Text Scaling

Loren Collingwood

This script reviews how to use wordscores and wordfish in R. These are two methods political scientists have used and use to generate unidimensional scales of text documents (e.g., ideology of speech liberal to conservative). Although significant advancements in scaling have been made, you can still use these methods to great effect to understand your textual data (based on theories you may have). A lot of work in this space looks at party manifestos (in European parliaments) or party platforms and/or speeches. However, if your theory makes sense, you can apply it in many contexts, for example, below I apply it to news coverage of homicides in Chicago.

Step 1

Training a Wordscores model requires reference scores for texts whose policy positions on well-defined a priori dimensions are "known". Afterwards, Wordscores estimates the positions for the remaining "virgin" texts.

We use manifestos of the 2013 and 2017 German federal elections. For the 2013 elections we assign the average expert evaluations from the 2014 Chapel Hill Expert Survey for the five major parties, and predict the party positions for the 2017 manifestos.

```
options(scipen = 999, digits = 4)
##############################
# Install and Load Packages #
##############################

#install.packages("quanteda")
library(quanteda)
```

```
## Package version: 2.1.1

## Parallel computing: 2 of 8 threads used.

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

##
## Attaching package: 'quanteda'

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

```
#install.packages("quanteda.textmodels")
library(quanteda.textmodels)
```

```
##
## Attaching package: 'quanteda.textmodels'

## The following object is masked from 'package:quanteda':
##
##     data_dfm_lbgexample
```

```r
#install.packages("readxl")
library(readxl)

# Gather the Corpus of text I've stored it locally in RDS file #
corp_ger <-  readRDS("~/Dropbox/collingwood_research/posc_fall_20/POSC-207/data/data_corpus_germanifest
summary(corp_ger)
```

```
## Corpus consisting of 12 documents, showing 12 documents:
##
##           Text Types Tokens Sentences year   party ref_score
##       AfD 2013   450    944        43 2013     AfD        NA
##   CDU-CSU 2013  7615  46535      2527 2013 CDU-CSU      5.92
##       FDP 2013  7953  42298      2375 2013     FDP      6.53
##    Gruene 2013 13839  93595      5126 2013  Gruene      3.61
##     Linke 2013  8451  43382      1850 2013   Linke      1.23
##       SPD 2013  8360  47348      2532 2013     SPD      3.76
##       AfD 2017  5947  18754       715 2017     AfD        NA
##   CDU-CSU 2017  4890  21510      1256 2017 CDU-CSU        NA
##       FDP 2017  8676  37609      1925 2017     FDP        NA
##    Gruene 2017 13353  72645      3220 2017  Gruene        NA
##     Linke 2017 11830  65728      2755 2017   Linke        NA
##       SPD 2017  8400  41938      2401 2017     SPD        NA
```

Step 2

Convert the corpus to a document term/frequency matrix

```r
# Create a Document-Feature/Term Matrix #
dfmat_ger <- dfm(corp_ger, remove = stopwords("de"), remove_punct = TRUE)
```

Step 3

Apply Wordscores algorithm to document-feature matrix

