Advanced Quantitative Text Analysis Using R and **quanteda**

Kenneth Benoit

Masterclass 2019-05-15

Contents

Textual statistics

Simple frequency analysis

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

ire_freqplot <- ire_dfm %>%
    textstat_frequency(n = 15) %>%
    ggplot(aes(x = reorder(feature, frequency),
               y = frequency)) +
    geom_point() +
    coord_flip() +
    labs(x = NULL, y = "Frequency")
```

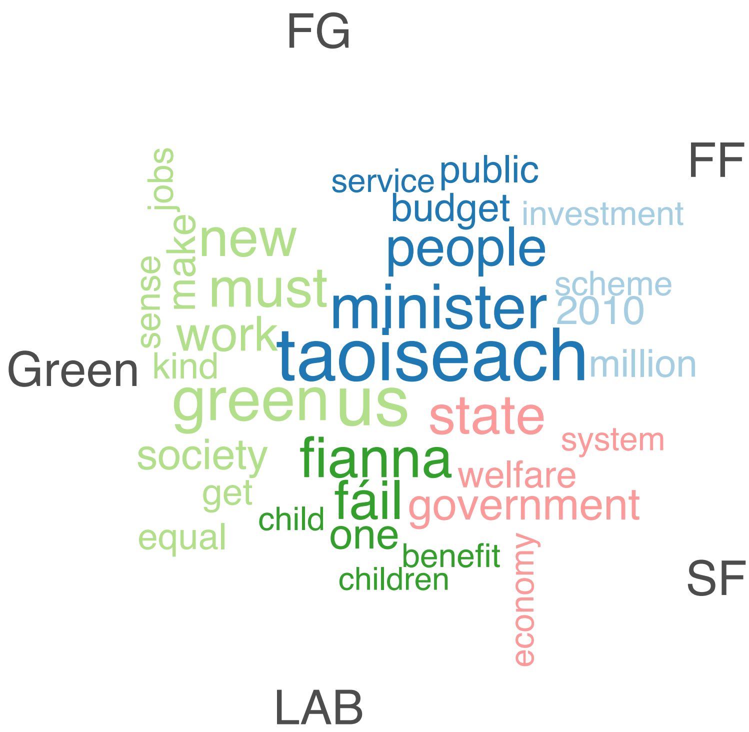
```
ire_freqplot
```

```
## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '€' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '€' in 'mbcsToSbcs': dot substituted for <82>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '€' in 'mbcsToSbcs': dot substituted for <ac>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '€' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '€' in 'mbcsToSbcs': dot substituted for <82>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '€' in 'mbcsToSbcs': dot substituted for <ac>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '€' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '€' in 'mbcsToSbcs': dot substituted for <82>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '€' in 'mbcsToSbcs': dot substituted for <ac>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '€' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in grid.Call(C_textBounds, as.graphicsAnnot(x$label), x$x, x$y, :
## conversion failure on '€' in 'mbcsToSbcs': dot substituted for <82>
```

Frequency analysis for groups

```
ire_dfm_group <- ire_dfm %>%
    dfm_group(groups = "party")
```

```r
# plot a word cloud (not recommended...)
set.seed(34)
textplot_wordcloud(ire_dfm_group,
                   comparison = TRUE,
                   max_words = 40)
```

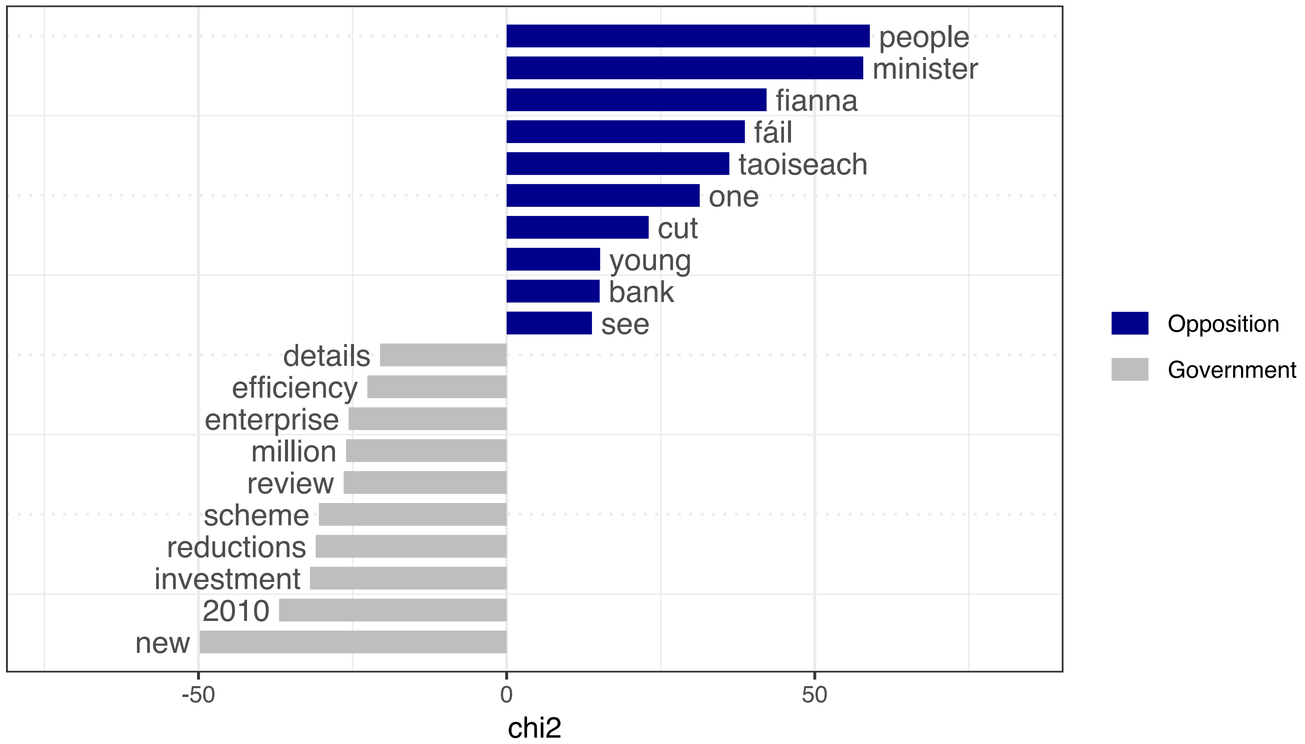


Relative frequency analysis (keyness)

"Keyness" assigns scores to features that occur differentially across different categories

```r
docvars(data_corpus_irishbudget2010, "gov_opp") <-
    ifelse(docvars(data_corpus_irishbudget2010, "party") %in%
               c("FF", "Green"), "Government", "Opposition")

# compare government to opposition parties by chi^2
dfmat_key <- data_corpus_irishbudget2010 %>%
     dfm(groups = "gov_opp",
         remove = stopwords("english"),
         remove_punct = TRUE, remove_symbols = TRUE)

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

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

```
tplot_key
```


people
minister
fianna
fáil
taoiseach
one
cut
young
bank
see
details
efficiency
enterprise
million
review
scheme
reductions
investment
2010
new
-50
0
50
chi2
Opposition
Government

Term frequency-inverse document frequency (tf-idf)

