_TM, docvars(dfm_toks_sub)) # You can compar
plot(cap_topics, "left_right_dum", method = "difference",
     cov.value1 = "Anti-Capitalism",
     cov.value2 = "Capitalism",
     labeltype = "custom",
     xlab = "More Liberal ... More Conservative",
     custom.labels = c('E. Warren', 'M. Ruffalo','Climate'),
     model = cap_TM)
```



Effect-1.bb

10 Analyzing a corpus: Scaling models

So far, we have explored tools that provide information about the text. We can also use the text to obtain information about the authors. The **Wordfish** model developed by Slapin and Proksch (2008), for example, positions the authors of documents in an (ideological) scale. How? In politics, the frequency with which politician i uses word k is drawn from a Poisson distribution:

$$w_{ik} Poisson(\lambda_{ik})$$

$$\lambda_{ik} = exp({}_i + {}_k + {}_k \text{»} {}_i)$$

with latent parameters:

- ${}_i$ is the “loquaciousness” of politician i
- ${}_k$ is the frequency of word k
- ${}_k$ is the discrimination parameter of word k
- ${}_i$ is the politician’s ideological position

The parameters of interest are the ’s, the position of the parties in each election year, and the ’s because they allow us to analyze which words differentiate between party positions.

The main assumption is that, indeed, λ_{ik} is generated by the parameters previously described. Let’s believe for a second that the peer-review system works and use the `textmodel_wordfish()` function to estimate the positions of our authorities in our corpus.

```
## I will subset my data since as it is there are too many documents:

data_capitalism <- data_capitalism %>%
  group_by(nameauth) %>%
  mutate(count_auth = n())

data_capitalism_sub <- data_capitalism[data_capitalism$count_auth>8,]
```

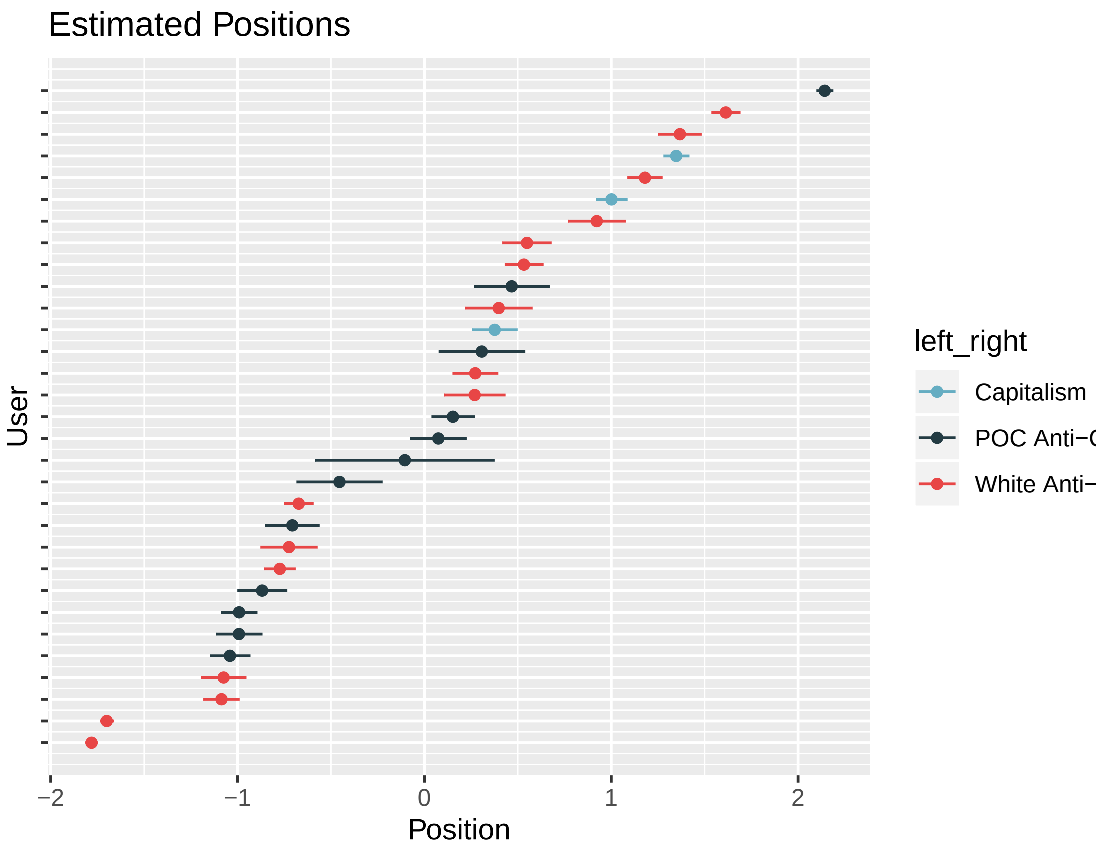
```
## I will also concatenate all the tweets by author:
data_capitalism_sub <- data_capitalism_sub %>%
  group_by(nameauth) %>%
  mutate(text_conca = paste0(text_clean, collapse = " "))

