Introduction to Quantitative Text Analysis (QTA)

Day Two Lab

Brian Fogarty

15 May 2018

Whereas last week's lab materials were about getting our text data set-up, etc., this week is about different statistical analysis options that are available with `Quanteda`.

1. Set-Up

This is about just getting set-up. More details at https://tutorials.quanteda.io/introduction/install/. In today's lab, we are going to work with the full inaugural speeches corpus and the 2010 Irish budget speeches data that is part of the `Quanteda` package.

```
setwd("C:/QTA/Day Two")
getwd()
```

```
## [1] "C:/QTA/Day Two"
```

```
.libPaths("C:/Program Files/R/R-3.4.2/library")
.libPaths()
```

```
## [1] "C:/Program Files/R/R-3.4.2/library"
```

```
library(quanteda)
```

```
## Warning: package 'quanteda' was built under R version 3.4.4
```

```
## Package version: 1.2.0
```

```
## Parallel computing: 2 of 4 threads used.
```

```
## See https://quanteda.io for tutorials and examples.
```

```
##
## Attaching package: 'quanteda'
```

```
## The following object is masked from 'package:utils':
##
##     View
```

```
library(readtext)

full_inaug_corp <- corpus(data_corpus_inaugural)
ndoc(full_inaug_corp)
```

```
## [1] 58
```

```
summary(full_inaug_corp)
```

```
## Corpus consisting of 58 documents:
##
##             Text Types Tokens Sentences Year  President       FirstName
##  1789-Washington   625   1538        23 1789 Washington          George
##  1793-Washington    96    147         4 1793 Washington          George
```

```
##       1797-Adams  826  2578        37 1797      Adams            John
##   1801-Jefferson  717  1927        41 1801  Jefferson          Thomas
##   1805-Jefferson  804  2381        45 1805  Jefferson          Thomas
##     1809-Madison  535  1263        21 1809    Madison           James
##     1813-Madison  541  1302        33 1813    Madison           James
##      1817-Monroe 1040  3680       121 1817     Monroe           James
##      1821-Monroe 1259  4886       129 1821     Monroe           James
##       1825-Adams 1003  3152        74 1825      Adams     John Quincy
##     1829-Jackson  517  1210        25 1829    Jackson          Andrew
##     1833-Jackson  499  1269        29 1833    Jackson          Andrew
##    1837-VanBuren 1315  4165        95 1837  Van Buren          Martin
##    1841-Harrison 1896  9144       210 1841   Harrison   William Henry
##        1845-Polk 1334  5193       153 1845       Polk      James Knox
##      1849-Taylor  496  1179        22 1849     Taylor         Zachary
##      1853-Pierce 1165  3641       104 1853     Pierce        Franklin
##    1857-Buchanan  945  3086        89 1857   Buchanan           James
##     1861-Lincoln 1075  4006       135 1861    Lincoln         Abraham
##     1865-Lincoln  360   776        26 1865    Lincoln         Abraham
##       1869-Grant  485  1235        40 1869      Grant      Ulysses S.
##       1873-Grant  552  1475        43 1873      Grant      Ulysses S.
##       1877-Hayes  831  2716        59 1877      Hayes   Rutherford B.
##    1881-Garfield 1021  3212       111 1881   Garfield        James A.
##   1885-Cleveland  676  1820        44 1885  Cleveland          Grover
##    1889-Harrison 1352  4722       157 1889   Harrison        Benjamin
##   1893-Cleveland  821  2125        58 1893  Cleveland          Grover
##    1897-McKinley 1232  4361       130 1897   McKinley         William
##    1901-McKinley  854  2437       100 1901   McKinley         William
##   1905-Roosevelt  404  1079        33 1905  Roosevelt        Theodore
##        1909-Taft 1437  5822       159 1909       Taft  William Howard
##      1913-Wilson  658  1882        68 1913     Wilson         Woodrow
##      1917-Wilson  549  1656        59 1917     Wilson         Woodrow
##     1921-Harding 1169  3721       148 1921    Harding        Warren G.
##    1925-Coolidge 1220  4440       196 1925   Coolidge          Calvin
##      1929-Hoover 1090  3865       158 1929     Hoover         Herbert
##   1933-Roosevelt  743  2062        85 1933  Roosevelt     Franklin D.
##   1937-Roosevelt  725  1997        96 1937  Roosevelt     Franklin D.
##   1941-Roosevelt  526  1544        68 1941  Roosevelt     Franklin D.
##   1945-Roosevelt  275   647        26 1945  Roosevelt     Franklin D.
##      1949-Truman  781  2513       116 1949     Truman        Harry S.
##  1953-Eisenhower  900  2757       119 1953 Eisenhower       Dwight D.
##  1957-Eisenhower  621  1931        92 1957 Eisenhower       Dwight D.
##     1961-Kennedy  566  1566        52 1961    Kennedy         John F.
##     1965-Johnson  568  1723        93 1965    Johnson   Lyndon Baines
##       1969-Nixon  743  2437       103 1969      Nixon Richard Milhous
##       1973-Nixon  544  2012        68 1973      Nixon Richard Milhous
##      1977-Carter  527  1376        52 1977     Carter           Jimmy
##      1981-Reagan  902  2790       128 1981     Reagan          Ronald
##      1985-Reagan  925  2921       123 1985     Reagan          Ronald
##        1989-Bush  795  2681       141 1989       Bush          George
##     1993-Clinton  642  1833        81 1993    Clinton            Bill
##     1997-Clinton  773  2449       111 1997    Clinton            Bill
##        2001-Bush  621  1808        97 2001       Bush       George W.
##        2005-Bush  773  2319       100 2005       Bush       George W.
##       2009-Obama  938  2711       110 2009      Obama          Barack
```

```
##       2013-Obama   814   2317         88 2013      Obama          Barack
##       2017-Trump   582   1660         88 2017      Trump       Donald J.
##
## Source: Gerhard Peters and John T. Woolley. The American Presidency Project.
## Created: Tue Jun 13 14:51:47 2017
## Notes: http://www.presidency.ucsb.edu/inaugurals.php
```

```
toks <- tokens(full_inaug_corp, remove_punct=TRUE)

toks1 <- tokens_select(toks, stopwords('english'), selection='remove')

dfm1 <- dfm(toks1)
```

2. Lexical Diversity

There are times when we might be interested in the complexity of the words used in text. There are a ton of measures here, but they all roughly do the same thing - comparing the number of unique words (tokens) and total words (tokens).

```
lexdiv <- textstat_lexdiv(dfm1)
head(lexdiv, 5)
```

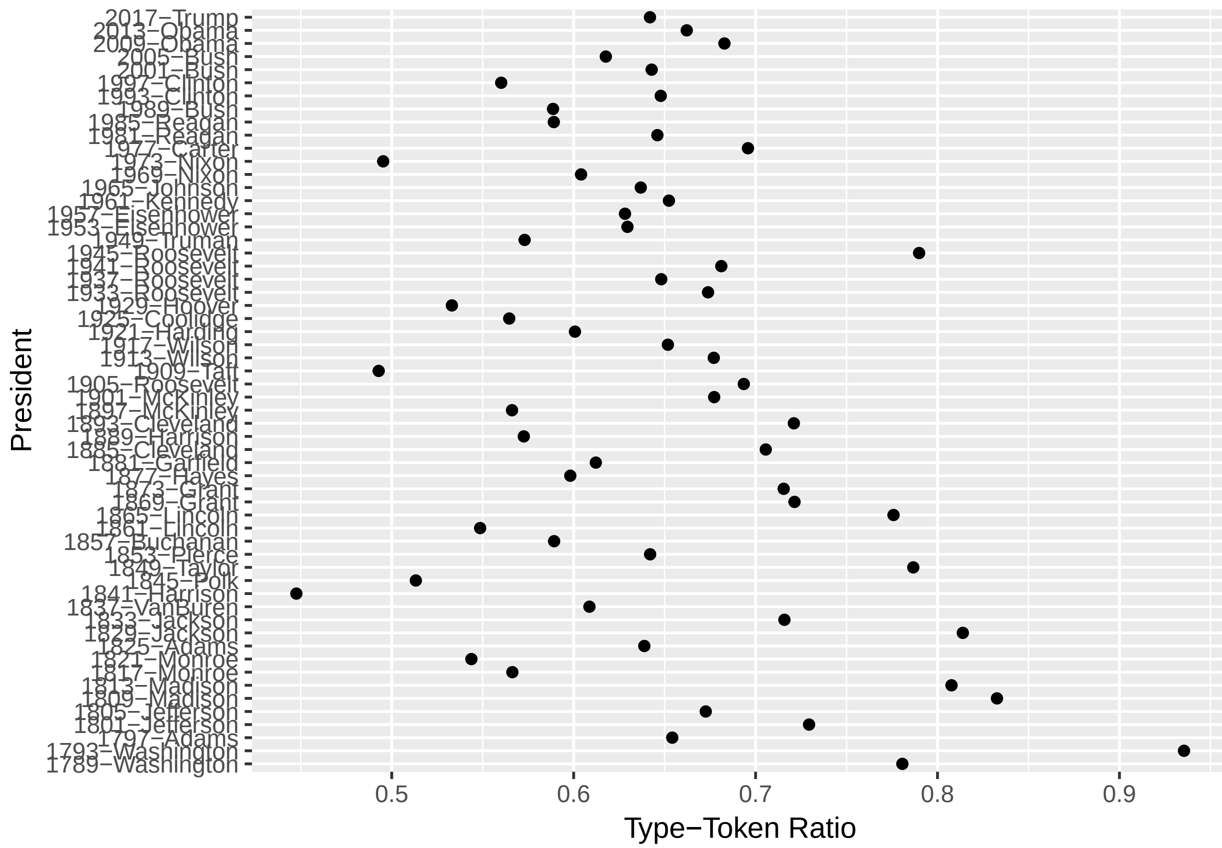
```
##          document       TTR         C         R      CTTR         U
## 1 1789-Washington 0.7806748 0.9617909 19.933978 14.095451  73.65389
## 2 1793-Washington 0.9354839 0.9838408  7.366007  5.208554 110.92051
## 3      1797-Adams 0.6542056 0.9391673 21.399624 15.131820  49.79857
## 4  1801-Jefferson 0.7293973 0.9529101 20.797418 14.705995  61.79862
## 5  1805-Jefferson 0.6726014 0.9426767 21.386186 15.122317  52.41763
##           S      Maas     lgV0    lgeV0
## 1 0.9623481 0.1165204 9.886289 22.76402
## 2 0.9720826 0.0949498 9.849038 22.67825
## 3 0.9433742 0.1417071 8.283634 19.07377
## 4 0.9548442 0.1272069 9.144363 21.05567
## 5 0.9463441 0.1381215 8.488001 19.54435
```

```
tail(lexdiv, 5)
```

```
##      document       TTR         C        R     CTTR        U         S
## 54  2001-Bush 0.6428571 0.9337026 18.00000 12.72792 43.65658 0.9354530
## 55  2005-Bush 0.6176753 0.9306568 19.92900 14.09193 43.51472 0.9349294
## 56 2009-Obama 0.6828645 0.9460250 23.38748 16.53745 56.86513 0.9505226
## 57 2013-Obama 0.6621622 0.9406254 21.31298 15.07055 50.78539 0.9445418
## 58 2017-Trump 0.6419580 0.9325604 17.16563 12.13793 42.32387 0.9334292
##         Maas     lgV0    lgeV0
## 54 0.1513475 7.547647 17.37910
## 55 0.1515940 7.674943 17.67221
## 56 0.1326102 8.959235 20.62940
## 57 0.1403236 8.355738 19.23980
## 58 0.1537118 7.373155 16.97732
```