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

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

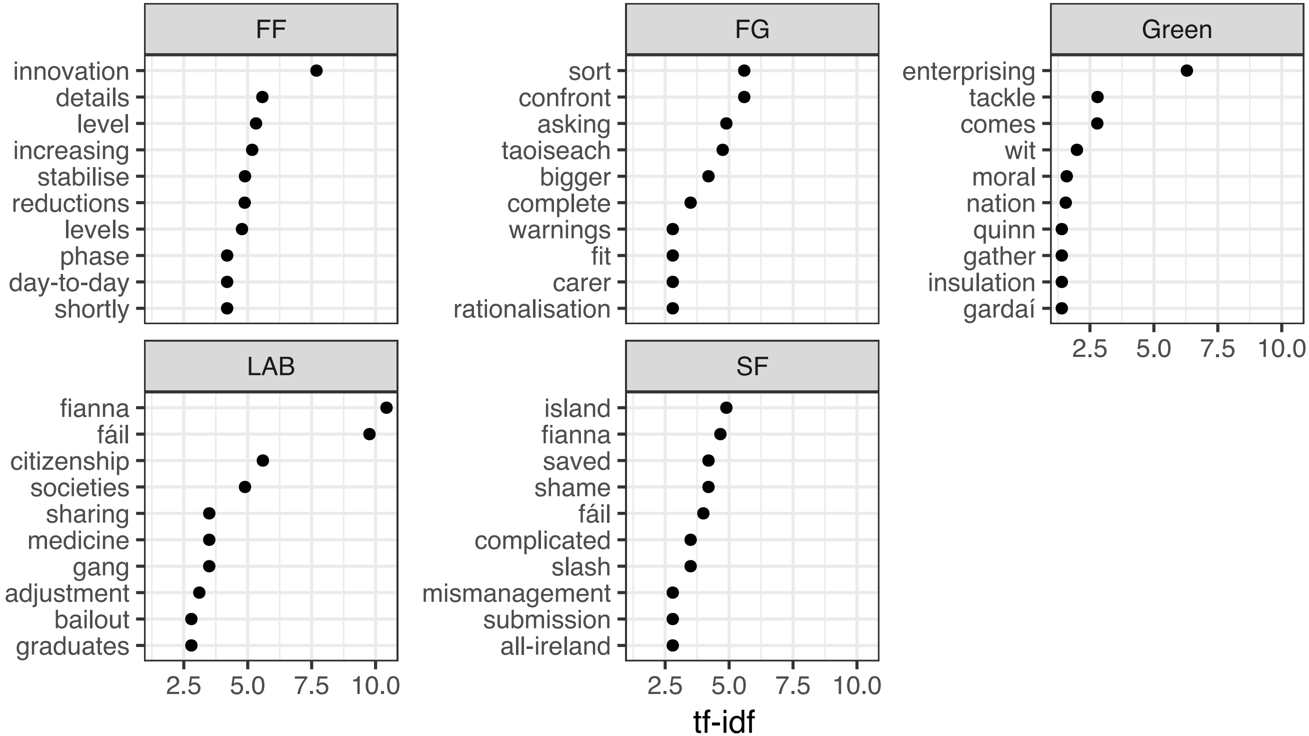
Apply tf-idf weighting

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

Plot tf-idf

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

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

```
tplot_tfidf
```


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

Collocation analysis: a strings of words approach

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

head(tstat_col, 7)
```

```
##      collocation count count_nested length   lambda        z
## 1          it is   188            0      2 3.683338 36.89575
## 2        will be   142            0      2 4.132301 35.59424
## 3    this budget   107            0      2 4.295062 31.81840
## 4        we have   119            0      2 3.382721 29.50924
## 5      more than    56            0      2 6.297167 28.82909
## 6 social welfare    70            0      2 8.081143 28.82286
## 7         in the   347            0      2 1.855052 27.87367
```

Exercise

1. Repeat the step above, but remove stopwords, and stem the `tokens` object.
2. Compare the most frequent collocations. What has changed?
3. Optional: Conduct a collocation analysis for the German inaugural speeches. Does it work well?

Solution

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

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

head(tstat_col_adjusted, 7)
```

```
##      collocation count count_nested length   lambda        z
## 1 public servic    68            0      2 5.467132 26.76130
## 2 social welfar    65            0      2 7.168046 25.84665
## 3 child benefit    36            0      2 6.875431 21.50678
## 4      per week    25            0      2 5.828731 19.40465
## 5     next year    34            0      2 5.200994 18.86080
## 6 public sector    30            0      2 4.089028 17.70144
## 7  Labour Parti    23            0      2 7.486375 17.12913
```

```
nrow(tstat_col_adjusted)
```

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

Compound collocations

Before compounding

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

```
## kwic object with 0 rows
```

Compound collocations

Compound based on collocations

```
# get 50 most frequent collocations

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

After compounding

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

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

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

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

Compound collocations

Check compounded words

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

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

Similarity between texts

Cosine similarity

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

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

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

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

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

```
##           text1
## text2 0.9746318
```

Cosine similarity

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

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

```
##           text1     text2
## text2 0.8571429
## text3 0.1428571 0.1428571
```

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

BONUS: Transform matrix into dyadic dataframe

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

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

tstat_simil_df
```

```
##    Var1  Var2     value
## 4 text1 text2 0.8571429
## 7 text1 text3 0.1428571
## 8 text2 text3 0.1428571
```

Exercise

1. Create a dfm from `data_corpus_irishbudget2010`
2. Group it by party and run `textstat_simil()`.
3. What parties are most similar?
4. Do the substantive conclusion change when removing stopwords and punctuation, and stemming the dfm?

Solution

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

tstat_simil1
```

```
##              FF        FG     Green       LAB
## FG    0.9540889
## Green 0.9454957 0.9495884
## LAB   0.9632787 0.9825914 0.9505133
## SF    0.9683231 0.9790162 0.9384105 0.9804133
```

Solution

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

tstat_simil2
```

```
##              FF        FG     Green       LAB
## FG    0.6427297
## Green 0.6594612 0.6518786
## LAB   0.6938182 0.8052380 0.6594711
## SF    0.7206824 0.8010105 0.6550911 0.8247349
```

Lexical diversity

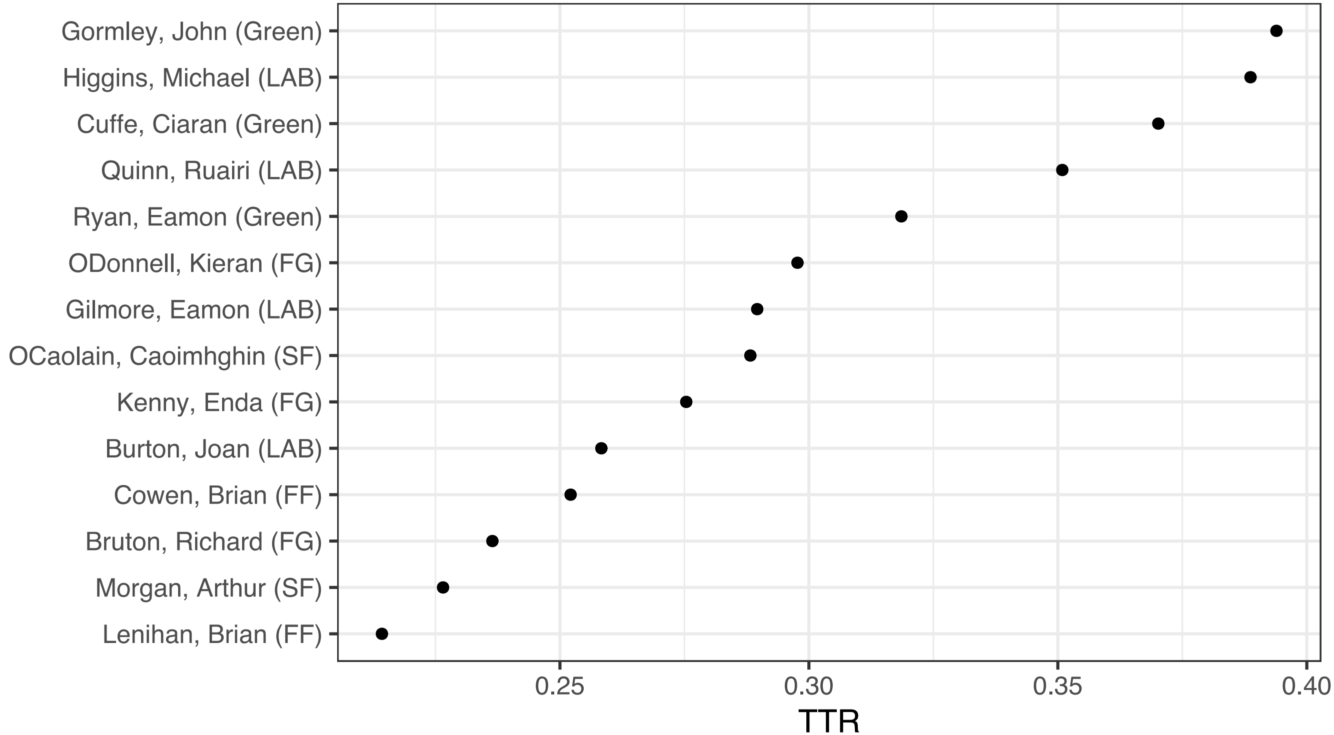
```r
dfmat_ire <- data_corpus_irishbudget2010 %>%
    dfm(remove_punct = TRUE, remove_numbers = TRUE, remove_symbols = TRUE)

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

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

head(df_lexdiv, 4)
```

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

Plot Type-Token Ratio

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

```
tplot_ttr
```



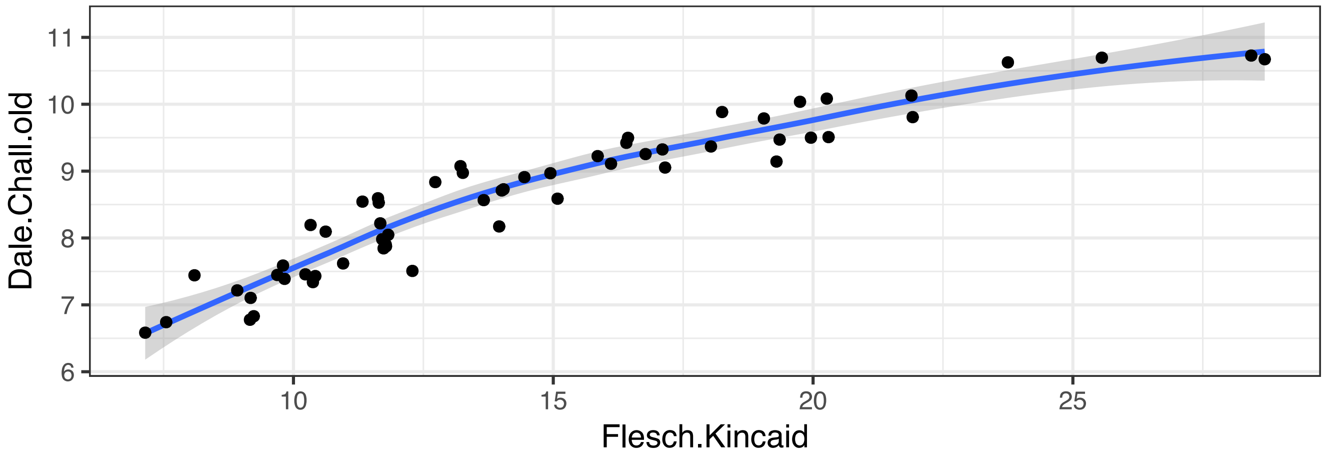
Gormley, John (Green)
Higgins, Michael (LAB)
Cuffe, Ciaran (Green)
Quinn, Ruairi (LAB)
Ryan, Eamon (Green)
ODonnell, Kieran (FG)
Gilmore, Eamon (LAB)
OCaolain, Caoimhghin (SF)
Kenny, Enda (FG)
Burton, Joan (LAB)
Cowen, Brian (FF)
Bruton, Richard (FG)
Morgan, Arthur (SF)
Lenihan, Brian (FF)
0.25
0.30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