## And finally drop dups:
data_capitalism_sub <- data_capitalism_sub[!duplicated(data_capitalism_sub$nameauth),]

cap_corp_sub <- corpus(data_capitalism_sub$text_conca,
                       docvars = data.frame(author = data_capitalism_sub$nameauth,
                        left_right = data_capitalism_sub$mem_name                   ))

## Again my dfm
cap_dmf_sub <- dfm(cap_corp_sub,
 remove_punct = TRUE,
 remove_numbers = TRUE,
 remove = stopwords("english"),
 stem = T)

cap_wfish <- textmodel_wordfish(cap_dmf_sub, dir = c(1,5)) #Does not really matter what the starting va

summary(cap_wfish)
```

```
##
## Call:
## textmodel_wordfish.dfm(x = cap_dmf_sub, dir = c(1, 5))
##
## Estimated Document Positions:
##           theta      se
## text1   0.27208 0.06189
## text2  -0.99178 0.04885
## text3  -0.77402 0.04343
## text4   0.37626 0.06202
## text5   0.53266 0.05240
## text6   0.54913 0.06721
## text7  -1.70036 0.01783
## text8  -1.04105 0.05488
## text9   1.18058 0.04790
## text10  0.46719 0.10257
## text11 -0.99258 0.06303
## text12  0.92237 0.07797
## text13  0.07457 0.07749
## text14 -0.72464 0.07788
## text15 -0.70685 0.07443
## text16 -0.45435 0.11727
## text17  0.39764 0.09232
## text18  0.26899 0.08314
## text19 -1.07466 0.06112
## text20 -0.10482 0.24448
## text21  1.61314 0.03896
## text22  0.15266 0.05828
## text23  0.30705 0.11761
## text24  1.34803 0.03473
```

```
## text25  1.36736 0.05965
## text26  2.14261 0.02236
## text27 -1.08596 0.04948
## text28 -0.67270 0.04066
## text29 -0.86851 0.06740
## text30  1.00146 0.04266
## text31 -1.78151 0.01650
##
## Estimated Feature Scores:
##        trump conserv republican democrat    amp  liber    link  common    bond
## beta  0.1831  0.3257    -0.1235   0.3558 0.7277 -1.428  0.8157 -0.1832  0.3446
## psi  -1.2239 -1.7671    -3.0482  -1.7695 0.6283 -2.368 -3.1269 -2.1448 -2.3282
##        capit  imperi  coloni   white supremaci patriarchi jingoism   allow
## beta 0.09024 -0.9719 -0.4575  0.4393   -0.1455    -0.1455   0.5496 -0.1235
## psi  2.40763 -2.0284 -2.7410 -1.6457   -3.0527    -3.0527  -3.7443 -3.0482
##      maintain   relat  luxuri lifestyl  expens  global    major planet   thus
## beta  -0.2717 -0.3965  0.1206    1.413 -0.1662  0.5085  0.01487  1.195  0.626
## psi   -3.0839 -2.0252 -2.6122   -2.523 -2.3640 -0.8460 -0.94706 -2.051 -2.376
##         call  realli personifi virtual
## beta  0.5985  0.3681   -0.3714 -0.3714
## psi  -1.3573 -0.9440   -3.1151 -3.1151
```

This is an interesting exercise, since the reasoning behind **wordfish** is similar to the one behind network analysis. In network analysis, rather than looking at the text, we look at the connections made from Tweets and ReTweets. Let's see how both approaches comapare:

```r
cap_preds <- predict(cap_wfish, interval = "confidence")
cap_pos <- data.frame(docvars(cap_corp_sub),
                       cap_preds$fit) %>%
  arrange(fit)