```r
tmod_ws <- textmodel_wordscores(dfmat_ger, y = corp_ger$ref_score, smooth = 1)
summary(tmod_ws)
```

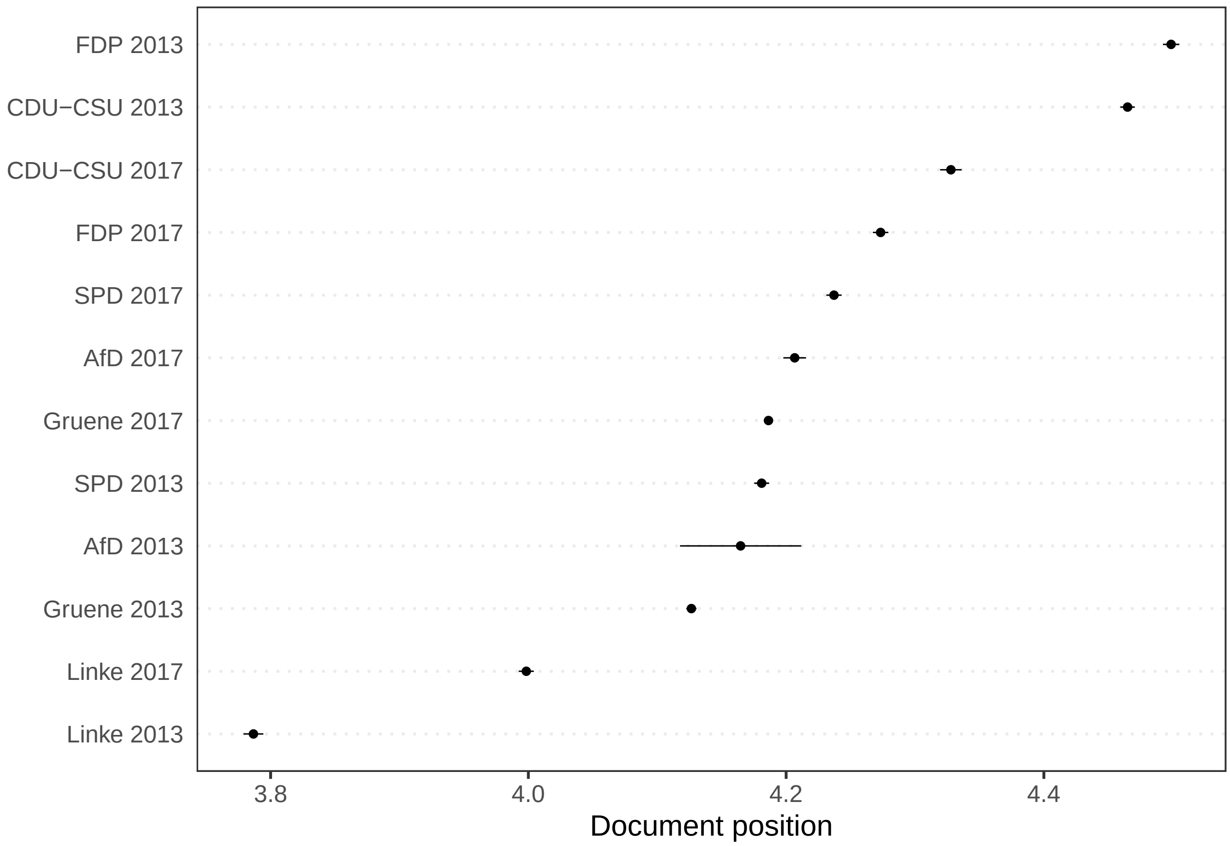
```
##
## Call:
## textmodel_wordscores.dfm(x = dfmat_ger, y = corp_ger$ref_score,
##     smooth = 1)
##
## Reference Document Statistics:
##              score total min  max   mean median
## AfD 2013        NA   455   0   23 0.0109      0
## CDU-CSU 2013  5.92 23060   0  245 0.5537      0
## FDP 2013      6.53 20603   0  186 0.4947      0
## Gruene 2013   3.61 45759   0  398 1.0988      0
## Linke 2013    1.23 21011   0  234 0.5045      0
## SPD 2013      3.76 23150   0  214 0.5559      0
## AfD 2017        NA  9899   0  108 0.2377      0
```

```
## CDU-CSU 2017    NA 10753   0  136 0.2582      0
## FDP 2017        NA 19358   0  261 0.4648      0
## Gruene 2017     NA 40982   0 1086 0.9841      0
## Linke 2017      NA 33347   0  788 0.8007      0
## SPD 2017        NA 20836   0  186 0.5003      0
##
## Wordscores:
## (showing first 30 elements)
##          alternative            deutschland            wahlprogramm
##                 3.29                   4.74                    3.30
##      währungspolitik                fordern               geordnete
##                 4.53                   3.26                    4.24
##            auflösung euro-währungsgebietes                 braucht
##                 3.34                   4.24                    4.15
##                 euro                ländern                 schadet
##                 3.33                   4.23                    3.91
##     wiedereinführung              nationaler              währungen
##                 4.46                   4.58                    4.24
##            schaffung              kleinerer              stabilerer
##                 4.29                   4.43                    4.24
##     währungsverbünde                     dm                    darf
##                 4.24                   4.24                    3.87
##                 tabu               änderung             europäischen
##                 4.16                   4.23                    4.36
##             verträge                  staat              ausscheiden
##                 3.55                   4.79                    3.70
##          ermöglichen                   volk             demokratisch
##                 4.36                   4.24                    2.27
```

Step 4

Predict the Wordscores on the virgin text, then plot it out.