We can also plot lexical diversity by selecting one of the measures. Below we do it with `Type-Token Ratio`, which is simple the number of unique words divided by the total number of words. **(Remember that if you are using a Mac, just skip the `x11()` function.)**

```r
library(ggplot2)

x11()
ggplot(data=lexdiv, aes(x=document, y=TTR)) +
  geom_point() +
  coord_flip() +
  labs(x="President",y="Type-Token Ratio")
```



3. Keyness

Another somewhat simple analysis that is available is **keyness**. The basic idea is to compare words between a target and a reference group. Below we first compare pre-war (WWII), our reference group, and post-war, our target group, speeches.

```r
period <- ifelse(docvars(data_corpus_inaugural, "Year") < 1945, "pre-war", "post-war")

dfm2 <- dfm(toks1, groups = period)
head(dfm2)
```

```
## Document-feature matrix of: 2 documents, 9,205 features (35.4% sparse).
```

```r
head(result <- textstat_keyness(dfm2), 10)
```

```
##      feature      chi2 p n_target n_reference
## 1         us 186.34621 0      262         216
```

```
## 2     america 176.58663 0      130          54
## 3       world 175.24439 0      188         123
## 4   americans 150.53636 0       67           7
## 5         new 141.37882 0      150          97
## 6     freedom 137.30596 0      121          64
## 7       today 128.77885 0       76          21
## 8         let 111.45125 0      100          54
## 9    together 104.88823 0       64          19
## 10 america's  93.97685 0       35           0
```

```r
tail(result <- textstat_keyness(dfm2), 10)
```

```
##           feature      chi2            p n_target n_reference
## 9196        state -25.85907 3.672751e-07        6         103
## 9197       powers -27.49893 1.571812e-07        4          97
## 9198    executive -28.42150 9.757587e-08        3          94
## 9199          may -30.12530 4.050120e-08       47         291
## 9200       policy -33.50265 7.116694e-09        1          96
## 9201   government -43.19681 4.950129e-11       84         480
## 9202         upon -52.21073 4.984901e-13       39         332
## 9203       public -56.19816 6.550316e-14       11         213
## 9204       states -59.43006 1.265654e-14       28         305
## 9205 constitution -61.37027 4.773959e-15        6         200
```

The two right-hand columns are the number of times the words appear between the two groups. The Chi-Square value is signed positively if the words appear greater than expected and negative if vice-versa. The *p*-values seem essentially worthless since they seem like they are always 'statistically significant'.

There are several different options for the statistics reported with `textstat_keyness()`. Let's look at the commonly utilised odds-ratio by including `measure="exact"`. The one problem for very sparse dfms is that the comparisons may be against 0, which makes for infinitely greater odds or 0 odds.

```r
head(result1 <- textstat_keyness(dfm2, measure="exact"), 10)
```

```
##         feature  or            p n_target n_reference
## 1     america's Inf 1.478186e-20       35           0
## 2     faintness Inf 2.714737e-01        1           0
## 3   schoolmaster Inf 2.714737e-01        1           0
## 4            dr Inf 7.369489e-02        2           0
## 5       peabody Inf 2.714737e-01        1           0
## 6     untroubled Inf 2.714737e-01        1           0
## 7      smoothly Inf 2.714737e-01        1           0
## 8      downward Inf 2.714737e-01        1           0
## 9         trend Inf 2.000456e-02        3           0
## 10        peaks Inf 7.369489e-02        2           0
```

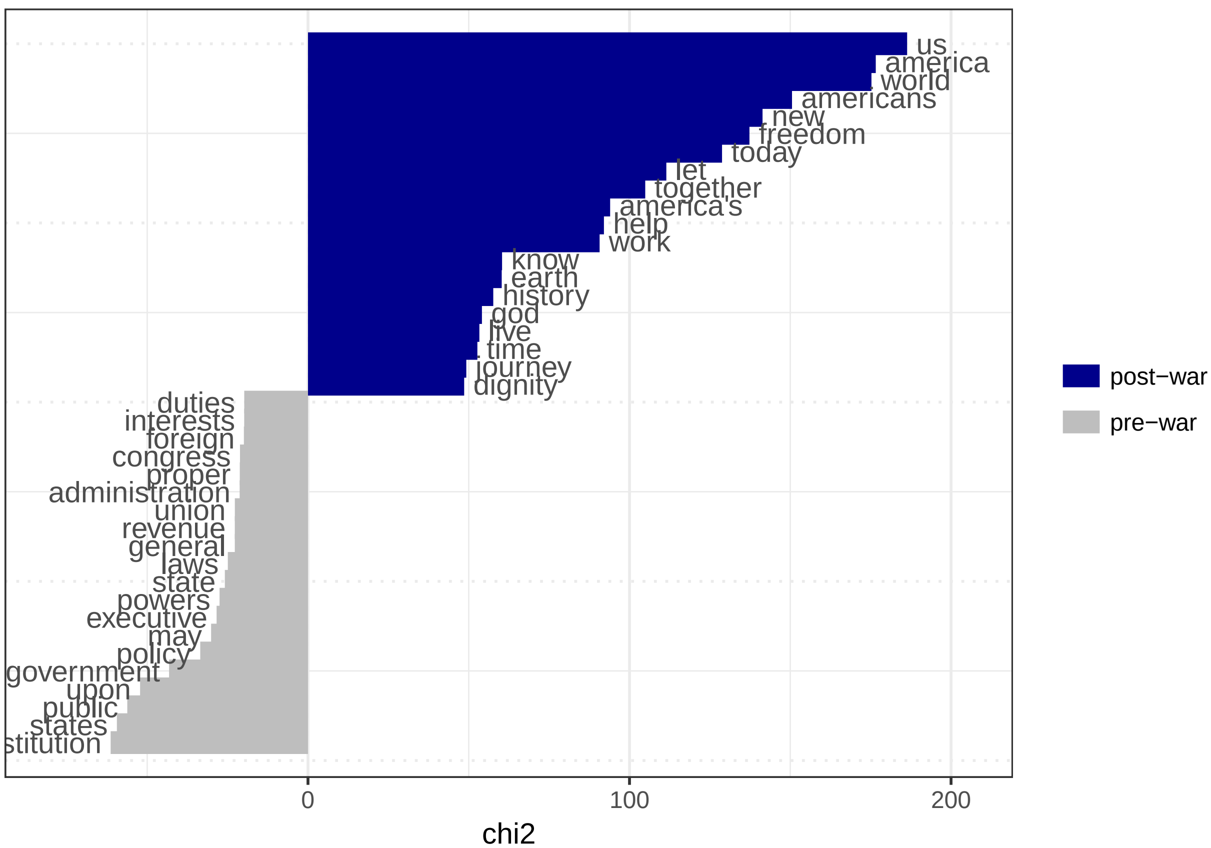
```r
tail(result1 <- textstat_keyness(dfm2, measure="exact"), 10)
```

```
##           feature or p n_target n_reference
## 9196    unnoticed  0 1        0           1
## 9197           48  0 1        0           1
## 9198     counties  0 1        0           1
## 9199        towns  0 1        0           1
## 9200     prophecy  0 1        0           1
## 9201         1941  0 1        0           1
## 9202    smothered  0 1        0           1
## 9203       strove  0 1        0           1
```

```
## 9204    valiantly  0 1        0            1
## 9205 triumphantly  0 1        0            1
```

Finally, we can do a plot using `textplot_keyness()` for keyness of the words with the largest chi-square values for the target and reference groups.

```
x11()
textplot_keyness(result)
```



4. Document Similarity

We may be interested in the similarity of documents in documents. To get basic correlations between documents, we can use the `textstat_simil()` function. Since the correlation matrix will be huge, let's subset the corpus to only include speeches since 1992.

```
presDfm <- dfm(corpus_subset(data_corpus_inaugural, Year > 1992),
               remove = stopwords("english"), stem = TRUE, remove_punct = TRUE)

simil1 <- textstat_simil(presDfm, margin="documents")
simil1
```

```
##              1993-Clinton 1997-Clinton 2001-Bush 2005-Bush 2009-Obama
## 1997-Clinton    0.6271129
## 2001-Bush       0.4734439    0.5327868
## 2005-Bush       0.4338601    0.4289271 0.5344415
## 2009-Obama      0.5707623    0.6026942 0.5241995 0.4330978
```