cap_pos <- cap_pos[order(cap_pos$fit),] # sort

ggplot(cap_pos, aes(x = fit, y = 1:nrow(cap_pos), xmin = lwr, xmax = upr,color = left_right)) +
   geom_point() +
   geom_errorbarh(height = 0) +
   scale_y_continuous(labels = cap_pos$docs, breaks = 1:nrow(cap_pos)) +
   labs(x = "Position", y = "User") +
   ggtitle("Estimated Positions")
```

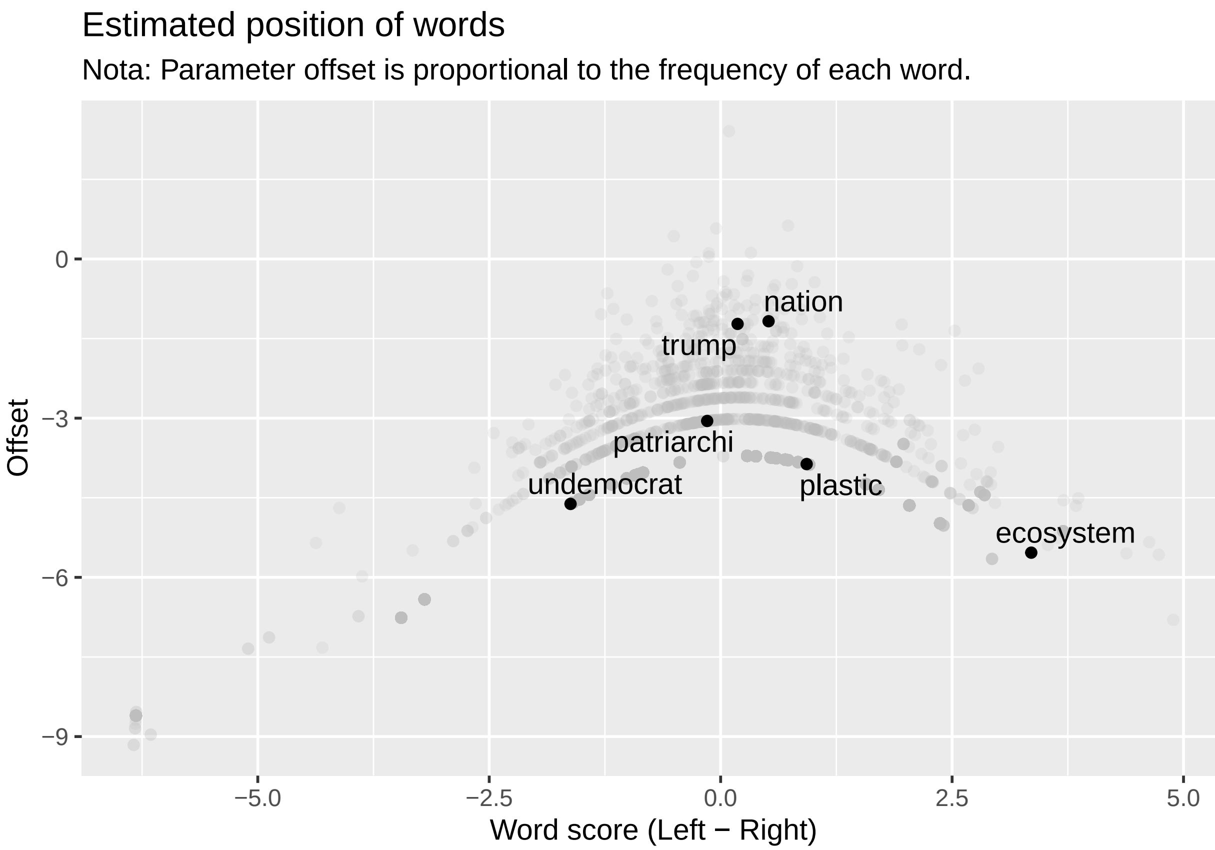


Positions from WordFish-1.bb

External validation, beibi! (?)

We can also turn around the scaling and see where each word is positioned on the same left-right scale as the authors. Here is the "Eiffel Tower" Slapin and Proksch (2008) and of scaled words:

```
wscores <- data.frame(word = cap_wfish$features,
                        score = cap_wfish$beta,
                        offset = cap_wfish$psi)

wscores <- wscores[wscores$score<5,]

testwords <- c("oligarch", "ecosystem", "undemocrat","plastic",
                 "nation", "trump", "patriarchi")

testscores <- wscores %>%
    filter(word %in% testwords) %>%
    arrange(score)

ggplot(wscores, aes(score, offset, label = word)) +
    geom_point(color = "grey", alpha = 0.2) +
    geom_text_repel(data = testscores, col = "black") +
    geom_point(data = testscores) +
    labs(x = "Word score (Left - Right)", y = "Offset") +
    ggtitle("Estimated position of words",
            subtitle = "Nota: Parameter offset is proportional to the frequency of each word.")
```



Tower-1.bb

One important limitation of **wordfish** is that it assumes that all the documents are addressing the same topic, which is not necessarily the case. But there are scaling models for every taste (for example this and this), so this should not be too much of a problem. Still, it is incumbent upon the researcher to choose the model that best reflects the data and the needs.

11 Analyzing a corpus: Natural Language Processing (NLP)

Working with natural language is not a solved problem. Language is messy and ever-changing and evolving. It takes us the better part of our childhood to learn it, "it is hard for the scientist who attempts to model the relevant phenomena, and it is hard for the engineer who attempts to build systems that deal with natural language input or output." (Kornai 2008))

"Statistical NLP aims to do statistical inference for the field of natural language. Statistical inference in general consists of taking some data (generated in accordance with some unknown probability distribution) and then making some inference about this distribution." (Manning and Schutze 1999)

NLP has been used for a while in software: word predictors in your phone, spell-checkers, spam filtering, etc. Its implementation in political science is still limited but the possibilities are vast. NLP makes it possible to move beyond simply establishing connections to investigating the state of relationships, from example by moving from 'whom' to 'who did what to whom.' (Welbers et al. 2017)

We will be testing three advanced NLP techniques: lemmatization, part-of-speech (POS) tagging, and dependency parsing.

11.0.1 Lemmatization

Much like stemming, but rather than cutting off words a dictionary is used to replace terms with their **lemmas**. More accurate at normalizing words with different verb forms (e.g. "gave" and "give"), which is a desirable quality when pre-processing a corpus.

11.0.2 Part-of-speech tagging

Syntactic categories for words, such as nouns, verbs, articles, and adjectives.

From Welbers et al. 2017: "This information can be used to focus an analysis on certain types of grammar categories, for example, using nouns and proper names to measure similar events in news items (Welbers, et al., 2016), or using adjectives to focus on subjective language (De Smedt and Daelemans, 2012)."