```r
pred_ws <- predict(tmod_ws, se.fit = TRUE, newdata = dfmat_ger)

# Plot it out real good #
textplot_scale1d(pred_ws)
```



Now try it out with toy example. This will give you sort of funky but still somewhat interpretable results.

```
# Create a corpus
feaux_corp <- corpus(
    c("this is love",
    "hate is all i've got",
    "these losers suck so much",
    "love and like the dogs they're pretty",
    "mitt romney hates to vote that way he won't",
    "trump is a hater and loser, I hate him so much",
    "biden is a loser and hater, he just loses always",
    "harris will win she's the best omg, love harris ",
    "when you're young you're idealstic but that's not wrong",
    "politics is about doing what's right so really its an effort of love")
)

# Add on the toy scores #
docvars(feaux_corp, "ref_score") <- c(10, 1, 2, 8,NA, NA, NA, NA, NA, 9)

# Take  look real nice #
summary(feaux_corp)
```

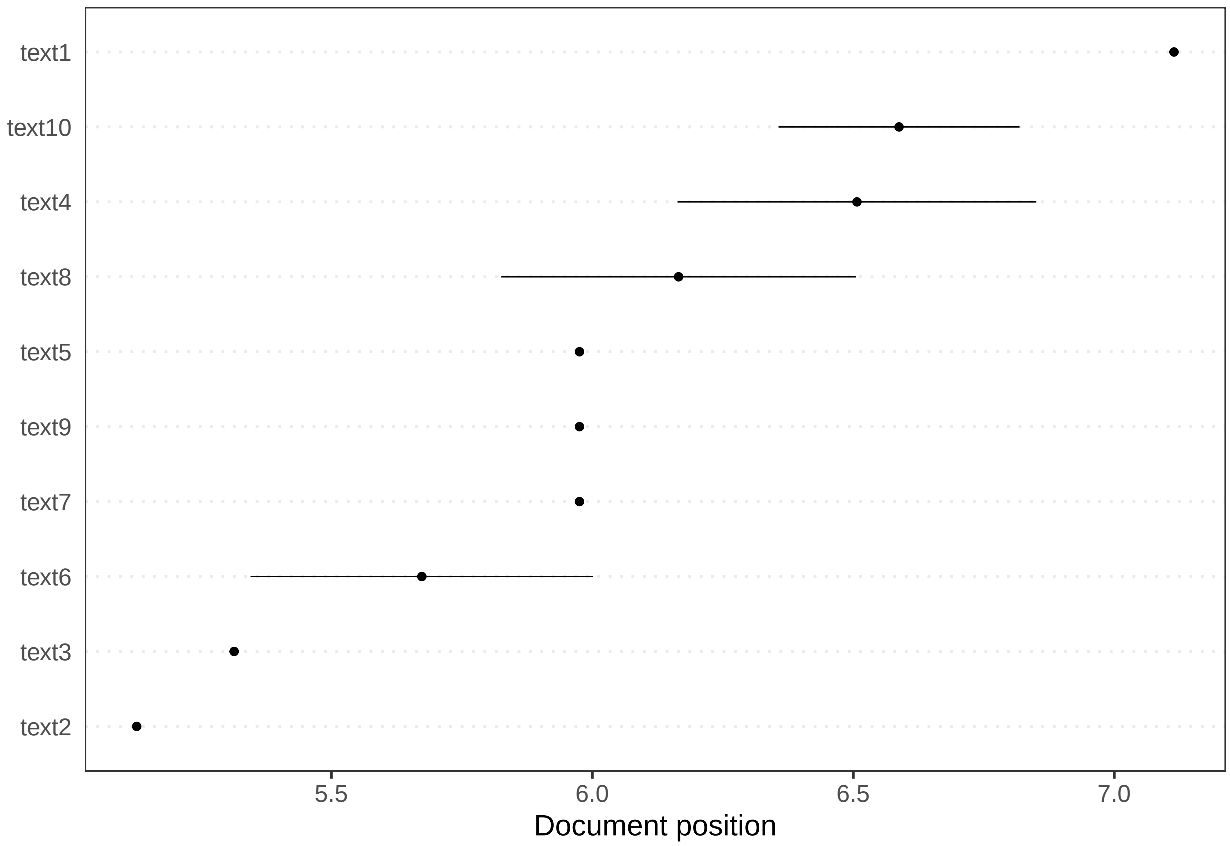
```
## Corpus consisting of 10 documents, showing 10 documents:
##
##    Text Types Tokens Sentences ref_score
##   text1     3      3         1        10
##   text2     5      5         1         1
##   text3     5      5         1         2
```