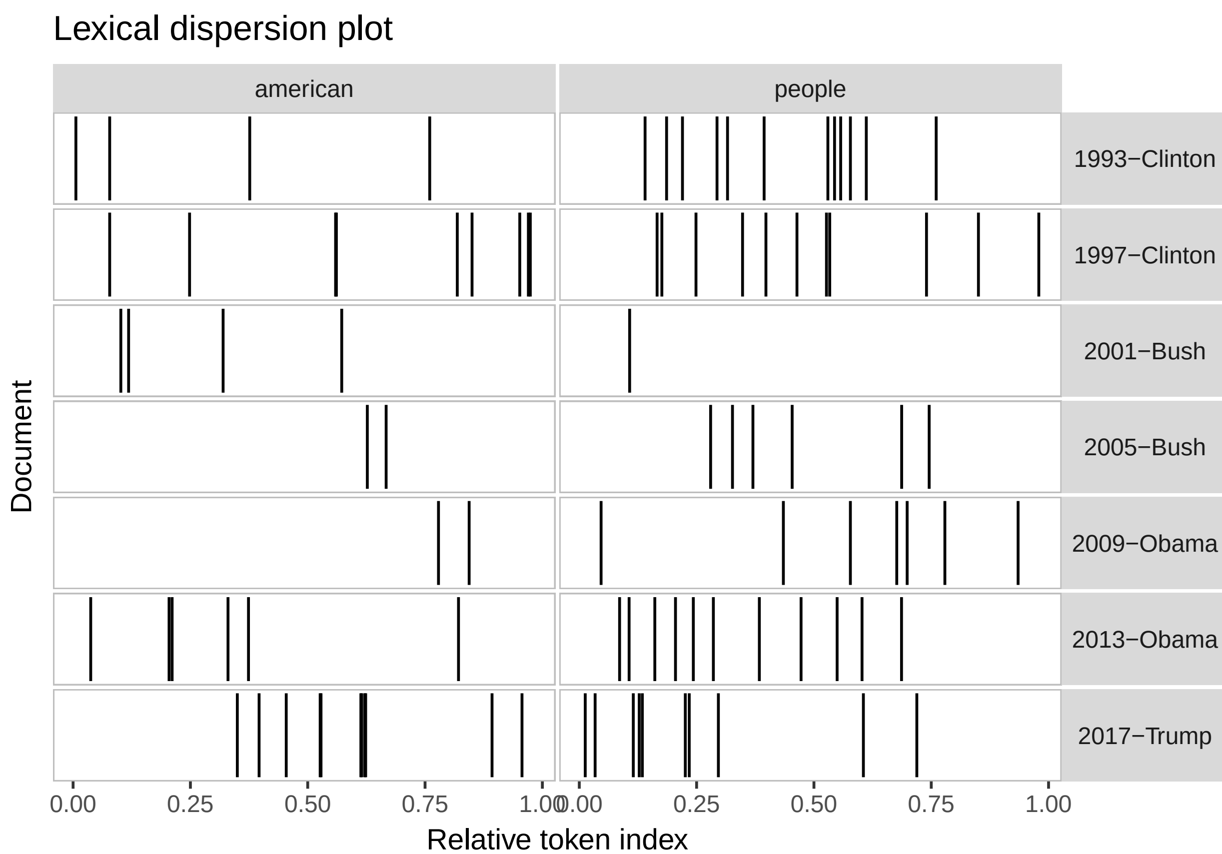
```
## 2013-Obama      0.5725041    0.5916516 0.5523905 0.5137096  0.6103693
## 2017-Trump      0.4967910    0.4989669 0.4672634 0.4739241  0.4377484
##             2013-Obama
## 1997-Clinton
## 2001-Bush
## 2005-Bush
## 2009-Obama
## 2013-Obama
## 2017-Trump    0.4414144
```

This gives the correlations between each of the speeches since 1992. As with most of the `Quanteda` functions, there are a bunch of different options for statistics to examine.

5. Locations of Keywords in Documents

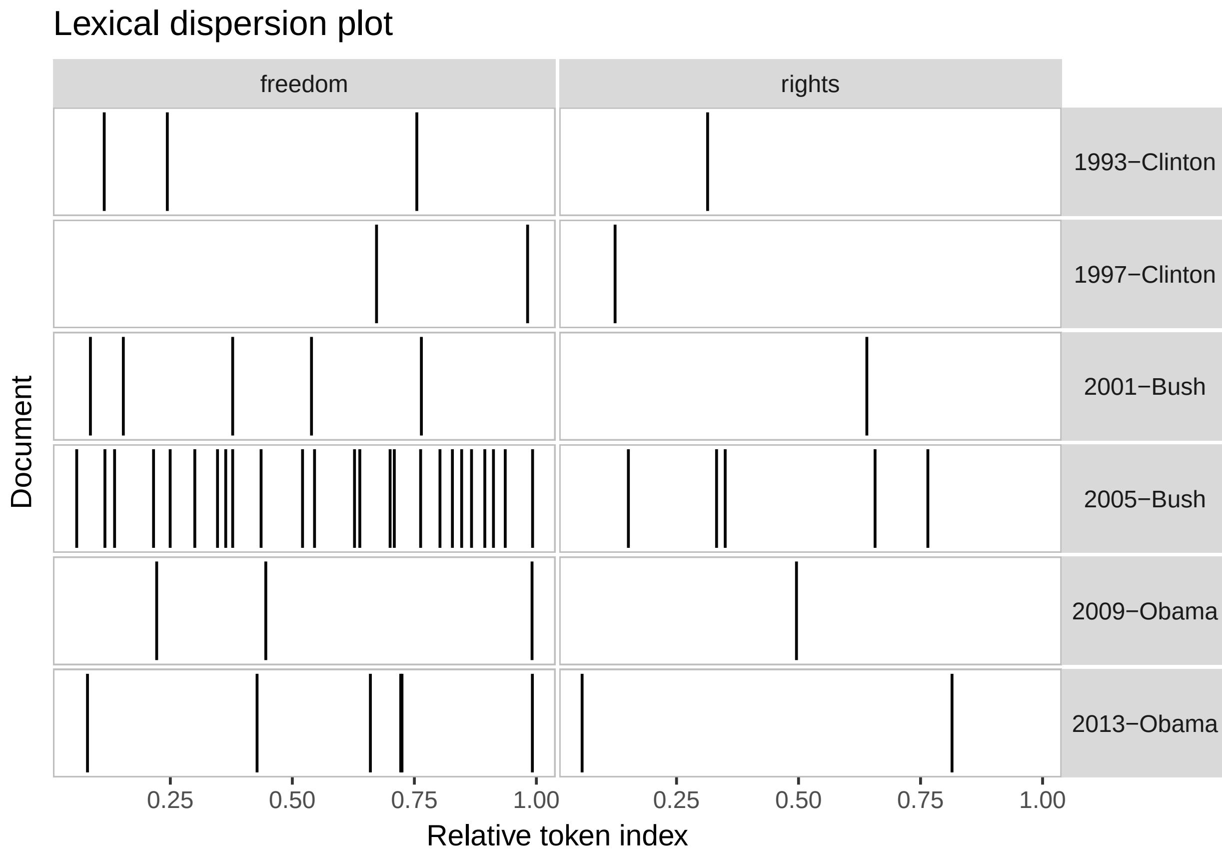
If we are interested in where certain words (i.e., keywords) are located in documents, we can use the `textplot_xray()` function. Because are considering the locations of certain words we need to work with the corpus and not the dfm, and we need to also need to use the `kwic()` function. Let's first look at the words 'american' and 'people'.

```
data_corpus_inaugural_subset <- corpus_subset(data_corpus_inaugural, Year > 1992)

x11()
textplot_xray(
     kwic(data_corpus_inaugural_subset, "american"),
     kwic(data_corpus_inaugural_subset, "people")
)
```



Now let's try 'freedom' and 'rights'.

```
x11()
textplot_xray(
     kwic(data_corpus_inaugural_subset, "freedom"),
     kwic(data_corpus_inaugural_subset, "rights")
)
```



(Note: Trump does not appear because it seems he did not talk about 'freedom' or 'rights'.)

6. Sentiment Analysis

Looking at the sentiment of text (how positive or negative) is a common type of analysis done in social science. As with all the different types of text analysis, whether or not to look at sentiment is dependent on your research question.

We are going to use the `Quanteda` built-in Lexicoder Sentiment Dictionary (LSD) developed by Young and Soroka.

```
lengths(data_dictionary_LSD2015)
```

```
##     negative     positive neg_positive neg_negative 
##         2858         1709         1721         2860
```

This shows the number of words that are classified as negative and positive in the dictionary. The `neg_positive` column are positive words that are proceed by a negation (e.g., 'not good') and thus are negative. The `neg_negative` column are negative words that are proceed by a negatiion (e.g., 'not bad') and thus are positive. Words that are not expressly positive or negative ('neutral' words) are excluded here. To create a sentiment dfm, we use the `dfm_lookup()` function.

```
lsd_dfm <- dfm_lookup(dfm1, data_dictionary_LSD2015)
head(lsd_dfm)
```

```
## Document-feature matrix of: 6 documents, 4 features (50% sparse).
## 6 x 4 sparse Matrix of class "dfm"
##                  features
```

```
## docs               negative positive neg_positive neg_negative
##   1789-Washington       39      121            0            0
##   1793-Washington        3       10            0            0
##   1797-Adams            60      238            0            0
##   1801-Jefferson        66      177            0            0
##   1805-Jefferson        85      164            0            0
##   1809-Madison          55      138            0            0
```

```
tail(lsd_dfm)
```

```
## Document-feature matrix of: 6 documents, 4 features (50% sparse).
## 6 x 4 sparse Matrix of class "dfm"
##              features
## docs           negative positive neg_positive neg_negative
##   1997-Clinton       69      147            0            0
##   2001-Bush          62      151            0            0
##   2005-Bush          86      212            0            0
##   2009-Obama        111      192            0            0
##   2013-Obama         76      169            0            0
##   2017-Trump         43      123            0            0
```

Even though Obama appears to have more words that are classified, it is surprising how many are classified as negative.