```
cap_parsed <- spacy_parse(data_capitalism$text_clean)
```

```
## Finding a python executable with spaCy installed...

## spaCy (language model: en) is installed in /usr/local/bin/python3

## successfully initialized (spaCy Version: 2.0.16, language model: en)

## (python options: type = "python_executable", value = "/usr/local/bin/python3")
```

```
head(cap_parsed,25)
```

```
##    doc_id sentence_id token_id         token      lemma   pos     entity
## 1   text1           1        1            if         if   ADP
## 2   text1           1        2           you     -PRON-  PRON
## 3   text1           1        3           're         be  VERB
## 4   text1           1        4        having       have  VERB
## 5   text1           1        5       trouble    trouble  NOUN
## 6   text1           1        6 understanding understand  VERB
## 7   text1           1        7    capitalism capitalism  NOUN
## 8   text1           1        8          just       just   ADV
## 9   text1           1        9      remember   remember  VERB
## 10  text1           1       10         there      there   ADV
## 11  text1           1       11           are         be  VERB
## 12  text1           1       12          only       only   ADV
## 13  text1           1       13             4          4   NUM CARDINAL_B
## 14  text1           1       14      websites    website  NOUN
## 15  text1           1       15           you     -PRON-  PRON
## 16  text1           1       16           use        use  VERB
## 17  text1           1       17            in         in   ADP
## 18  text1           1       18          2019       2019   NUM     DATE_B
## 19  text1           2        1           and        and CCONJ
## 20  text1           2        2           you     -PRON-  PRON
## 21  text1           2        3          hate       hate  VERB
## 22  text1           2        4           all        all   DET
## 23  text1           2        5            of         of   ADP
## 24  text1           2        6          them     -PRON-  PRON
## 25  text2           1        1             "          " PUNCT
```

11.0.3 Dependency parsing

Dependency parsing provides the syntactic relations between tokens. For example, "Kendrick" is related to "Lamar", thus recognizing "Kendrick Lamar" as a single entity. This can be particularly useful is you are searching for certain types of entities (e.g. locations, institutions, etc.) in a corpus. We can see the differences in the location mentioned by capitalists and anti-capitalists:

```
cap_entities <- entity_extract(cap_parsed)
head(cap_entities,20)
```

```
##    doc_id sentence_id                 entity entity_type
## 1   text6           1   Ariana_Grande_Breaks      PERSON
## 2   text6           2         Bernie_Sanders      PERSON
## 3   text7           3                     \n         GPE
## 4   text7           4                 Bernie      PERSON
## 5   text7           4                America         GPE
## 6   text7           5                     \n         GPE
## 7   text7           6            @cornelWest         ORG
## 8   text7           6        @chaunceydevega         ORG
## 9   text7           6                     \n         GPE
## 10 text10           1          Arundhati_Roy      PERSON
## 11 text11           2 Elites_Lost_Their_Grip      PERSON
## 12 text11           2                   Time         ORG
## 13 text12           2                    Dem        NORP
## 14 text12           3                   \n\n         ORG
## 15 text12           3               \n\n_Dem      PERSON
## 16 text12           4        \n\n_Capitalism      PERSON
## 17 text12           5                   \n\n         ORG
## 18 text12           5        \n\n_Capitalism      PERSON
## 19 text14           2               American        NORP
## 20 text15           1    DemocraticSocialism         ORG
```

```
# Similar to extract but converts  multi-word entities into single "tokens":
# cap_entities <- entity_consolidate(cap_parsed_dep)

cap_persons <- cap_entities$entity[cap_entities$entity_type == "PERSON"]
cap_persons <- unique(cap_persons)
head(cap_persons,40)
```

```
##  [1] "Ariana_Grande_Breaks"   "Bernie_Sanders"         "Bernie"
##  [4] "Arundhati_Roy"          "Elites_Lost_Their_Grip" "\n\n_Dem"
##  [7] "\n\n_Capitalism"        "Ilhan_Omar"             "Free_Speech"
## [10] "Mark_Ruffalo"           "Ariana_Grande"          "John_Legend"
## [13] "Chrissy_Teigen"         "Elizabeth_Warren"       "Harry_Haywood"
## [16] "Angela_Davis"           "Mark"                   "We_Need_'_Revolution_'"
## [19] "Einstein"               "Wayne"                  "Nonstop"
## [22] "Deval_Patrick_'s"       "Hitler"                 "\n\n_Me"
## [25] "\n\n_Marx"              "Danny"                  "Greed"
## [28] "Deval_Patrick_’s"       "Jobs"                   "Colin"
## [31] "Nigel_Farage"           "Lamar"                  "Pete"
## [34] "Rhodes_Scholar"         "Obama"                  "\n\n_Spreading"
## [37] "Black"                  "Hillary_Clinton"        "John_Cornyn"
## [40] "Lech_Walesa"
```

In other contexts, spaCyR can recognize which subject is doing the action and which subject is on the receiving end. Van Atteveldt et al. (2017) use this to analyze who is attacking whom in news about the

Gaza war. Here we can see something similar at work. Let's….