```
##    text4     7      7          1         8
##    text5     9      9          1        NA
##    text6    12     12          1        NA
##    text7    11     11          1        NA
##    text8     9     10          1        NA
##    text9     8      9          1        NA
##   text10    13     13          1         9
```

```r
# Create a Document-Feature/Term Matrix #
dfmat_feaux <- dfm(feaux_corp,
                 remove = stopwords("english"),
                 remove_punct = TRUE)

# Apply Wordscores algorithm to document-feature matrix
tmod_ws <- textmodel_wordscores(dfmat_feaux, y = feaux_corp$ref_score, smooth = 1)
summary(tmod_ws)
```

```
##
## Call:
## textmodel_wordscores.dfm(x = dfmat_feaux, y = feaux_corp$ref_score,
##     smooth = 1)
##
## Reference Document Statistics:
##        score total min max   mean median
## text1     10     1   0   1 0.0312      0
## text2      1     2   0   1 0.0625      0
## text3      2     3   0   1 0.0938      0
## text4      8     4   0   1 0.1250      0
## text5     NA     5   0   1 0.1562      0
## text6     NA     5   0   1 0.1562      0
## text7     NA     6   0   1 0.1875      0
## text8     NA     6   0   2 0.1875      0
## text9     NA     3   0   1 0.0938      0
## text10     9     5   0   1 0.1562      0
##
## Wordscores:
## (showing first 30 elements)
##      love      hate       got    losers      suck      much      like      dogs
##      7.11      5.13      5.13      5.31      5.31      5.31      6.30      6.30
##    pretty      mitt    romney     hates      vote       way     trump     hater
##      6.30      5.98      5.98      5.98      5.98      5.98      5.98      5.98
##     loser     biden      just     loses    always    harris       win      best
##      5.98      5.98      5.98      5.98      5.98      5.98      5.98      5.98
##       omg     young idealstic     wrong  politics     right
##      5.98      5.98      5.98      5.98      6.46      6.46
```

```r
# Predict the Wordscores on the virgin text #
pred_ws <- predict(tmod_ws, se.fit = TRUE, newdata = dfmat_feaux)

# Plot it out real good #
textplot_scale1d(pred_ws)
```



Wordfish Scaling

Step 1

Read in the data, this comes from media stories about homicide victims in Chicago in 2014 during the months of August and September (or so).

```
# Read in Data #
nc <- read_xlsx("~/Dropbox/collingwood_research/posc_fall_20/POSC-207/data/news_coverage_WordfishReady.:
```

```
## New names:
## * `` -> ...3
## * `` -> ...4
## * `` -> ...5
## * `` -> ...6
## * `` -> ...7
## * ...
```

```
# Relabel column 3 #
colnames(nc)[3] <- "victim_text"
```

Step 2

Turn data into corpus then document frequency/term matrix

```r
# Turn text into corpus #
vcorpus <- corpus(nc$victim_text)
head(summary(vcorpus))
```

```
##    Text Types Tokens Sentences
## 1 text1   101    240        10
## 2 text2   297    693        38
## 3 text3    82    178         5
## 4 text4    89    180         6
## 5 text5    82    247         7
## 6 text6   193    483        16
```