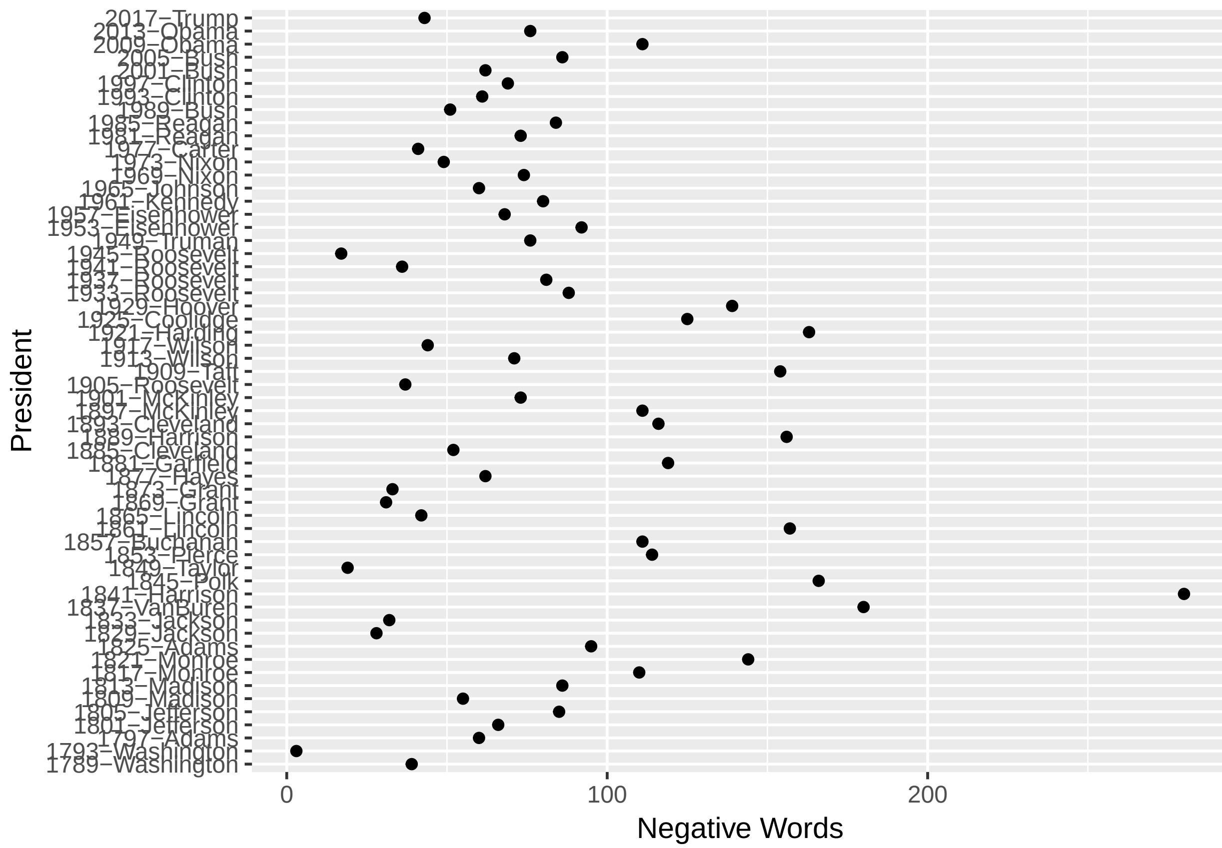
Often we prefer to plot out the sentiment values. Because `ggplot()` does not work with dfms, we will transform the dfm to a standard data frame using the `as.data.frame()` function. We first plot the negative words used.

```
lsd <- as.data.frame(lsd_dfm)
lsd
```

```
##           document negative positive neg_positive neg_negative
## 1  1789-Washington       39      121            0            0
## 2  1793-Washington        3       10            0            0
## 3       1797-Adams       60      238            0            0
## 4   1801-Jefferson       66      177            0            0
## 5   1805-Jefferson       85      164            0            0
## 6     1809-Madison       55      138            0            0
## 7     1813-Madison       86      106            0            0
## 8      1817-Monroe      110      338            0            0
## 9      1821-Monroe      144      344            0            0
## 10      1825-Adams       95      242            0            0
## 11    1829-Jackson       28      110            0            0
## 12    1833-Jackson       32       90            0            0
## 13   1837-VanBuren      180      351            0            0
## 14   1841-Harrison      280      569            0            0
## 15       1845-Polk      166      388            0            0
## 16     1849-Taylor       19      127            0            0
## 17     1853-Pierce      114      331            0            0
## 18   1857-Buchanan      111      247            0            0
## 19    1861-Lincoln      157      209            0            0
## 20    1865-Lincoln       42       37            0            0
## 21      1869-Grant       31       97            0            0
## 22      1873-Grant       33       95            0            0
## 23      1877-Hayes       62      218            0            0
## 24   1881-Garfield      119      233            0            0
## 25  1885-Cleveland       52      176            0            0
```

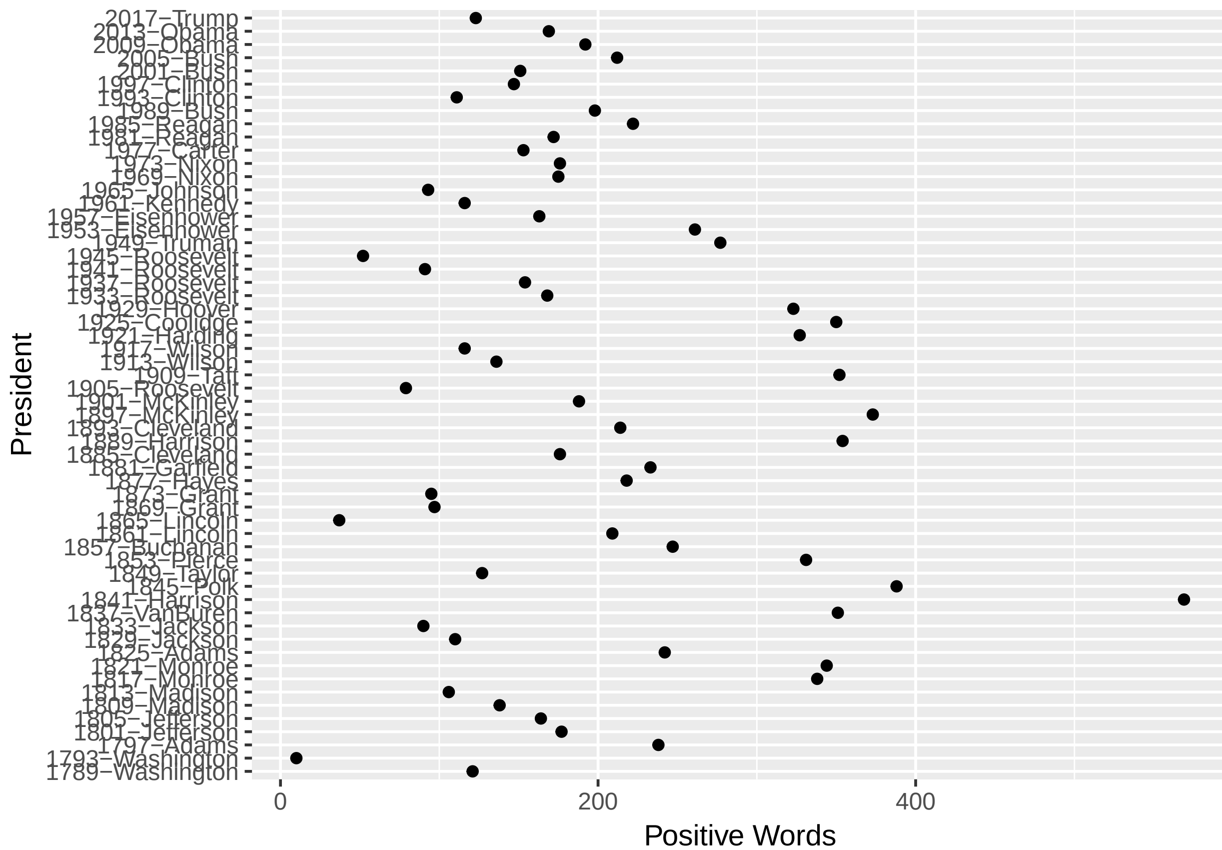
```
## 26    1889-Harrison      156      354            0             0
## 27   1893-Cleveland      116      214            0             0
## 28    1897-McKinley      111      373            0             0
## 29    1901-McKinley       73      188            0             0
## 30   1905-Roosevelt       37       79            0             0
## 31        1909-Taft      154      352            0             0
## 32      1913-Wilson       71      136            0             0
## 33      1917-Wilson       44      116            0             0
## 34     1921-Harding      163      327            0             0
## 35    1925-Coolidge      125      350            0             0
## 36      1929-Hoover      139      323            0             0
## 37   1933-Roosevelt       88      168            0             0
## 38   1937-Roosevelt       81      154            0             0
## 39   1941-Roosevelt       36       91            0             0
## 40   1945-Roosevelt       17       52            0             0
## 41      1949-Truman       76      277            0             0
## 42 1953-Eisenhower       92      261            0             0
## 43 1957-Eisenhower       68      163            0             0
## 44     1961-Kennedy       80      116            0             0
## 45     1965-Johnson       60       93            0             0
## 46       1969-Nixon       74      175            0             0
## 47       1973-Nixon       49      176            0             0
## 48      1977-Carter       41      153            0             0
## 49      1981-Reagan       73      172            0             0
## 50      1985-Reagan       84      222            0             0
## 51        1989-Bush       51      198            0             0
## 52     1993-Clinton       61      111            0             0
## 53     1997-Clinton       69      147            0             0
## 54        2001-Bush       62      151            0             0
## 55        2005-Bush       86      212            0             0
## 56       2009-Obama      111      192            0             0
## 57       2013-Obama       76      169            0             0
## 58       2017-Trump       43      123            0             0
```

```
x11()
ggplot(data=lsd, aes(x=document, y=negative)) +
  geom_point() +
  coord_flip() +
  labs(x="President",y="Negative Words")
```