Readability

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

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

Plot relationship between readability measures

```
tplot_read
```



Correlation between readability measures

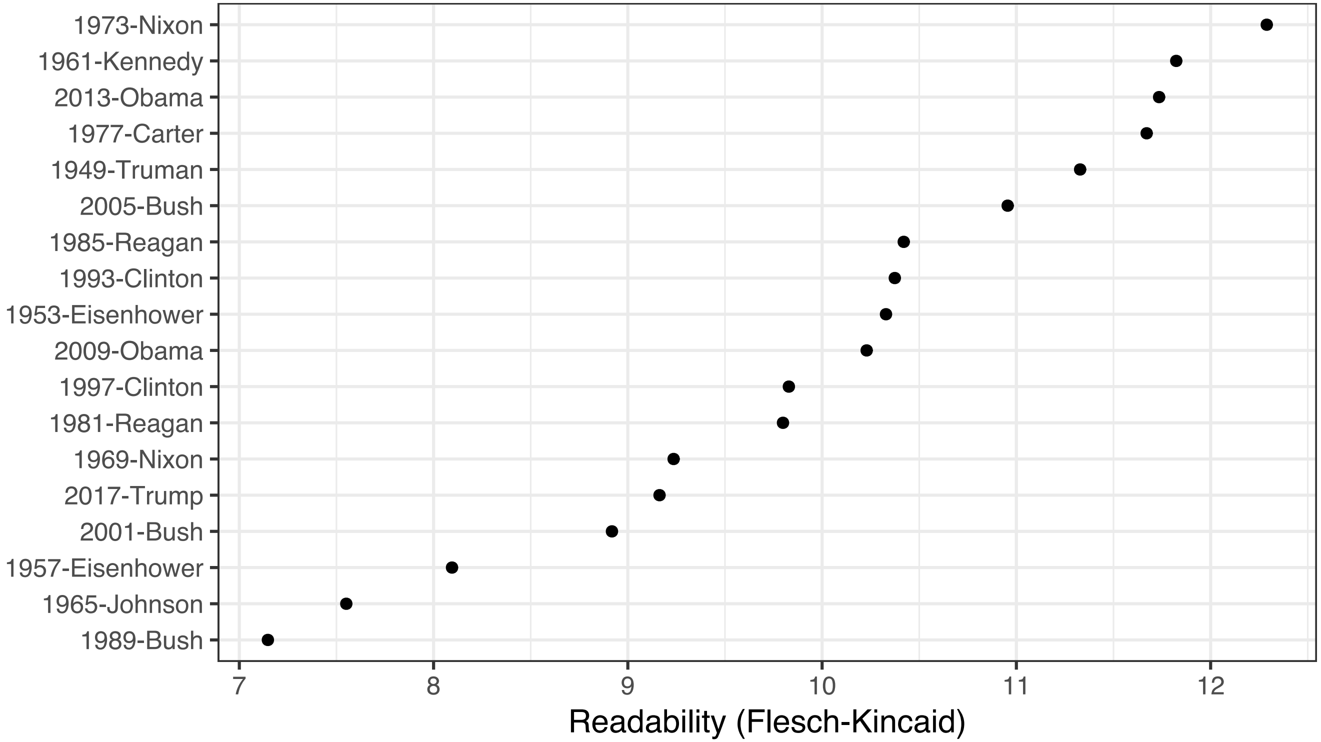
```
cor.test(tstat_read$Flesch.Kincaid, tstat_read$Dale.Chall.old)
```

```
##
##      Pearson's product-moment correlation
##
## data:  tstat_read$Flesch.Kincaid and tstat_read$Dale.Chall.old
## t = 19.714, df = 56, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8920197 0.9611119
## sample estimates:
##       cor
## 0.9349081
```

Readability of inaugural speeches since 1945

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

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

```
tplot_read_us
```



See extensively Benoit et al. (2019)

Exercise: Wrapping it all up

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

Solution

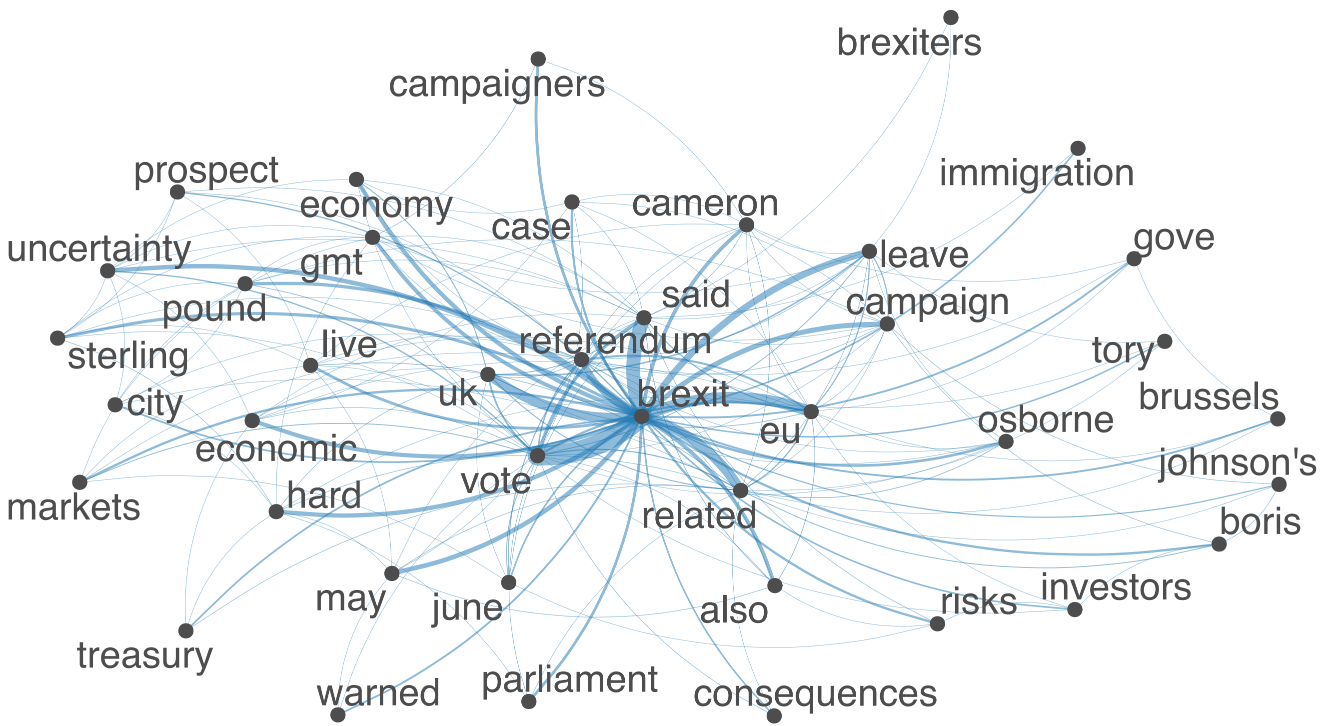
Import Guardian corpus using **quanteda.corpora**'s `download()` function.

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

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

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

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

fcmat_brexit_subset <- fcmat_brexit %>%
    fcm_select(feat)

tplot_brexit <- textplot_network(fcmat_brexit_subset)
```

```
## Registered S3 method overwritten by 'network':
```

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



Supervised text classification

Supervised classification: separate documents into pre-existing categories

We need:

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

Goals of supervised classification

1. Flexibility
2. Efficiency
3. Interpretability

Unsupervised classification

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

Basic principles of supervised learning

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

How to assess the performance of a classifier?

		Positive	Negative
Prediction	Positive	True Positive	False Positive (Type I error)
	Negative	False Negative (Type II error)	True Negative

Important measures

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

How do we get a labelled training set?

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

Naive Bayes: Probabilistic learning

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

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

Naive Bayes

Advantages

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

Disadvantages

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

Naive Bayes in quanteda: Example 1

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

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

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

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

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

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

Naive Bayes in quanteda: Example 2

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

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

# create dummy (before/after 1945)
docvars(data_corpus_inaugural, "pre_post1945") <-
    ifelse(docvars(data_corpus_inaugural, "Year") > 1945, "Post 1945", "Pre 1

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

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

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

Naive Bayes in quanteda: Example 2

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

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

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

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

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

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

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

Naive Bayes in quanteda: Example 2

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

print(tab)
```

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

Naive Bayes in quanteda: Example 2

Assess accuracy using the **caret** package

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

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

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

Scaling models

Scaling: unsupervised and supervised

Supervised

- Wordscores
- "class affinity scaling"

Unsupervised

- Wordfish
- Correspondence Analysis

Supervised scaling

```
tmod <- textmodel_wordscores(data_dfm_lbgexample, y = c(seq(-1.5, 1.5, .75),
summary(tmod)
```

```
##
## Call:
## textmodel_wordscores.dfm(x = data_dfm_lbgexample, y = c(seq(-1.5,
##     1.5, 0.75), NA))
##
## Reference Document Statistics:
##    score total min max  mean median
## R1 -1.50  1000   0 158 27.03      0
## R2 -0.75  1000   0 158 27.03      0
## R3  0.00  1000   0 158 27.03      0
## R4  0.75  1000   0 158 27.03      0
## R5  1.50  1000   0 158 27.03      0
## V1    NA  1000   0 158 27.03      0
##
## Wordscores:
## (showing first 30 elements)
##       A       B       C       D       E       F       G       H       I
## -1.5000 -1.5000 -1.5000 -1.5000 -1.5000 -1.4812 -1.4809 -1.4519 -1.4083
##       J       K       L       M       N       O       P       Q       R
## -1.3233 -1.1846 -1.0370 -0.8806 -0.7500 -0.6194 -0.4508 -0.2992 -0.1306
##       S       T       U       V       W       X       Y       Z      ZA
##  0.0000  0.1306  0.2992  0.4508  0.6194  0.7500  0.8806  1.0370  1.1846
##      ZB      ZC      ZD
##  1.3233  1.4083  1.4519
```

Extracting useful information

```
coef(tmod)
```

```
##          A          B          C          D          E          F
## -1.5000000 -1.5000000 -1.5000000 -1.5000000 -1.5000000 -1.4812500
##          G          H          I          J          K          L
## -1.4809322 -1.4519231 -1.4083333 -1.3232984 -1.1846154 -1.0369898
##          M          N          O          P          Q          R
## -0.8805970 -0.7500000 -0.6194030 -0.4507576 -0.2992424 -0.1305970
##          S          T          U          V          W          X
##  0.0000000  0.1305970  0.2992424  0.4507576  0.6194030  0.7500000
##          Y          Z         ZA         ZB         ZC         ZD
##  0.8805970  1.0369898  1.1846154  1.3232984  1.4083333  1.4519231
##         ZE         ZF         ZG         ZH         ZI         ZJ
##  1.4809322  1.4812500  1.5000000  1.5000000  1.5000000  1.5000000
##         ZK
##  1.5000000
```