```
cap_parsed_dep <- spacy_parse(data_capitalism$text_clean, dependency = TRUE, entity = TRUE, lemma = FALS
head(cap_parsed_dep,25)
```

```
##    doc_id sentence_id token_id         token   pos tag head_token_id dep_rel
## 1   text1           1        1            if   ADP  IN             4    mark
## 2   text1           1        2           you  PRON PRP             4   nsubj
## 3   text1           1        3           're  VERB VBP             4     aux
## 4   text1           1        4        having  VERB VBG             9   advcl
## 5   text1           1        5       trouble  NOUN  NN             4    dobj
## 6   text1           1        6 understanding  VERB VBG             5     acl
## 7   text1           1        7    capitalism  NOUN  NN             6    dobj
## 8   text1           1        8          just   ADV  RB             9  advmod
## 9   text1           1        9      remember  VERB VBP             9    ROOT
## 10  text1           1       10         there   ADV  EX            11    expl
## 11  text1           1       11           are  VERB VBP             9   ccomp
## 12  text1           1       12          only   ADV  RB            13  advmod
## 13  text1           1       13             4   NUM  CD            14  nummod
## 14  text1           1       14      websites  NOUN NNS            11    attr
## 15  text1           1       15           you  PRON PRP            16   nsubj
## 16  text1           1       16           use  VERB VBP            14   relcl
## 17  text1           1       17            in   ADP  IN            11    prep
## 18  text1           1       18          2019   NUM  CD            17    pobj
## 19  text1           2        1           and CCONJ  CC             3      cc
## 20  text1           2        2           you  PRON PRP             3   nsubj
## 21  text1           2        3          hate  VERB VBP             3    ROOT
## 22  text1           2        4           all   DET  DT             3    dobj
## 23  text1           2        5            of   ADP  IN             4    prep
## 24  text1           2        6          them  PRON PRP             5    pobj
## 25  text2           1        1             " PUNCT  ``             2    amod
##        entity
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13 CARDINAL_B
## 14
## 15
## 16
## 17
## 18     DATE_B
## 19
## 20
## 21
## 22
## 23
```

```
## 24
## 25
```

spacyR can also detect other attributes of tokens in a text:

```r
cap_parsed_att <- spacy_parse(data_capitalism$text,
            additional_attributes = c("like_num", "like_url"),
            lemma = FALSE, pos = FALSE, entity = FALSE)
head(cap_parsed_att,30)
```

```
##    doc_id sentence_id token_id         token like_num like_url
## 1   text1           1        1            if    FALSE    FALSE
## 2   text1           1        2           you    FALSE    FALSE
## 3   text1           1        3           're    FALSE    FALSE
## 4   text1           1        4        having    FALSE    FALSE
## 5   text1           1        5       trouble    FALSE    FALSE
## 6   text1           1        6 understanding    FALSE    FALSE
## 7   text1           1        7    capitalism    FALSE    FALSE
## 8   text1           1        8          just    FALSE    FALSE
## 9   text1           1        9      remember    FALSE    FALSE
## 10  text1           1       10         there    FALSE    FALSE
## 11  text1           1       11           are    FALSE    FALSE
## 12  text1           1       12          only    FALSE    FALSE
## 13  text1           1       13             4     TRUE    FALSE
## 14  text1           1       14      websites    FALSE    FALSE
## 15  text1           1       15           you    FALSE    FALSE
## 16  text1           1       16           use    FALSE    FALSE
## 17  text1           1       17            in    FALSE    FALSE
## 18  text1           1       18          2019     TRUE    FALSE
## 19  text1           2        1           and    FALSE    FALSE
## 20  text1           2        2           you    FALSE    FALSE
## 21  text1           2        3          hate    FALSE    FALSE
## 22  text1           2        4           all    FALSE    FALSE
## 23  text1           2        5            of    FALSE    FALSE
## 24  text1           2        6          them    FALSE    FALSE
## 25  text2           1        1             "    FALSE    FALSE
## 26  text2           1        2    capitalism    FALSE    FALSE
## 27  text2           1        3           not    FALSE    FALSE
## 28  text2           1        4      cronyism    FALSE    FALSE
## 29  text2           1        5             "    FALSE    FALSE
## 30  text2           1        6        poster    FALSE    FALSE
```

12 Other techniques not covered

12.1 Word networks

How are words and authors connected? We can create networks of authors as nodes and edges based on the overlap in words between authors. Let's use the `textnet` package created by Chris Bail:

```r
# library(devtools)
# install_github("cbail/textnets")
library(textnets)
```