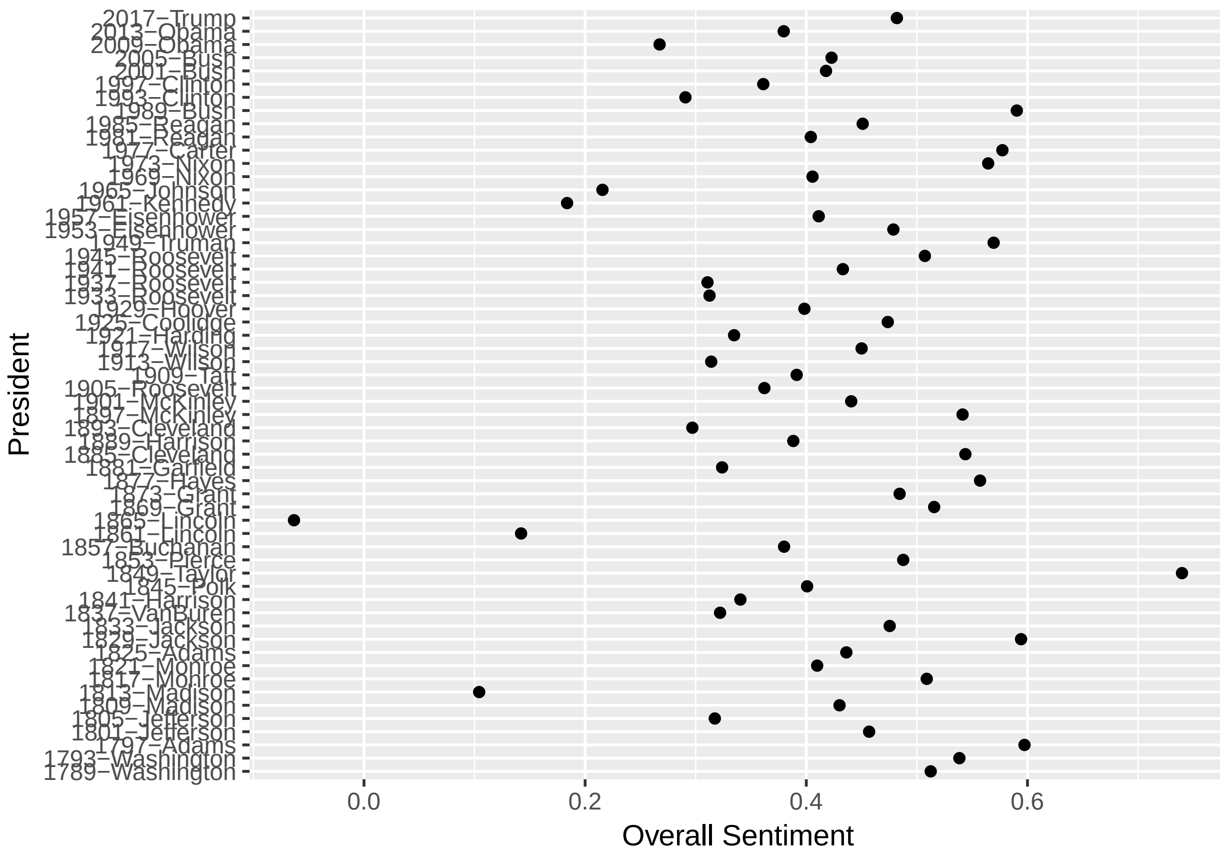
Now, we plot the positive words used.

```
x11()
ggplot(data=lsd, aes(x=document, y=positive)) +
  geom_point() +
  coord_flip() +
  labs(x="President",y="Positive Words")
```



Since the previous plots are a bit misleading because it is just the raw counts, let's do a plot of how positive or negative the speech was out of the total number of positive and negative words used.

```
lsd$overall <- (lsd$positive-lsd$negative)/(lsd$positive+lsd$negative)

x11()
ggplot(data=lsd, aes(x=document, y=overall)) +
  geom_point() +
  coord_flip() +
  labs(x="President",y="Overall Sentiment")
```



Interestingly, Lincoln's 1865 speech is the only one that had more negative than positive words; makes some sense since it was the last year of the US Civil War.

7. Topic Models

Topic models in general are increasingly used by social scientists in situations where a researcher wants to uncover themes or topics within the text. There are a variety of approaches researchers have used (Grimmer 2010; Grimmer 2013), but the Latent Dirichlet Allocation (LDA) approach is one of the simplest and most widely employed (Blei et al. 2003; Blei 2012). The basic concept of the LDA process is to assume unobserved topics exist amongst a set of documents; the documents are observable, but the topics are latent. The assumption is that the topics arise probabilistically from the words in the documents following a Dirichlet distribution. Different topics have different words that arise with higher probabilities than other words. For example, a 'free speech' topic will have words associated with free speech at higher probabilities than non-free speech words. This unsupervised, algorithmic approach is an efficient way of ascertaining meaning and structure out of a large set of text.

Topic models are not part of `Quanteda`'s internal package, but they are still simple to do using `Quanteda`. To do so, we use the `convert()` function to create a dtm from the dfm. Then we use the `LDA()` function from the `topicmodels` package and ask for 10 topics ($k = 10$). The number of topics is subjective and is more-or-less an art dependent on the size of the corpus; though 10 topics is usually the standard starting point. Running LDA topic models is computationally intensive where the larger the size of the corpus/dfm/dtm, the longer it will take.

```
library(topicmodels)
```

```
## Warning: package 'topicmodels' was built under R version 3.4.4
```

```r
dtm <- convert(dfm1, to="topicmodels")
lda <- LDA(dtm, k=10)

terms(lda,10)
```

```
##       Topic 1        Topic 2      Topic 3   Topic 4      Topic 5
##  [1,] "people"       "government" "us"      "upon"       "us"
##  [2,] "shall"        "great"      "world"   "government" "new"
##  [3,] "can"          "states"     "freedom" "people"     "world"
##  [4,] "upon"         "country"    "can"     "may"        "people"
##  [5,] "may"          "people"     "must"    "every"      "america"
##  [6,] "states"       "public"     "peace"   "can"        "must"
##  [7,] "constitution" "every"      "people"  "us"         "nation"
##  [8,] "government"   "united"     "nation"  "states"     "can"
##  [9,] "laws"         "union"      "nations" "public"     "time"
## [10,] "now"          "may"        "new"     "country"    "let"
##       Topic 6        Topic 7      Topic 8      Topic 9      Topic 10
##  [1,] "government"   "people"     "people"     "government" "can"
##  [2,] "states"       "government" "government" "upon"       "great"
##  [3,] "union"        "states"     "must"       "people"     "upon"
##  [4,] "may"          "war"        "can"        "can"        "government"
##  [5,] "country"      "nation"     "world"      "may"        "people"
##  [6,] "people"       "upon"       "shall"      "power"      "country"
##  [7,] "shall"        "can"        "us"         "must"       "every"
##  [8,] "one"          "shall"      "may"        "states"     "must"
##  [9,] "constitution" "may"        "public"     "country"    "nation"
## [10,] "can"          "united"     "national"   "great"      "world"
```

As should be apparent from the output, the substantive meaning of each topic is often not readily clear. This will vary depending on the substance of our corpus. For example, if our corpus has fairly similar documents/texts, then differences will be hard to discern. If our corpus has very different documents/texts, then the topic differences will be clearer. Here, in particular, the meaning of each topic is not clear; particularly since the same words can appear in different topics. Hence, we might want to re-run the model and look at only 5 topics.

```r
lda2 <- LDA(dtm, k=5)

terms(lda2,10)
```

```
##       Topic 1      Topic 2        Topic 3      Topic 4      Topic 5
##  [1,] "can"        "government"   "government" "people"     "us"
##  [2,] "us"         "people"       "can"        "government" "must"
##  [3,] "world"      "states"       "people"     "upon"       "world"
##  [4,] "peace"      "may"          "upon"       "states"     "new"
##  [5,] "must"       "great"        "must"       "public"     "america"
##  [6,] "people"     "country"      "states"     "shall"      "people"
##  [7,] "shall"      "power"        "may"        "country"    "nation"
##  [8,] "great"      "upon"         "world"      "may"        "freedom"
##  [9,] "new"        "union"        "peace"      "can"        "can"
## [10,] "government" "constitution" "now"        "great"      "time"
```

Even with only 5 topics, the differences in topics is not evident.