Predictions

```r
predict(tmod)
```

```
##             R1            R2            R3            R4            R5 
## -1.317931e+00 -7.395598e-01 -8.673617e-18  7.395598e-01  1.317931e+00 
##             V1 
## -4.480591e-01
```

```r
predict(tmod, newdata = data_dfm_lbgexample["V1", ])
```

```
##         V1 
## -0.4480591
```

```r
predict(tmod, rescaling = "lbg")
```

```
##          R1          R2          R3          R4          R5          V1 
## -1.58967683 -0.88488724  0.01632248  0.91753220  1.62232179 -0.52967149
```

Predictions with confidence intervals

```r
predict(tmod, se.fit = TRUE, interval = "confidence", rescaling = "mv")
```

```
## Warning in predict.textmodel_wordscores(tmod, se.fit = TRUE, interval =
## "confidence", : More than two reference scores found with MV rescaling;
## using only min, max values.

## $fit
##           fit         lwr         upr
## R1 -1.5000000 -1.51494501 -1.48505499
## R2 -0.8417280 -0.86723325 -0.81622274
## R3  0.0000000 -0.02678045  0.02678045
## R4  0.8417280  0.81622274  0.86723325
## R5  1.5000000  1.48505499  1.51494501
## V1 -0.5099572 -0.53649769 -0.48341678
##
## $se.fit
##          R1          R2          R3          R4          R5          V1
## 0.007625147 0.013013126 0.013663743 0.013013126 0.007625147 0.013541297
```

Plotting a textmodel

```
dfmat <- dfm(data_corpus_irishbudget2010)
refscores <- c(rep(NA, 4), 1, -1, rep(NA, 8))
tmod1 <- textmodel_wordscores(dfmat, y = refscores, smooth = 1)
# plot estimated document positions
textplot_scale1d(predict(tmod1, se.fit = TRUE),
                 groups = docvars(data_corpus_irishbudget2010, "party"))
```

Unsupervised method: Wordfish

- Does not require reference texts
- Estimates a latent position for each text, represented as

$$\theta$$

- Estimates word weights and word "coefficients"
- Includes confidence intervals
- No "predict" step

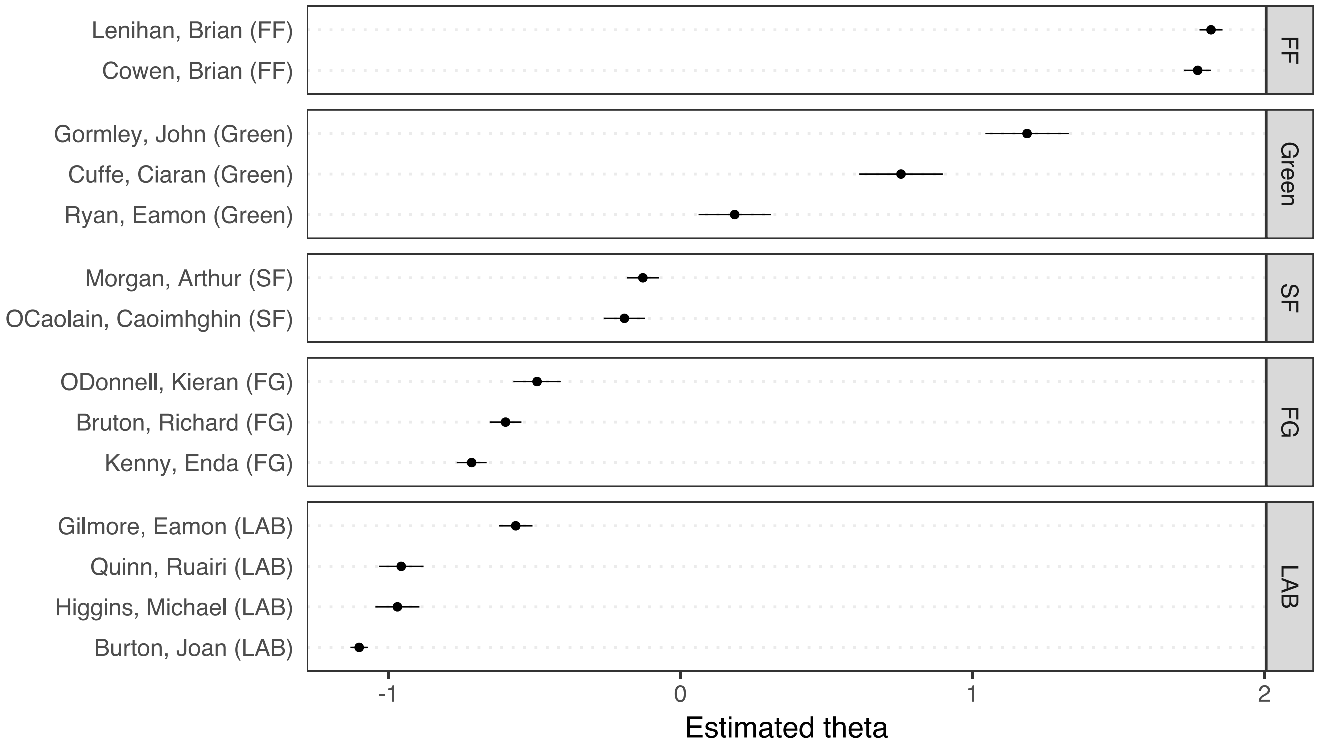
Fitting the Wordfish model

```
dfmat <- dfm(data_corpus_irishbudget2010)
tmod2 <- textmodel_wordfish(dfmat, dir = c(6,5))
summary(tmod2)
```

```
##
## Call:
## textmodel_wordfish.dfm(x = dfmat, dir = c(6, 5))
##
## Estimated Document Positions:
##                              theta      se
## Lenihan, Brian (FF)         1.8170 0.02011
## Bruton, Richard (FG)       -0.5990 0.02775
## Burton, Joan (LAB)         -1.1003 0.01531
## Morgan, Arthur (SF)        -0.1287 0.02801
## Cowen, Brian (FF)           1.7712 0.02343
## Kenny, Enda (FG)           -0.7151 0.02617
## ODonnell, Kieran (FG)      -0.4912 0.04135
## Gilmore, Eamon (LAB)       -0.5642 0.02930
## Higgins, Michael (LAB)     -0.9693 0.03847
## Quinn, Ruairi (LAB)        -0.9562 0.03895
## Gormley, John (Green)       1.1869 0.07281
## Ryan, Eamon (Green)         0.1856 0.06309
## Cuffe, Ciaran (Green)       0.7553 0.07288
## OCaolain, Caoimhghin (SF) -0.1919 0.03624
##
## Estimated Feature Scores:
```

Plotting the wordfish model

```
textplot_scale1d(tmod2, groups = docvars(dfmat, "party"))
```



Other textmodel types

- `textmodel_affinity` - Class affinity maximum likelihood text scaling model
- `textmodel_ca` - Correspondence analysis of a document-feature matrix
- `textmodel_lsa` - Latent Semantic Analysis
- `textmodel_svm` - (soon) fast linear Support Vector Machines for text classification
- `textmodel_nnseq` - (soon) two-layer sequential neural network classifier
- `textmodel_lstm` - (soon) long short-term memory neural network classifier fit to word embeddings

Topic models

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

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

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

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

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

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

What are topic models?

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

Examples of a study using topic models

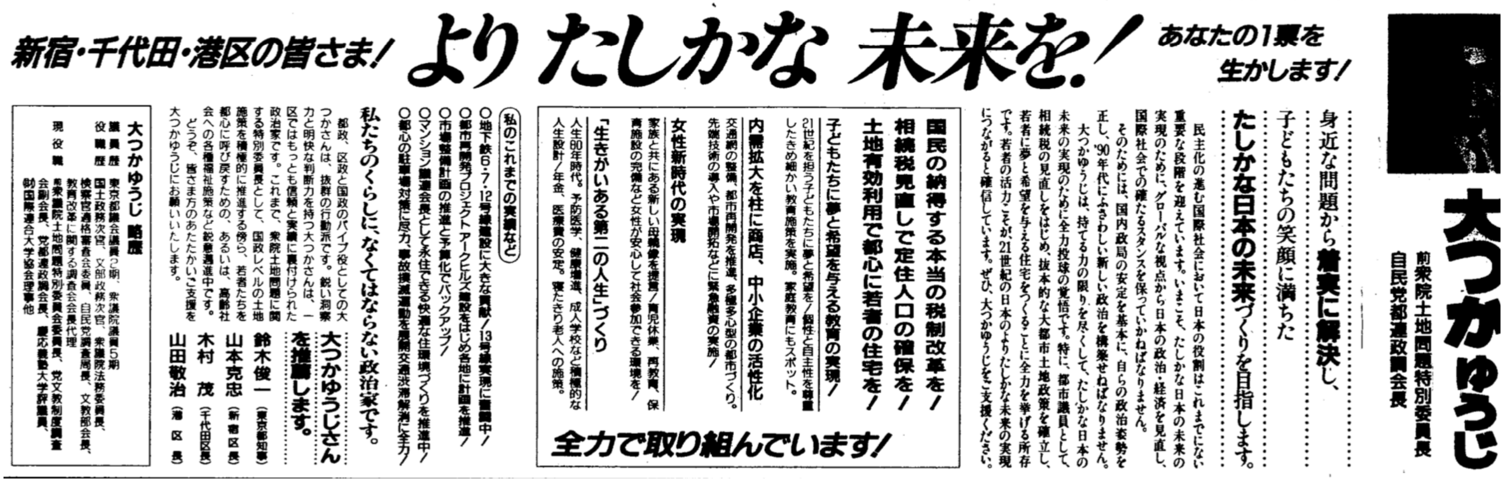
Catalinac (2016)

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

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

Catalinac (2016): Research design

A Japanese candidate manifesto



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

大つかゆうじ

前衆院土地問題特別委員長
自民党都連政調会長

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

民主化の進む国際社会において日本の役割はこれまでにない重要な段階を迎えています。いまこそ、たしかな日本の未来の実現のために、グローバルな視点から日本の政治・経済を見直し、国際社会での確たるスタンスを保っていかねばなりません。