```r
vdfm <- dfm(vcorpus, stem=T,
            remove_numbers=T,
            remove_punct=T,
            remove = stopwords("english"))
# Look at top set of rows
vdfm
```

```
## Document-feature matrix of: 39 documents, 842 features (91.4% sparse).
##        features
## docs    man shot kill one block away chicago polic depart headquart
##   text1   3    5    1   2     7    1       2     5      1         3
##   text2   2    7    6   2     4    1       1     4      0         0
##   text3   1    4    1   0     3    0       0     1      0         0
##   text4   1    3    1   0     1    0       0     2      0         0
##   text5   1    4    0   0     7    0       0     4      0         0
##   text6   2    4    3   0     5    0       3     4      1         0
## [ reached max_ndoc ... 33 more documents, reached max_nfeat ... 832 more features ]
```

Step 3

Estimate a Wordfish model but before you do you need to identify documents that are polar on the dimension of interest. A priori here I had identified documents 27 and 11, respectively.

```r
# Look at 27
vcorpus[[27]]
```

```
## [1] "Antoine Stewart, 51, of the 9500 block of South Oglesby Avenue, was police say was shot several
```

```r
# Look at 11
vcorpus[[11]]
```

```
## [1] "A nine-year-old boy was killed Wednesday afternoon after being shot multiple times in the chest
```

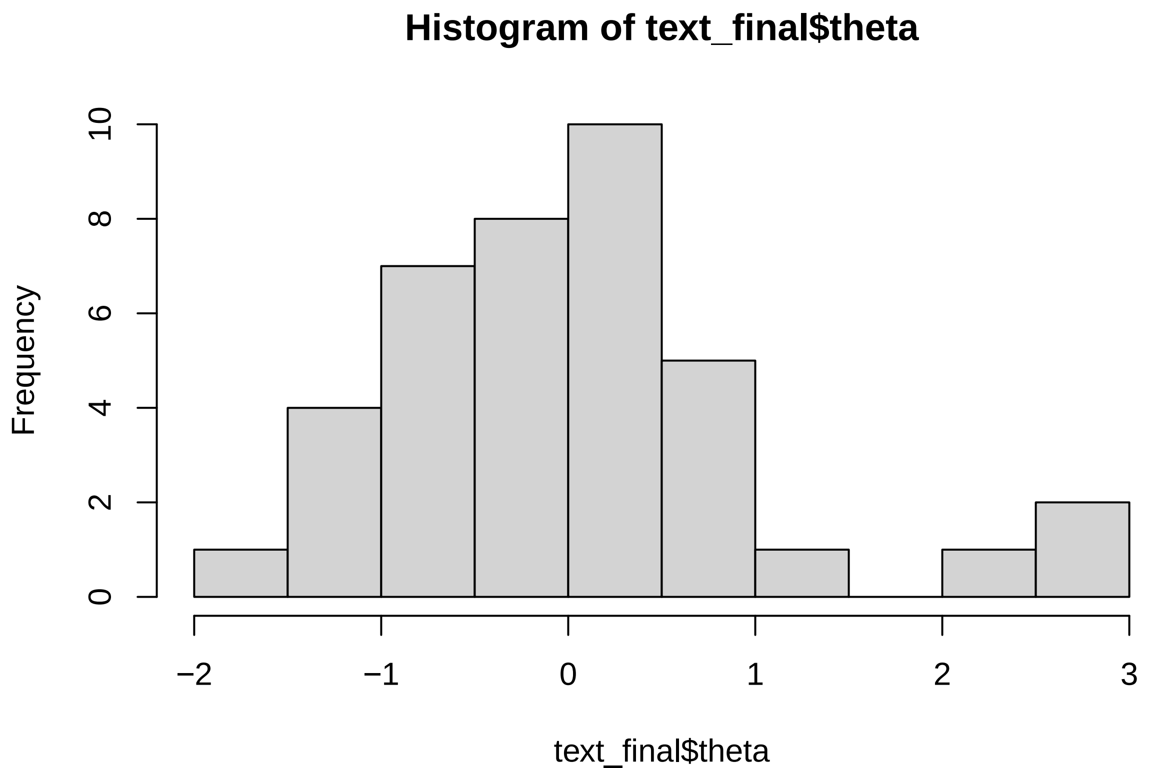
```r
# Wordfish Model #
wf <- textmodel_wordfish(vdfm,
                dir=c(27,11))# directional command -- want global identification
                             # so that document 2 receives lower value than
                             # document 1