We can also look at the most likely topic for each document using the `topics()` function.

```r
head(topics(lda),20)
```

```
## 1789-Washington 1793-Washington      1797-Adams  1801-Jefferson
##              10               1               7               6
##  1805-Jefferson    1809-Madison    1813-Madison     1817-Monroe
##               4               2               7               2
##     1821-Monroe      1825-Adams    1829-Jackson    1833-Jackson
##               2               2               2               4
##   1837-VanBuren  1841-Harrison       1845-Polk     1849-Taylor
##               4               9               6               1
##     1853-Pierce   1857-Buchanan    1861-Lincoln    1865-Lincoln
##               4               6               1               7
```

```r
tail(topics(lda),20)
```

```
##  1941-Roosevelt  1945-Roosevelt     1949-Truman 1953-Eisenhower
##               3               3               3               3
## 1957-Eisenhower    1961-Kennedy    1965-Johnson      1969-Nixon
##               3               3               5               5
##      1973-Nixon     1977-Carter     1981-Reagan     1985-Reagan
##               3               3               3               5
##       1989-Bush    1993-Clinton    1997-Clinton       2001-Bush
##               5               5               5               3
##       2005-Bush      2009-Obama      2013-Obama      2017-Trump
##               3               5               3               5
```

8. Ideological Placements (Scalings)

One particularly large area of political science deals with ideological placements of parties, candidates, politicians, interest groups, etc. Historically, this meant using votes (e.g., DW-Nominate), 'experts' (e.g., Chapel Hill Expert Survey), or the subject themselves to create ideological placements. Increasingly, researchers are using text (e.g., party platforms, speeches, content of bills, etc.) to make ideological placements; moving from what actors do to what actors do and say.

Today, we look at two varieties of placements using **Wordscores** and **Wordfish**. Wordscores has been around for over decade, while Wordfish is much newer. Wordscores requires reference texts that says what is 'liberal' and 'conservative'; or whatever version one is interested in. Then we compare all remaining texts to the reference texts. Therefore, we can consider Wordscores a **supervised method**. Therefore, if your research does not have legitimate reference texts then you cannot use Wordscores. For example, we cannot use either approach for creating ideological placements of newspapers. Why? Because we do not have clear reference texts as to what is conservative and what is liberal.

Wordfish meanwhile can be considered an **unsupervised method** method because it does not rely on reference texts. Instead of relying on the reference texts, it estimates documents positions solely based on the frequency of the words. So, can we use Wordfish for ideological placments/scalings of any type of document? According to Slapin (co-author of Wordfish), no. Even though Wordfish does not rely on reference texts, Slapin still is very cautious about using Wordfish when working with documents that do not clearly have ideological differences. For example, Slapin says we cannot use Wordfish to place newspapers… unless the newspapers are owned/controlled by political parties.

8.1. Wordscores

Due to the requirement to need a reference, we will make use of `Quanteda`'s built-in corpus of 2010 Irish budget speeches (`data_corpus_irishbudget2010`).

```
ie_dfm <- data_corpus_irishbudget2010 %>%
     dfm(remove = stopwords("english"), remove_punct = TRUE)
ie_dfm
```

```
## Document-feature matrix of: 14 documents, 5,008 features (82.7% sparse).
```

```
# Set reference scores
refscores <- c(rep(NA, 4), 1, -1, rep(NA, 8))

# Predict Wordscores model
ws <- textmodel_wordscores(ie_dfm, refscores, smooth = 1)
ws
```

```
##
## Call:
## textmodel_wordscores.dfm(x = ie_dfm, y = refscores, smooth = 1)
##
## Scale: linear; 2 reference scores; 5008 scored features.
```

```
summary(ws)
```

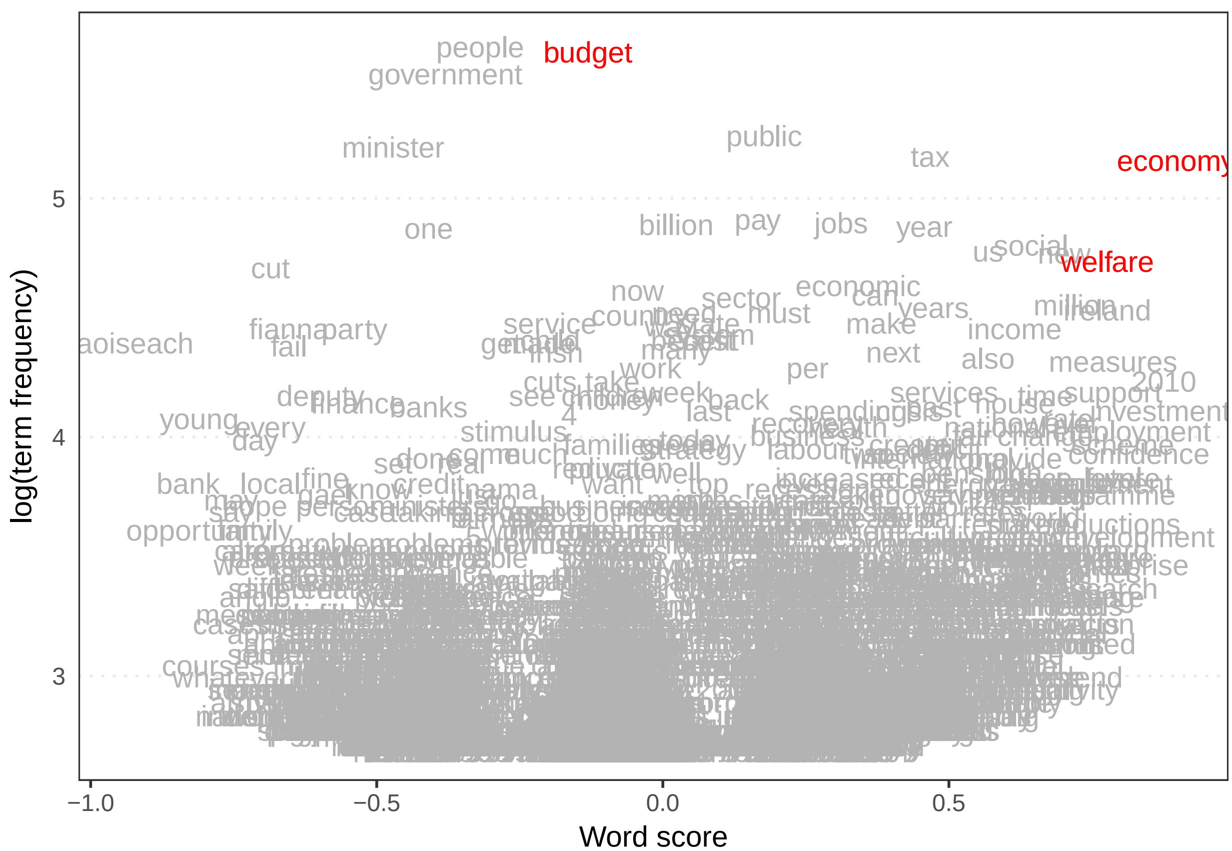
```
##
## Call:
## textmodel_wordscores.dfm(x = ie_dfm, y = refscores, smooth = 1)
##
## Reference Document Statistics:
##                                         score total min max  mean median
## 2010_BUDGET_01_Brian_Lenihan_FF            NA  9286   1  39 1.854      1
## 2010_BUDGET_02_Richard_Bruton_FG           NA  6865   1  46 1.371      1
## 2010_BUDGET_03_Joan_Burton_LAB             NA  7976   1  42 1.593      1
## 2010_BUDGET_04_Arthur_Morgan_SF            NA  8219   1  54 1.641      1
## 2010_BUDGET_05_Brian_Cowen_FF               1  8205   1  34 1.638      1
## 2010_BUDGET_06_Enda_Kenny_FG               -1  6879   1  28 1.374      1
## 2010_BUDGET_07_Kieran_ODonnell_FG          NA  6011   1  26 1.200      1
## 2010_BUDGET_08_Eamon_Gilmore_LAB           NA  6934   1  29 1.385      1
## 2010_BUDGET_09_Michael_Higgins_LAB         NA  5526   1   9 1.103      1
## 2010_BUDGET_10_Ruairi_Quinn_LAB            NA  5522   1  13 1.103      1
## 2010_BUDGET_11_John_Gormley_Green          NA  5485   1  11 1.095      1
## 2010_BUDGET_12_Eamon_Ryan_Green            NA  5692   1  15 1.137      1
## 2010_BUDGET_13_Ciaran_Cuffe_Green          NA  5582   1  19 1.115      1
## 2010_BUDGET_14_Caoimhghin_OCaolain_SF      NA  6911   1  29 1.380      1
##
## Wordscores:
## (showing first 30 elements)
##       presented   supplementary          budget           house
##        -0.40925        -0.40925        -0.13089         0.61479
##            last           april            said            work
##         0.07993        -0.71280        -0.71280        -0.02137
##             way          period          severe        economic
##         0.01220         0.35400         0.25283         0.34126
##        distress           today             can          report
```

```
##        -0.08791      0.05565      0.37103      0.43104
## notwithstanding    difficulties         past        eight
##        -0.08791      0.25283      0.47294     -0.08791
##          months          now         road     recovery
##         0.05565     -0.04460     -0.40925      0.25283
##        enormous      benefit         main    political
##        -0.40925      0.05565      0.11410     -0.65344
##         parties        share
##         0.43104     -0.40925
```

We see that two of the texts are set as the reference points - one at 1 and the other at -1.

Next, we plot the word position and highlight a few in red.

```r
x11()
textplot_scale1d(ws,
                 highlighted = c("economy", "welfare", "budget"),
                 highlighted_color = "red")
```



This plot shows the placements of the words relative to the reference texts and their frequency.