そのためには、国内政局の安定を基本に、自らの政治姿勢を正し、'90年代にふさわしい新しい政治を構築せねばなりません。

大つかゆうじは、持てる力の限りを尽くして、たしかな日本の未来の実現のために全力投球の覚悟です。特に、都市議員として、相続税の見直しをはじめ、抜本的な大都市土地政策を確立し、若者に夢と希望を与える住宅をつくることに全力を挙げる所存です。若者の活力こそが、21世紀の日本のよりたしかな未来の実現につながると確信しています。ぜひ、大つかゆうじをご支援ください。

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

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

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

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

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

全力で取り組んでいます！

私のこれまでの実績など

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

私たちのくらしに、なくてはならない政治家です。

都政、区政と国政のパイプ役としての大つかさんは、抜群の行動派です。鋭い洞察力と明快な判断力を持つ大つかさんは、一区ではもっとも信頼と実績に裏付けられた政治家です。これまで、衆院土地問題に関する特別委員長として、国政レベルの土地施策を積極的に推進する傍ら、若者たちを都心に呼び戻すための、あるいは、高齢社会への各種福祉施策など鋭意邁進中です。

どうぞ、皆さま方のあたたかいご支援を大つかゆうじにお願いいたします。

大つかゆうじさんを推薦します。

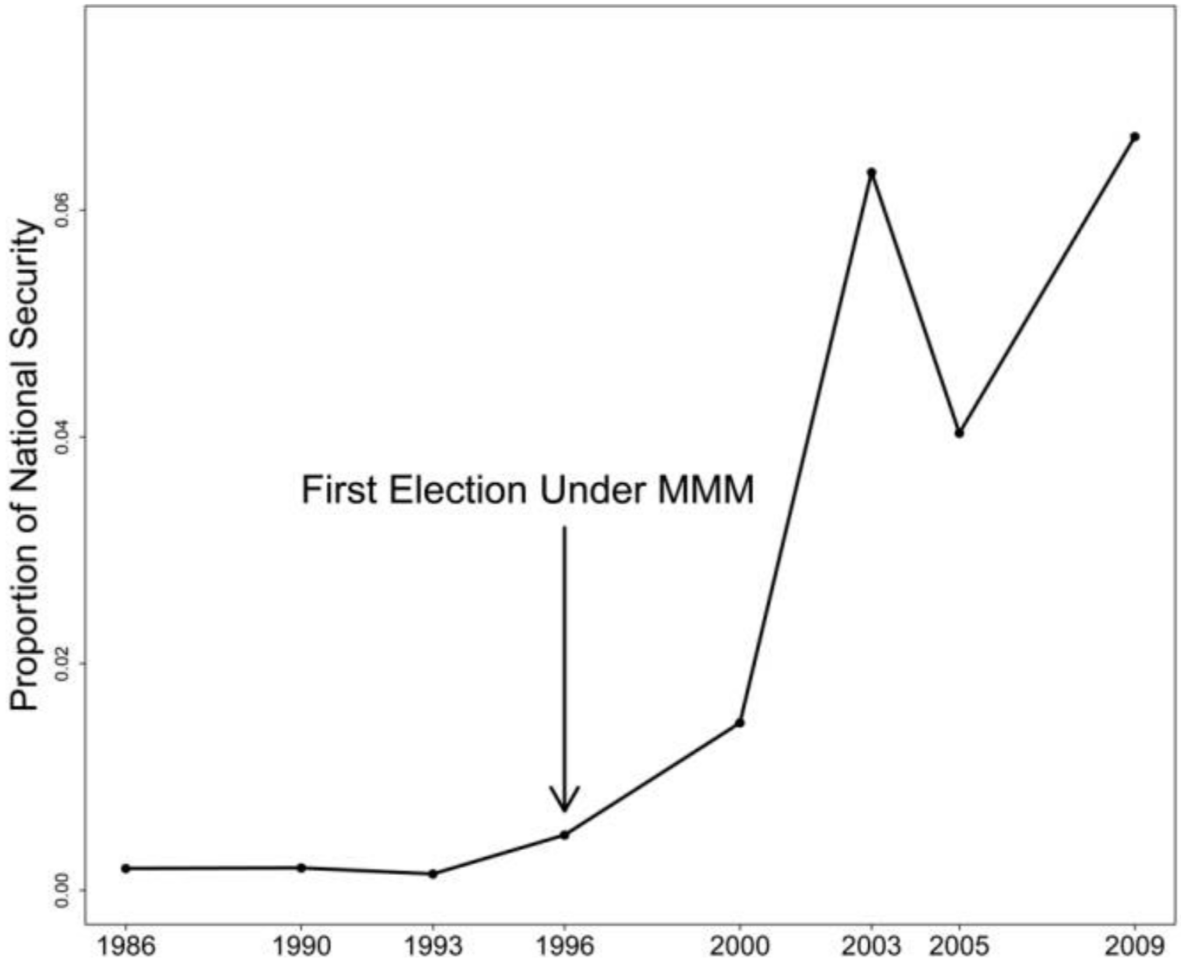
鈴木俊一（東京都知事）
山本克忠（新宿区長）
木村　茂（千代田区長）
山田敬治（港区長）

大つかゆうじ 略歴

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

Catalinac (2016): Results

Catalinac (2016): Results



Catalinac (2016): Results

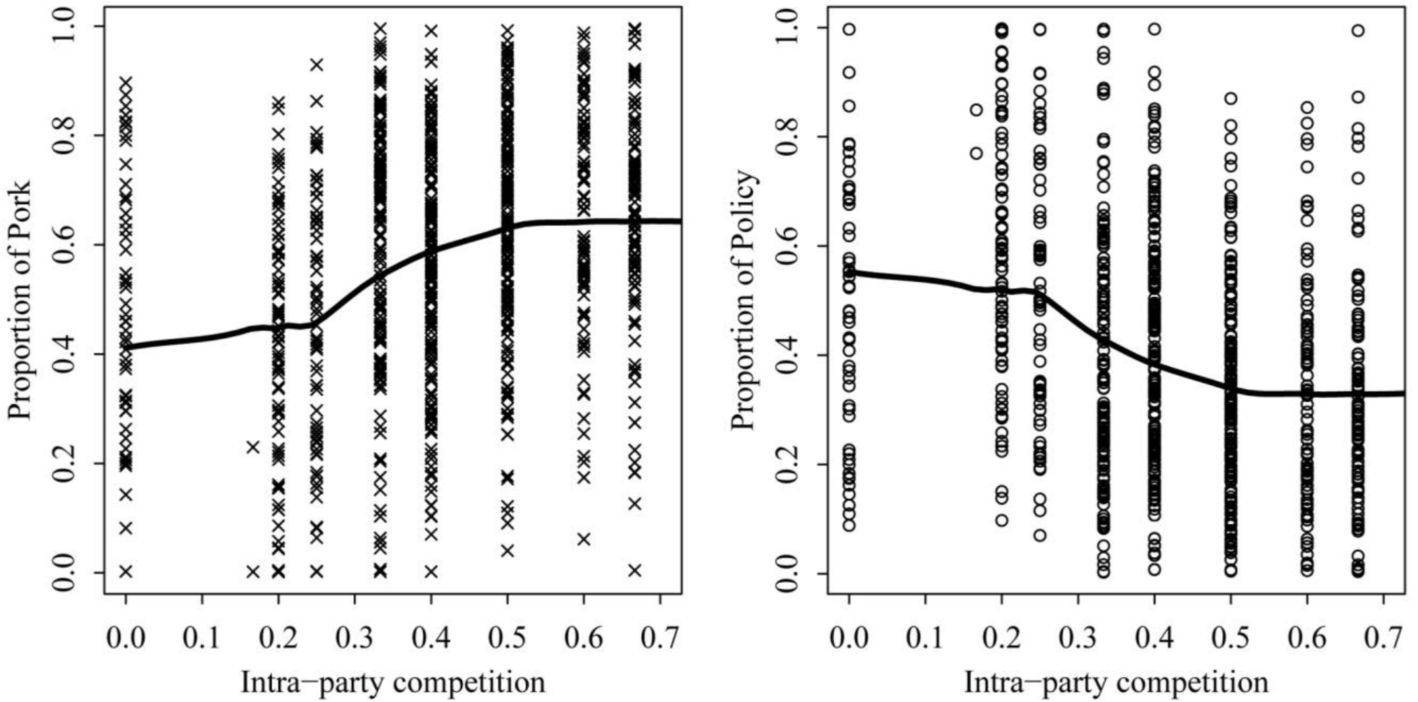


Figure 3. LDP candidates facing higher levels of intraparty competition discussed more pork and less policy than LDP candidates facing lower levels of intraparty competition under SNTV-MMD, where level of intraparty competition is measured by number of LDP opponents relative to district magnitude. The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