```
## Loading required package: udpipe
```

```
## Warning: package 'udpipe' was built under R version 3.5.2

## Loading required package: ggraph

## Warning: package 'ggraph' was built under R version 3.5.2

## Loading required package: networkD3

## Warning: replacing previous import 'dplyr::union' by 'igraph::union' when
## loading 'textnets'

## Warning: replacing previous import 'dplyr::as_data_frame' by
## 'igraph::as_data_frame' when loading 'textnets'

## Warning: replacing previous import 'dplyr::groups' by 'igraph::groups' when
## loading 'textnets'
```

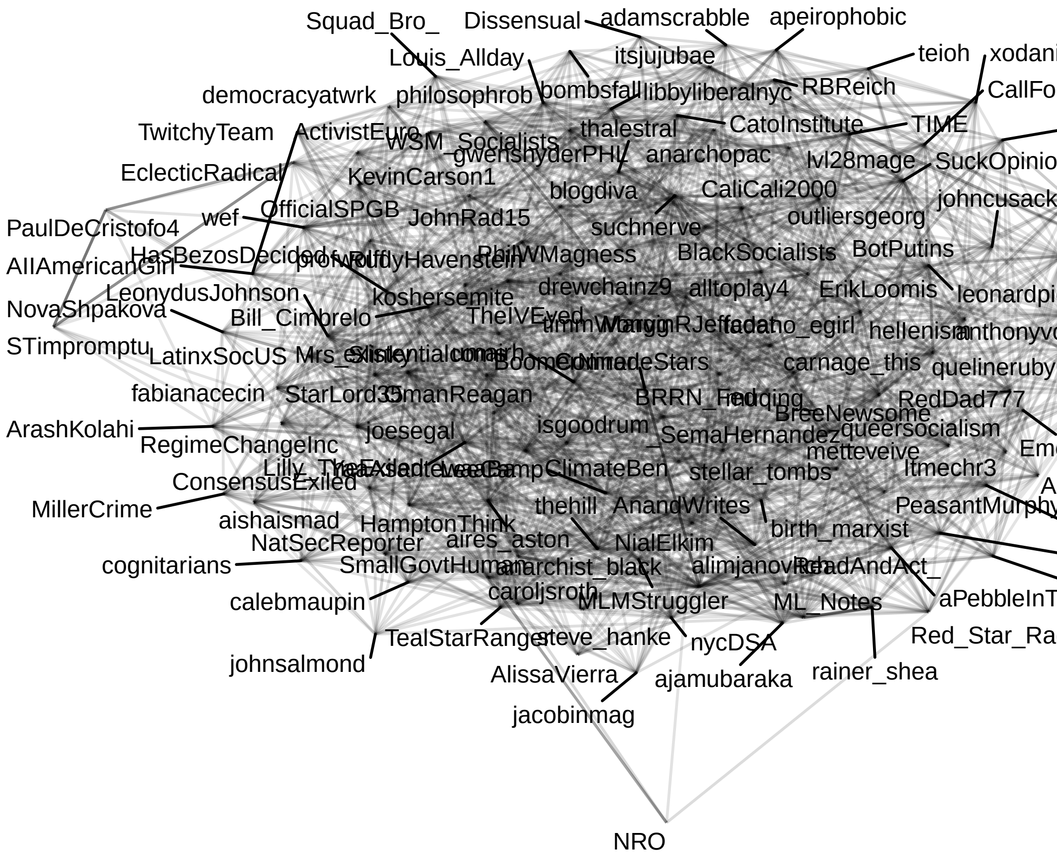
```
# I will subset my data but lower the threshold to get more authorities:

data_capitalism_sub <- data_capitalism[data_capitalism$count_auth>4,]

## I will also concatenate all the tweets by author:
data_capitalism_sub <- data_capitalism_sub %>%
  group_by(nameauth) %>%
  mutate(text_conca = paste0(text_clean, collapse = " "))

## And finally drop dups:
data_capitalism_sub <- data_capitalism_sub[!duplicated(data_capitalism_sub$nameauth),]

# We are only going to be using nouns.
data_capitalism_sub$text_conca_nocap <- str_remove_all(data_capitalism_sub$text_conca,"[Cc]apitalism")

# We prep our data (takes a while):
prepped_cap <- PrepText(data_capitalism_sub, groupvar = "nameauth", textvar = "text_conca_nocap", node_
```

```
## Downloading udpipe model from https://raw.githubusercontent.com/jwijffels/udpipe.models.ud.2.4/maste

## Visit https://github.com/jwijffels/udpipe.models.ud.2.4 for model license details
```

To create the adjacency matrix for the network, we use the `CreateTextnet()` function. The cells of the adjacency matrix are the transposed crossproduce of the term-frequency inverse-document frequency (TFIDF) for overlapping terms between two documents for PrepText and the matrix product of TFIDF crosspropduct (See Bail, 2016).