Next, we use the word scores to predict the locations of the documents using the `pred()` function.

```r
pred <- predict(ws)
pred
```

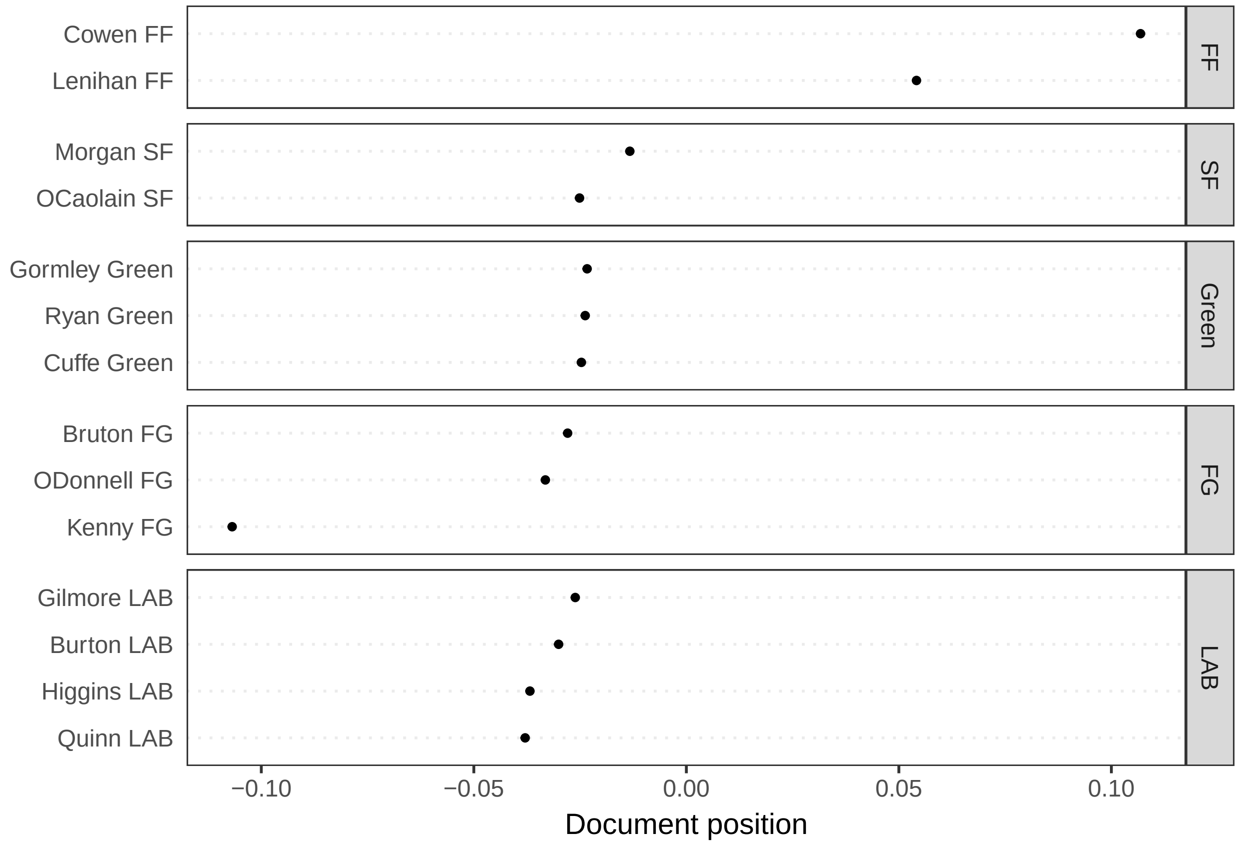
```
##       2010_BUDGET_01_Brian_Lenihan_FF
##                            0.05414996
##      2010_BUDGET_02_Richard_Bruton_FG
##                           -0.02792832
```

```
##          2010_BUDGET_03_Joan_Burton_LAB
##                             -0.03005066
##         2010_BUDGET_04_Arthur_Morgan_SF
##                             -0.01329439
##           2010_BUDGET_05_Brian_Cowen_FF
##                              0.10685927
##            2010_BUDGET_06_Enda_Kenny_FG
##                             -0.10685927
##       2010_BUDGET_07_Kieran_ODonnell_FG
##                             -0.03316757
##        2010_BUDGET_08_Eamon_Gilmore_LAB
##                             -0.02613156
##      2010_BUDGET_09_Michael_Higgins_LAB
##                             -0.03679662
##         2010_BUDGET_10_Ruairi_Quinn_LAB
##                             -0.03792328
##       2010_BUDGET_11_John_Gormley_Green
##                             -0.02334390
##         2010_BUDGET_12_Eamon_Ryan_Green
##                             -0.02379303
##       2010_BUDGET_13_Ciaran_Cuffe_Green
##                             -0.02469264
## 2010_BUDGET_14_Caoimhghin_OCaolain_SF
##                             -0.02512630
```

This is now the placements of the budget speeches.

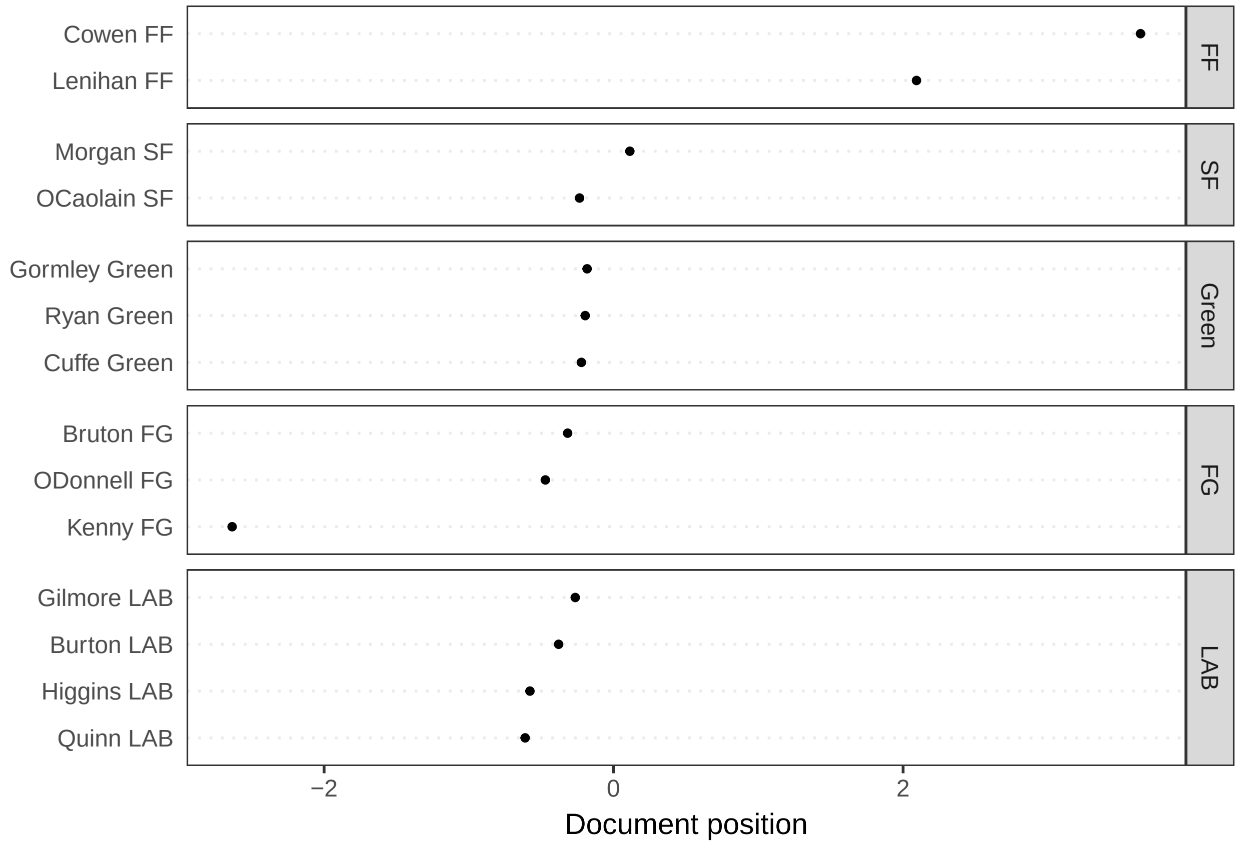
We can also plot the placements of the budget speeches using the `textplot_scale1d()`. First we set the document labels for the y-axis and then we plot the predicted placements.

```r
# Set document labels for y-axis
doclab <- apply(docvars(data_corpus_irishbudget2010, c("name", "party")),
                1, paste, collapse = " ")

# Plot estimated document positions and group by "party" variable
x11()
textplot_scale1d(pred, margin = "documents",
                 doclabels = doclab,
                 groups = docvars(data_corpus_irishbudget2010, "party"))
```



We can re-scale the x-axis to make a clearer comparison with Wordfish below.

```
pred_lbg <- predict(ws, rescaling = "lbg")

x11()
textplot_scale1d(pred_lbg, margin = "documents",
                 doclabels = doclab,
                 groups = docvars(data_corpus_irishbudget2010, "party"))
```



8.2. Wordfish

Now let's replicate the placements using the `textmodel_wordfish()` function with `ie_dfm`. There are a few different ways to identify the Wordfish model. Below we will follow the `Quanteda` tutorial by fixing two texts and specifying that text 6 is to the left of text 5 (`dir=c(6,5)`); though note that doing this does not set the scaling domain like Wordscores does.

```
wf <- textmodel_wordfish(ie_dfm, dir = c(6,5))
wf
```

```
##
## Call:
## textmodel_wordfish.dfm(x = ie_dfm, dir = c(6, 5))
##
## Dispersion: poisson; direction: 6 < 5; 14 documents; 5008 features.
```

```
summary(wf)
```