Assumptions

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

Assumptions

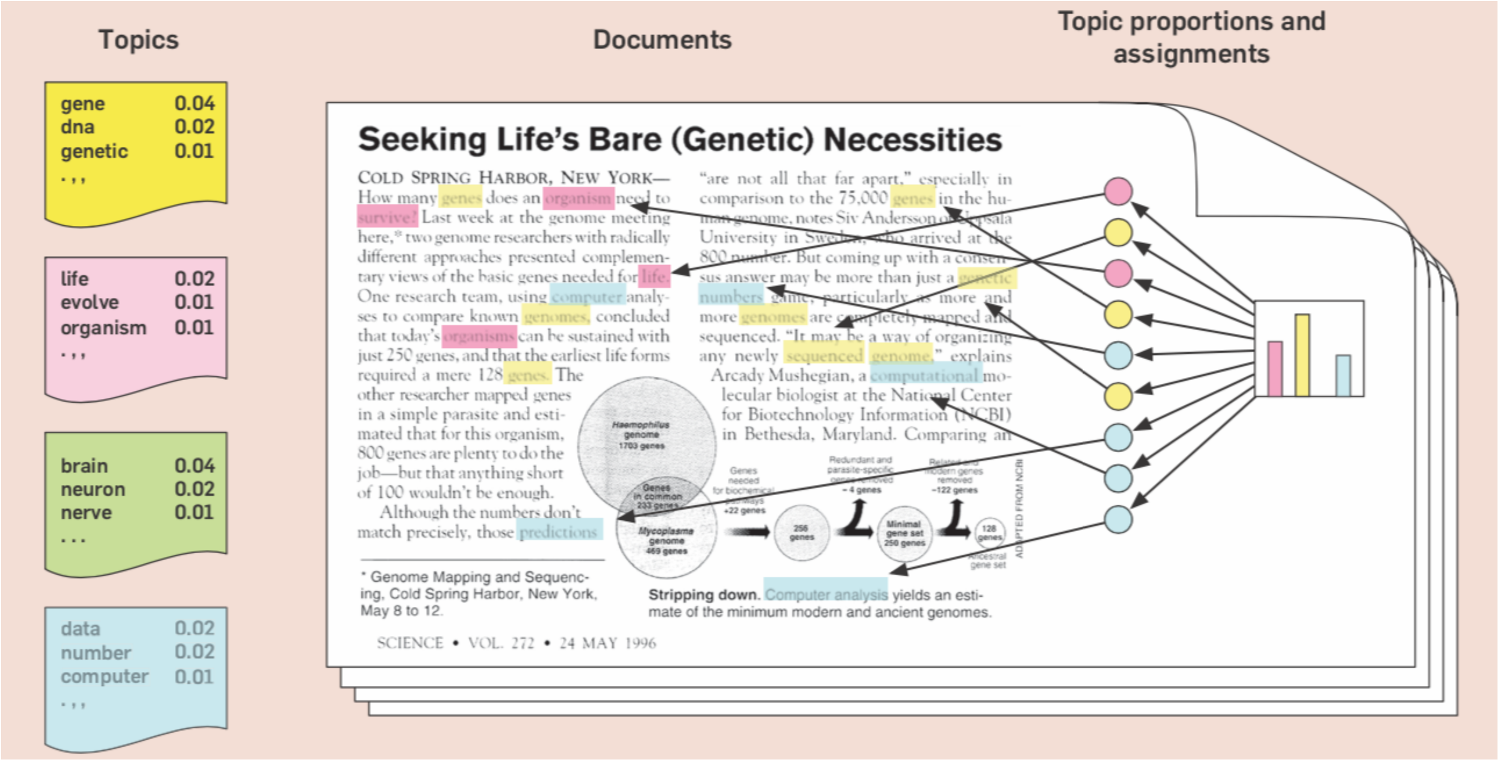
1) Topical Prevalence Matrix $(D\text{x}K)$

$$\boldsymbol{\theta} = \begin{bmatrix} \begin{array}{c|cccc} & Topic1 & Topic2 & \dots & TopicK \\ \hline Doc1 & .2 & .1 & \dots & 0.05 \\ Doc2 & .2 & .1 & \dots & .3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ DocD & 0 & 0 & \dots & .5 \end{array} \end{bmatrix}$$

2) Topical Content Matrix (VxK)

$$\boldsymbol{\beta^T} = \begin{bmatrix} \begin{array}{c|cccc} & Topic1 & Topic2 & \dots & TopicK \\ \hline \text{``}text'' & .02 & .001 & \dots & 0.001 \\ \text{``}data'' & .001 & .02 & \dots & 0.001 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \text{``}analysis'' & .01 & .01 & \dots & 0.0005 \end{array} \end{bmatrix}$$

Example from Blei (2012)



Latent Dirichlet Allocation

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

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

Source: Tutorial by Chris Bail.

Example: CMU 2008 Political Blog Corpus

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

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

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

# download blog posts
dat_blogs <- readtext(url_blogs)

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

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

STM example: create corpus

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

```
## Corpus consisting of 1500 documents, showing 6 documents:
##
##                     Text Types Tokens Sentences  text          docname
##  poliblogs2008.csv.13132   162    322        11 13132 tpm0834400_0.text
##   poliblogs2008.csv.8362   196    324        10  8362 ha0831900_11.text
##   poliblogs2008.csv.8618   246    449        14  8618  ha0834700_3.text
##     poliblogs2008.csv.83   441    935        30    83  at0801200_4.text
##   poliblogs2008.csv.7206   169    266        11  7206  ha0821900_5.text
##  poliblogs2008.csv.11124   416    721        27 11124  tp0829400_9.text
##        rating day blog
##       Liberal 344  tpm
##  Conservative 319   ha
##  Conservative 347   ha
##  Conservative  12   at
##  Conservative 219   ha
##       Liberal 294   tp
##
## Source: /Users/kbenoit/Dropbox (Personal)/GitHub/quanteda/workshops/modules/* on x86_64 by kbenoit
## Created: Wed May 15 08:48:52 2019
## Notes:
```

STM example: create dfm and remove features

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

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

STM example: convert to stm

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

Running the STM

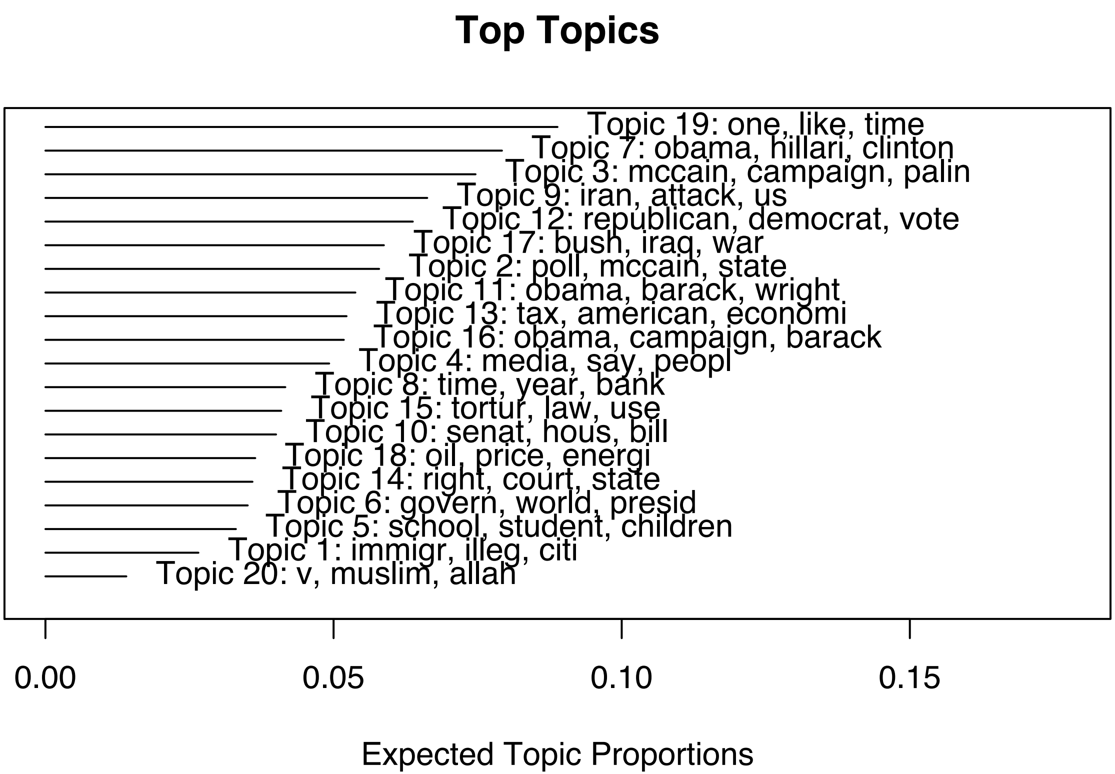
- Use the `stm()` function. `K` specifies the number of topics
- With `K = 0` the **stm** package selects the number of topics automatically (based on algorithm developed by Lee and Mimno (2014))
- Specify covariates with `prevalence`

Running the STM

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

```
## Beginning Spectral Initialization
##      Calculating the gram matrix...
##      Finding anchor words...
##      ....................
##      Recovering initialization...
##      ........................................
## Initialization complete.
## ....................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 1 (approx. per word bound = -7.348)
## ....................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
## Completing Iteration 2 (approx. per word bound = -7.135, relative change = 2.902e-02)
## ....................................................................................................
## Completed E-Step (0 seconds).
## Completed M-Step.
```

```
plot(stm_object)
```



Estimate effects

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

Identify topics of interest

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

```
## Topic 3 Top Words:
##       Highest Prob: mccain, campaign, palin, john, obama, said, say
##       FREX: mccain, biden, palin, sarah, john, campaign, joe
##       Lift: frederick, scheunemann, cindi, couric, sarah, pin, palin
##       Score: mccain, palin, frederick, biden, campaign, obama, sarah
## Topic 6 Top Words:
##       Highest Prob: govern, world, presid, countri, elect, china, said
##       FREX: chavez, mugab, china, britain, chines, british, colombia
##       Lift: colombian, czech, mugab, tsvangirai, chavez, chines, hugo
##       Score: mugab, czech, chavez, irish, china, zimbabw, colombia
```

Identify topics of interest

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

```
## Topic 16 Top Words:
##       Highest Prob: obama, campaign, barack, state, report, senat, group
##       FREX: blagojevich, president-elect, registr, acorn, appoint, transit, staff
##       Lift: blagojevich, carolin, spakovski, blago, fitzgerald, jarrett, emanuel
##       Score: blagojevich, spakovski, obama, registr, acorn, fitzgerald, jarrett
## Topic 18 Top Words:
##       Highest Prob: oil, price, energi, state, $, compani, tax
##       FREX: oil, drill, price, energi, gas, gift, product
##       Lift: interior, barrel, suv, drill, offshor, oil, pipelin
##       Score: oil, interior, price, energi, drill, tax, gas
```