```
cap_text_network <- CreateTextnet(prepped_cap)
VisTextNet(cap_text_network, label_degree_cut = 0)
```

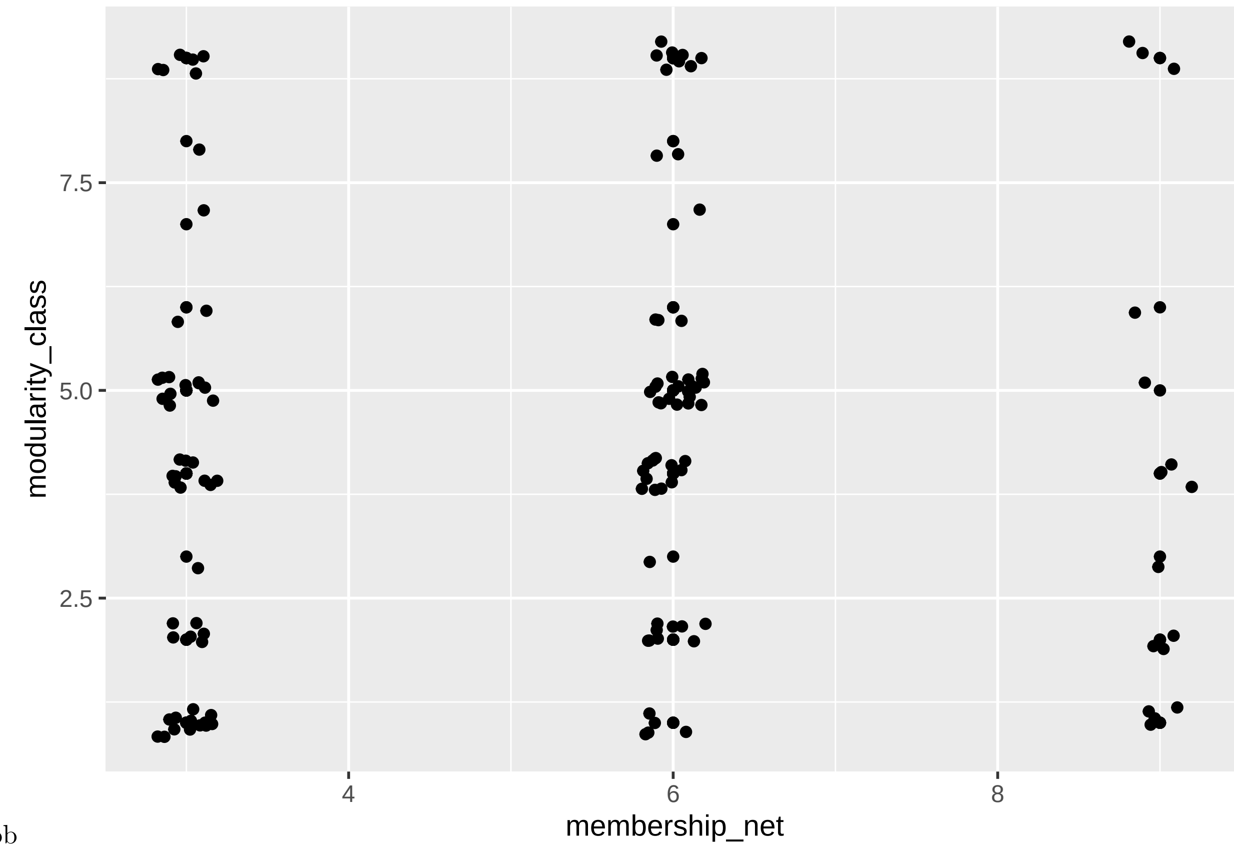
```
## Using `stress` as default layout
```



and Visualizing the Network-1.bb

A mess... We can see the communities and compare with the communities that we got from the original network analysis.

```r
cap_communities <- TextCommunities(cap_text_network)

cap_communities_full <- cbind.data.frame(cap_communities, data_capitalism_sub$to_membership)
colnames(cap_communities_full)[3] <- "membership_net"
cap_communities_full$modularity_class <- as.numeric(cap_communities_full$modularity_class)