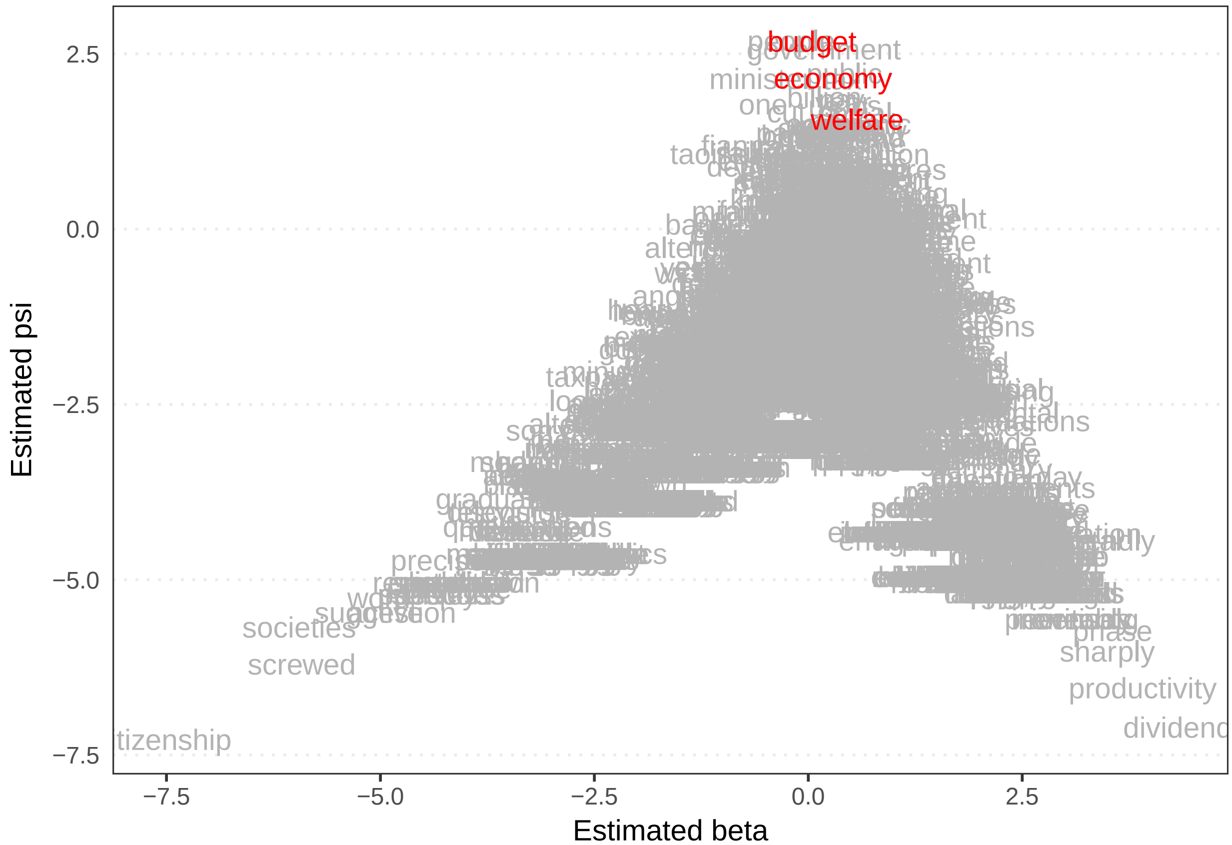
```
##
## Call:
## textmodel_wordfish.dfm(x = ie_dfm, dir = c(6, 5))
##
## Estimated Document Positions:
##                                          theta      se
## 2010_BUDGET_01_Brian_Lenihan_FF        1.726667 0.02238
## 2010_BUDGET_02_Richard_Bruton_FG      -0.497193 0.03087
```

```
## 2010_BUDGET_03_Joan_Burton_LAB        -0.991714 0.01804
## 2010_BUDGET_04_Arthur_Morgan_SF        0.085603 0.02889
## 2010_BUDGET_05_Brian_Cowen_FF          1.903242 0.02479
## 2010_BUDGET_06_Enda_Kenny_FG          -0.776897 0.02635
## 2010_BUDGET_07_Kieran_ODonnell_FG     -0.283465 0.04612
## 2010_BUDGET_08_Eamon_Gilmore_LAB      -0.378940 0.03201
## 2010_BUDGET_09_Michael_Higgins_LAB    -1.210983 0.03562
## 2010_BUDGET_10_Ruairi_Quinn_LAB       -1.231395 0.03503
## 2010_BUDGET_11_John_Gormley_Green      1.039543 0.07677
## 2010_BUDGET_12_Eamon_Ryan_Green        0.114674 0.06288
## 2010_BUDGET_13_Ciaran_Cuffe_Green      0.504108 0.07199
## 2010_BUDGET_14_Caoimhghin_OCaolain_SF -0.003249 0.03671
##
## Estimated Feature Scores:
##      presented supplementary  budget   house   last   april    said   work
## beta    0.3181         1.092 0.04255 0.03628 0.2550 -0.2138 -0.9591 0.5193
## psi    -1.8281        -1.184 2.68012 1.01149 0.9405 -0.5944 -0.4582 1.0680
##         way  period severe economic distress  today    can  report
## beta 0.2949  0.5526  1.316   0.4712    1.704 0.1071 0.3369  0.6585
## psi  1.3705 -0.2549 -2.151   1.5029   -4.316 0.8039 1.5100 -0.3204
##      notwithstanding difficulties   past  eight months    now    road
## beta           1.704        1.231 0.5029  1.704 0.7138 0.3265  0.1144
## psi           -4.316       -1.455 0.8701 -4.316 0.2002 1.5366 -0.1131
##      recovery enormous  benefit    main political parties    share
## beta   0.3986 -0.05312 -0.04021  0.9277   -0.3670  0.5242 -0.09684
## psi    0.8296 -1.08890  1.33930 -0.9169   -0.4442 -0.1218 -0.17513
```

There are three things to note about the output. First, `theta` provides the estimated document positions; again, positive means conservative and negative means liberal. Second, for the estimates feature scores, `beta` is the word weight (informativeness) and `psi` is the word frequency; these are normalised to 0, thus we can have negative values.

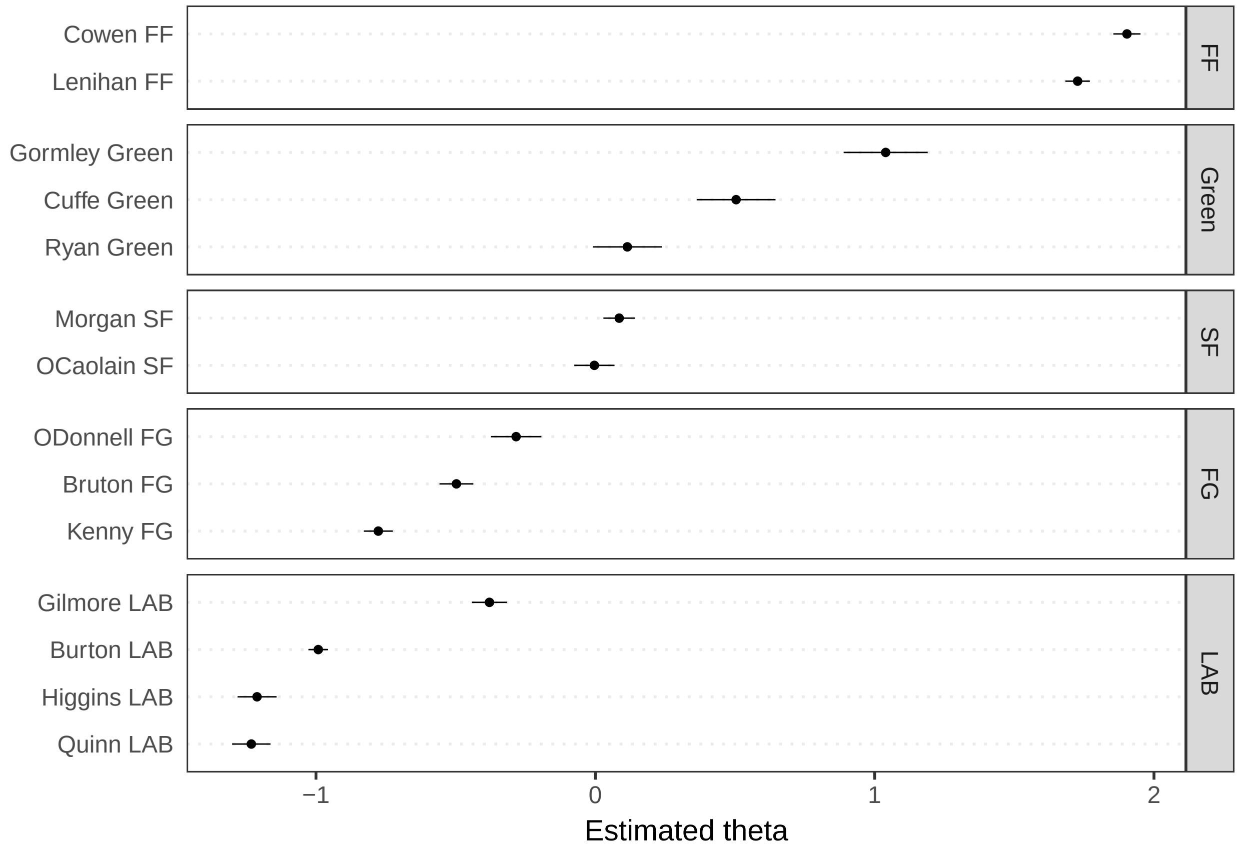
Next, we plot β and ψ and highlight a few in red.

```r
x11()
textplot_scale1d(wf, margin="features",
                 highlighted = c("economy", "welfare", "budget"),
                 highlighted_color = "red")
```



Unlike Wordscores, we do not need to run a prediction function to estimate the document positions, because Wordfish has already done that. Now let's plot the documents.

```
x11()
textplot_scale1d(wf, margin = "documents",
                 doclabels = doclab,
                 groups = docvars(data_corpus_irishbudget2010, "party"))
```



We see that compared to Wordscores, the Wordfish placements are similar and also automatically provide confidence intervals.

9. On Your Own

Try working through the above with your own data or the voter fraud data. For the voter fraud data, without additional manipulation, you can only look at lexical diversity, sentiment, and topic models.