Meanings of Output

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

Find representative documents of topic

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

Estimate effect of covariates

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

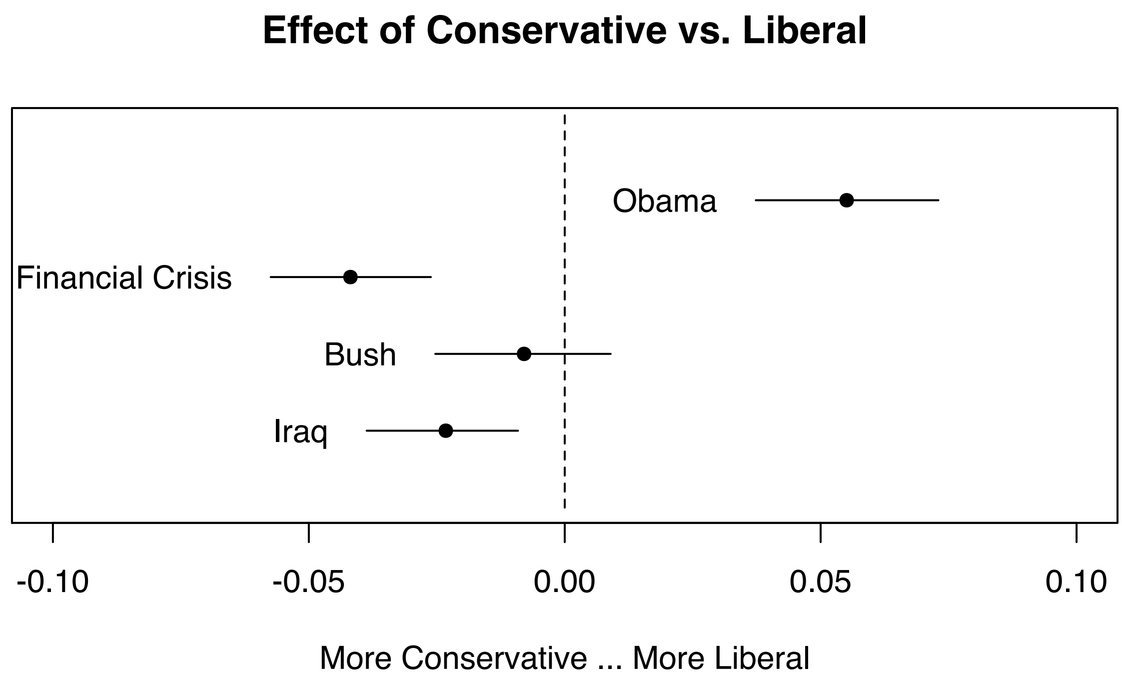
```
summary(stm_effects)
```

Plot influence of discrete variable

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

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

```
plot_effect_libcon
```

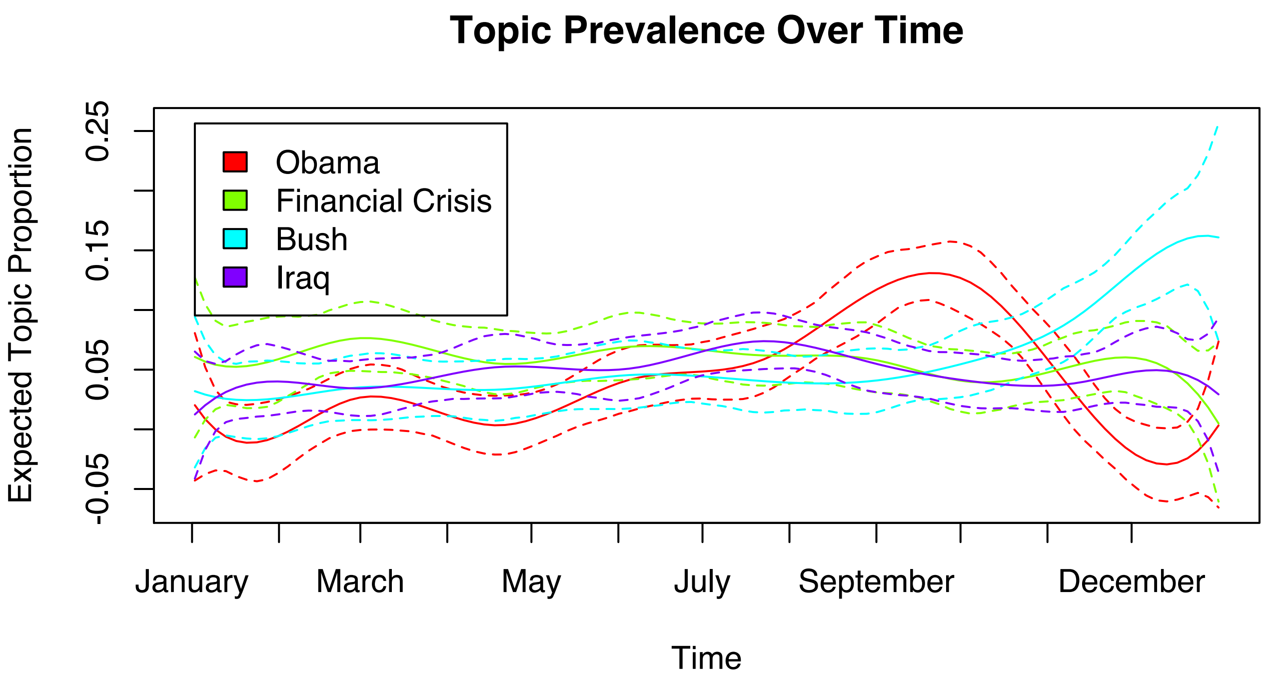


Plot effect of continuous variable

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

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

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

```
plot_effect_time
```



Create (somewhat) better/tidyier plots

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

References and useful descriptions

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

EXTRAS

Sentiment analysis

Why are we analyzing Irish budget speeches?

Irish Wince as a Budget Proposal Cuts to the Bone

By SARAH LYALL DEC. 9, 2009

Sentiment analysis

Sentiment analysis using the Lexicoder Sentiment Dictionary (`data_dictionary_LSD2015`)

```
summary(data_dictionary_LSD2015)
```

```
##              Length Class  Mode
## negative     2858   -none- character
## positive     1709   -none- character
## neg_positive 1721   -none- character
## neg_negative 2860   -none- character
```

```
docvars(data_corpus_irishbudget2010, "gov_opp") <-
    ifelse(docvars(data_corpus_irishbudget2010, "party") %in%
               c("FF", "Green"),
           "Government", "Opposition")

# tokenize and apply dictionary
toks_dict <- data_corpus_irishbudget2010 %>%
    tokens() %>%
    tokens_lookup(dictionary = data_dictionary_LSD2015)

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

Sentiment Analysis

```
print(dfmat_dict)
```

```
## Document-feature matrix of: 14 documents, 4 features (12.5% sparse).
## 14 x 4 sparse Matrix of class "dfm"
##                              features
## docs                           negative positive neg_positive neg_negative
##   Lenihan, Brian (FF)               188      397            2            1
##   Bruton, Richard (FG)              163      147            5            2
##   Burton, Joan (LAB)                225      266            8            3
##   Morgan, Arthur (SF)               260      249            5            2
##   Cowen, Brian (FF)                 150      368            1            2
##   Kenny, Enda (FG)                  104      146            3            3
##   ODonnell, Kieran (FG)              49       84            4            0
##   Gilmore, Eamon (LAB)              164      176            3            0
##   Higgins, Michael (LAB)             37       42            1            0
##   Quinn, Ruairi (LAB)                34       40            1            1
##   Gormley, John (Green)              17       56            0            0
##   Ryan, Eamon (Green)                24       78            1            2
##   Cuffe, Ciaran (Green)              38       56            0            0
##   OCaolain, Caoimhghin (SF)         154      145            1            1
```

Convert to a data frame using `convert()`

```
dict_output <- convert(dfmat_dict, to = "data.frame")
```

Estimate sentiment

```
dict_output$sent_score <- log((dict_output$positive +
                                   dict_output$neg_negative + 0.5) /
                              (dict_output$negative +
                                   dict_output$neg_positive+ 0.5))