ggplot(cap_communities_full, aes(x=membership_net, y=modularity_class)) +
  geom_point() +
  geom_jitter(width = 0.2, height = 0.2)
```



Communities-1.bb

We would expect a bit more exclusivity of membership, but oh well…

12.2 Cosine-similarity

How similar are two texts? If we think of each text as a vector in space, we can think of the angle between the two vectors as their "distance". The smaller the angle θ, the closer and the more similar. If we had a measure of each text (e.g. TF-IDF) we could compute the cosine of the angle between them. How? Math. We must solve the dot product for the $\cos\theta$:

$$\vec{a} \cdot \vec{b} = \|\vec{a}\|\|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

That is the cosine similarity formula. Cosine similarity will generate a metric that says how related are two documents by looking at the angle instead of the magnitude:

12.3 Word embeddings

Word embeddings use a similar intuition as cosine similarities. By using vector representation of text, we can estimate how much alike two words are rather than treating words as single independent units (which is the case for dictionaries).

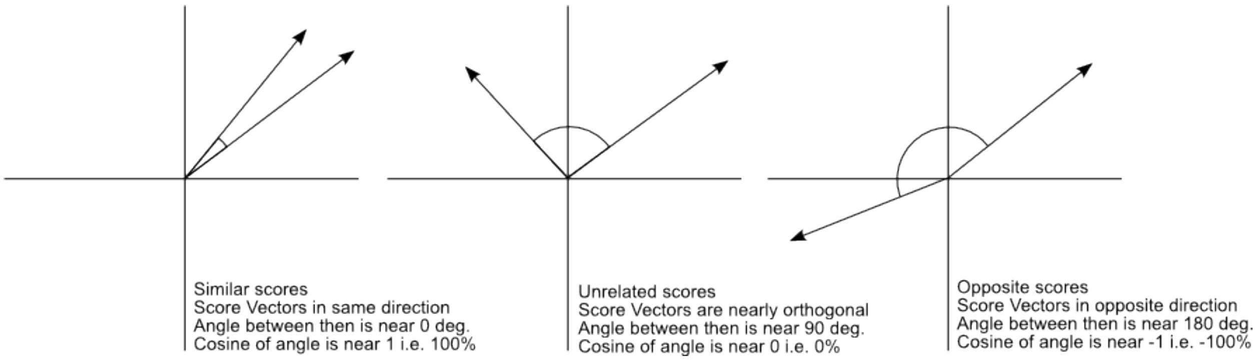
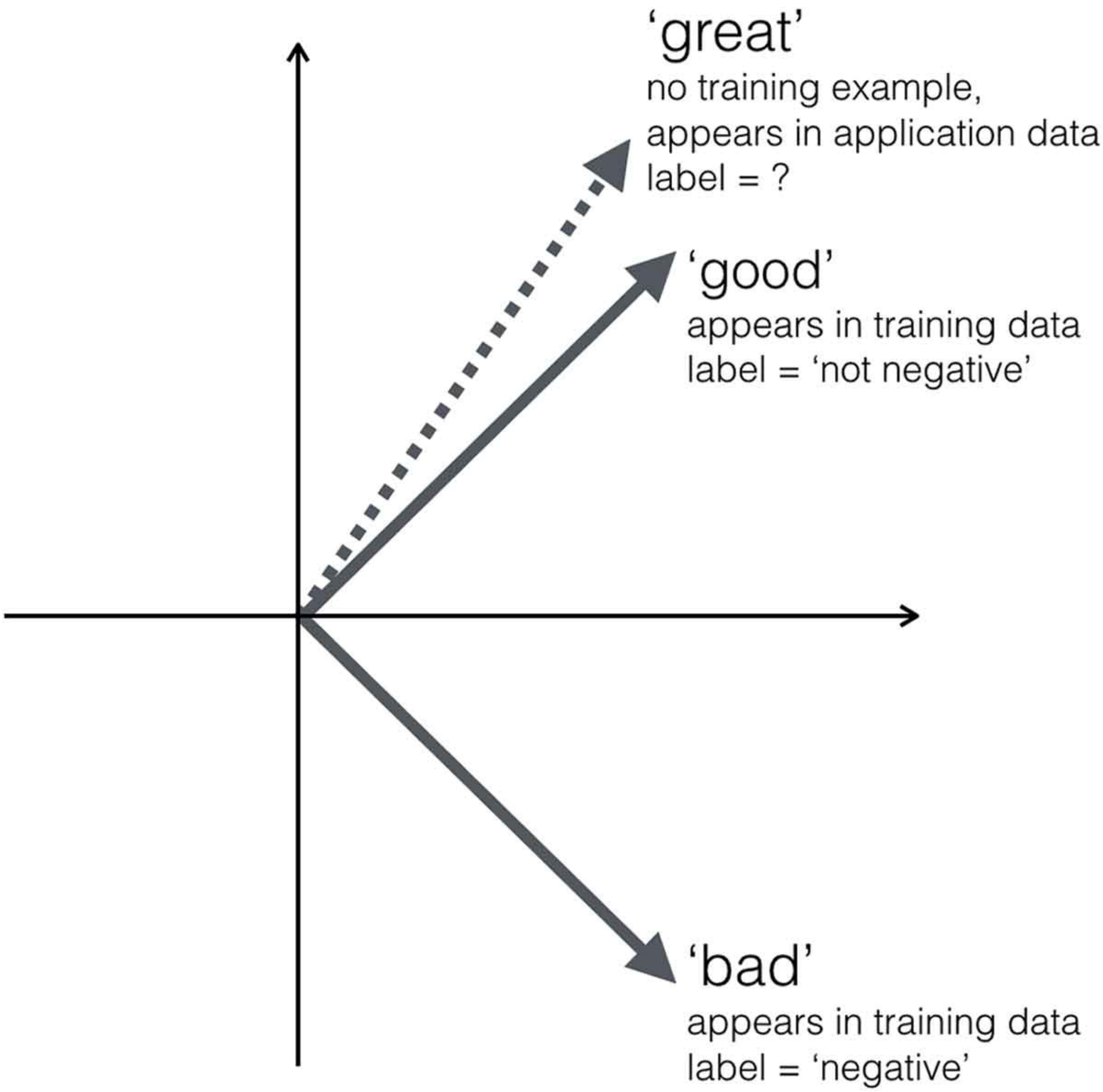


Figure 6: Cosine similarity



We can train our models to predict how words are related. This requires a previously annotated training data set and some technical skills (and the pre-processing of the data). The literature suggest that word embeddings can produce better results than similar "bag-of-words" approaches. It is particularly attractive since these models can be ran in most languages, helping us overcome the limitations from canned English-focused dictionaries.

13 Goodbye Notes

- If your are interested in text analysis, the internet is your best friend. I am also happy to help. And to be your friend.
- The resources suggested at the beginning will help you navigate text analysis and see how it fits within your research.
- Learn to scrape the interwebs.
- Learn regex.
- Ask.
- Remember: 2 locations, 3 versions for all your data. Backups must be *Regular, Automatic, and Incremental*.