dict_output <- cbind(dict_output, docvars(data_corpus_irishbudget2010))
```

Plotting with ggplot2

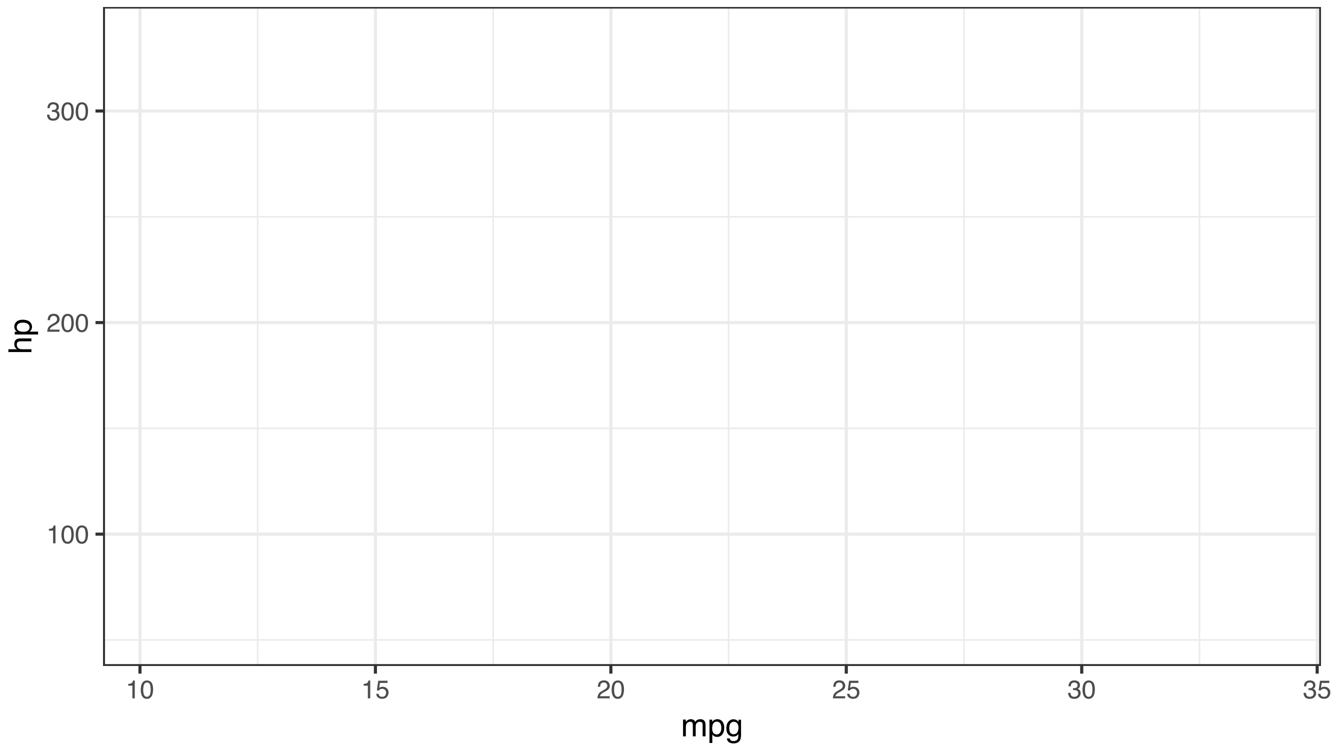
- Specify `data` (a data.frame)
- Specify the aestetics (x-axis, y-axis, colours, shapes etc.)
- Choose a geometric objects (e.g. scatterplot, boxplot)

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

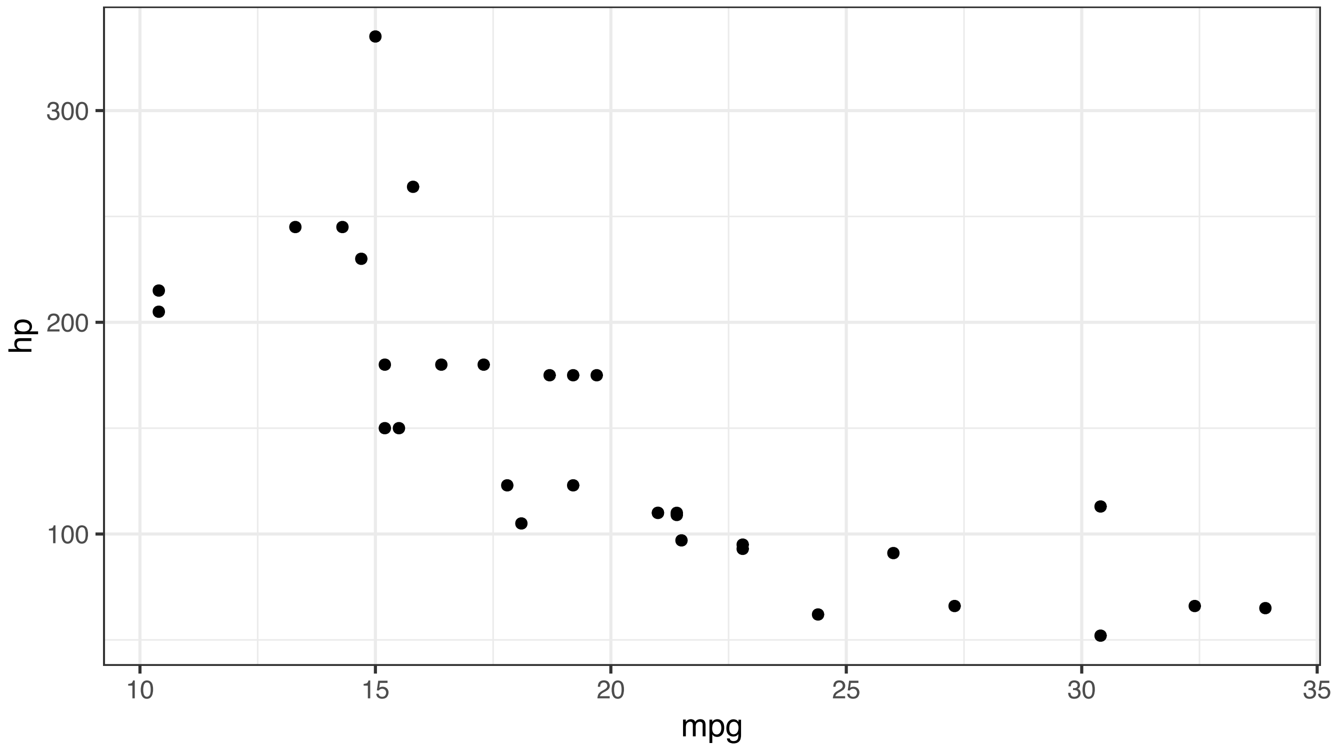
```
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

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

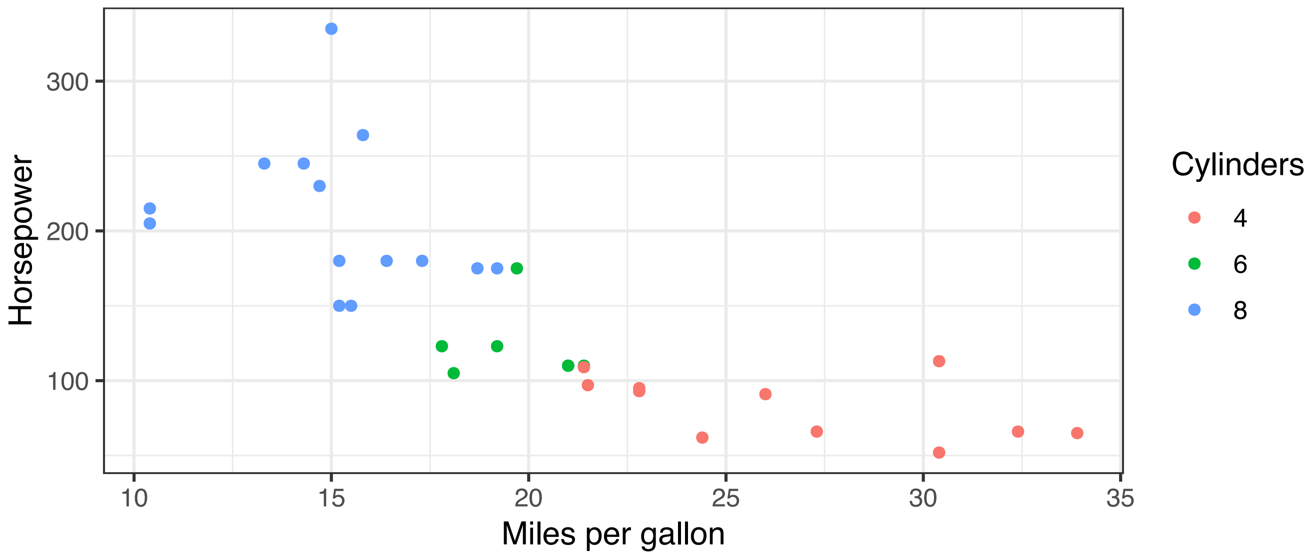
```r
myplot
```



```
myplot +
  geom_point()
```



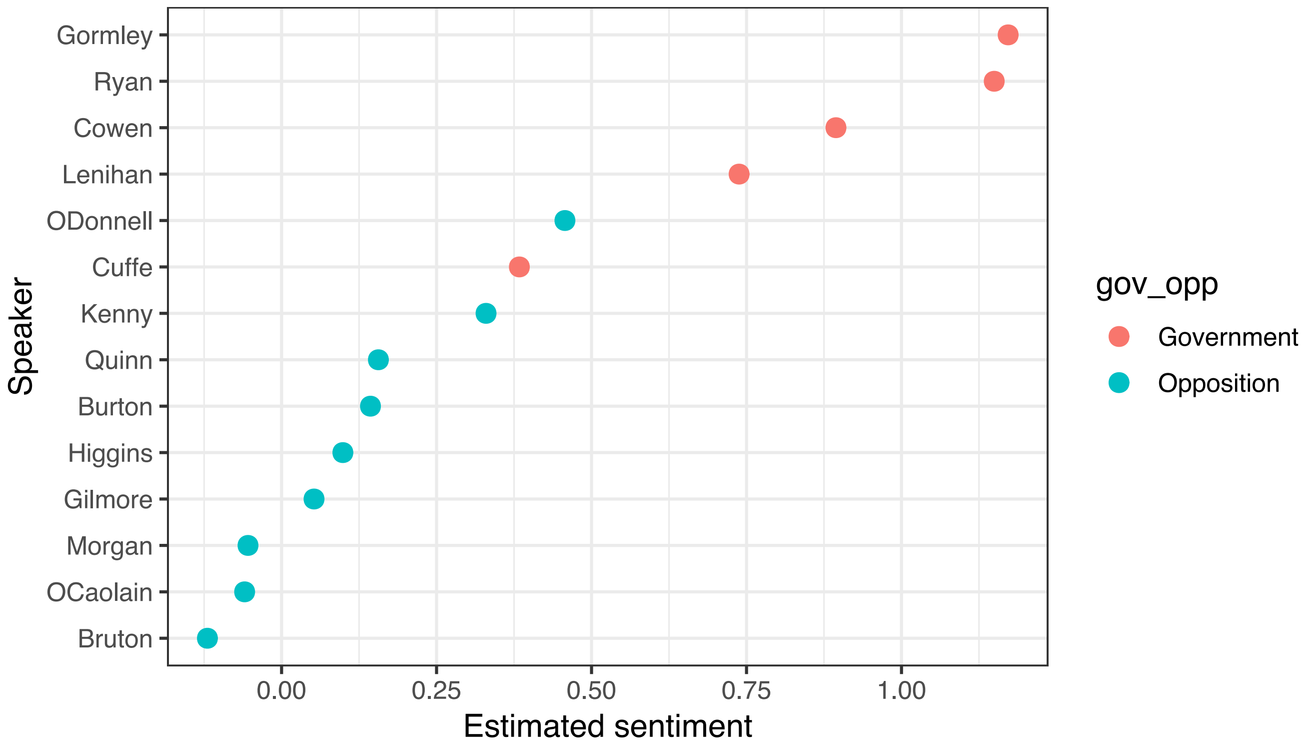
```r
myplot +
  geom_point(aes(colour = factor(cyl))) +
  scale_colour_discrete(name = "Cylinders") +
  scale_shape_discrete(name = "Cylinders") +
  labs(x = "Miles per gallon",
       y = "Horsepower")
```



Plot sentiment

```
tplot_sent <- ggplot(dict_output,
                        aes(x = reorder(name, sent_score),
                        y = sent_score,
                        colour = gov_opp)) +
    geom_point(size = 3) +
    coord_flip() +
    labs(x = "Speaker", y = "Estimated sentiment")
```

```
tplot_sent
```



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