# Take a look at the summary #
summary(wf)
```

```
##
## Call:
```

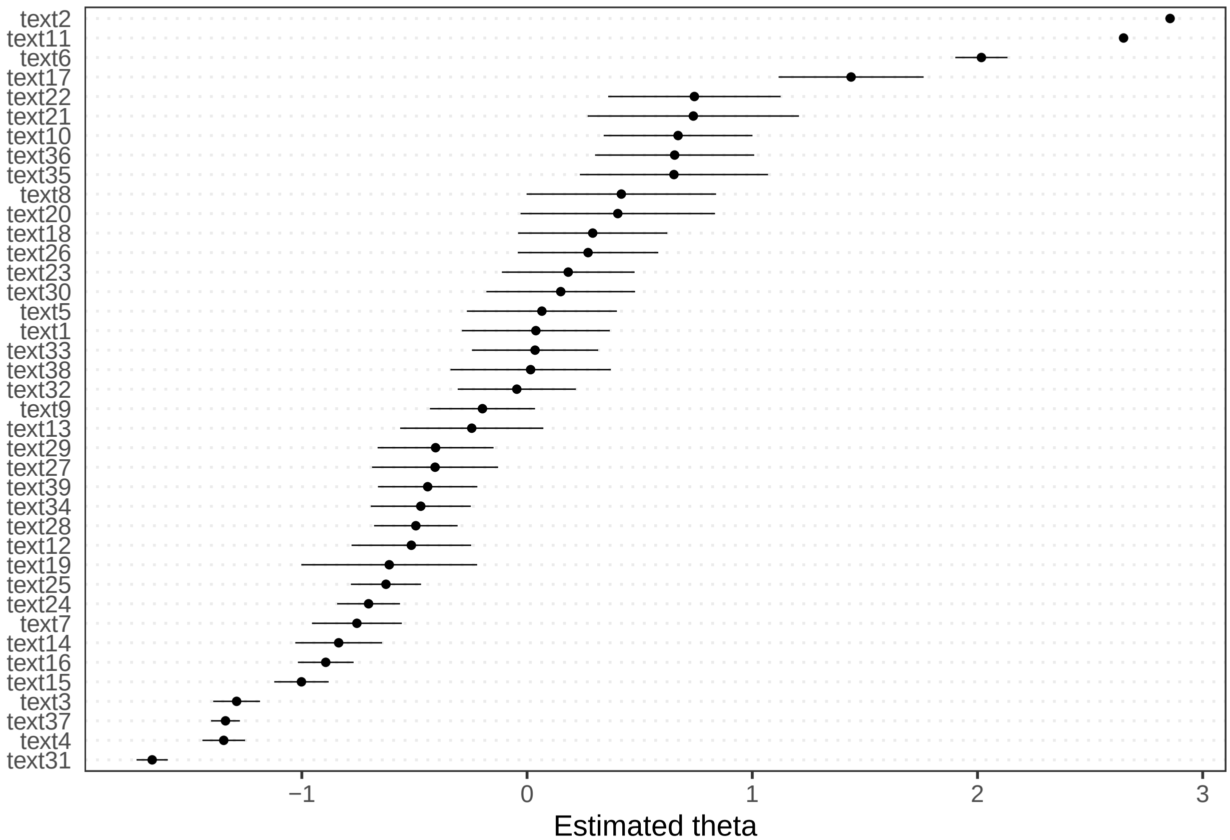
```
## textmodel_wordfish.dfm(x = vdfm, dir = c(27, 11))
##
## Estimated Document Positions:
##          theta      se
## text1   0.0391 0.16750
## text2   2.8551 0.00941
## text3  -1.2895 0.05305
## text4  -1.3467 0.04835
## text5   0.0659 0.16995
## text6   2.0174 0.05911
## text7  -0.7555 0.10178
## text8   0.4187 0.21469
## text9  -0.1980 0.11912
## text10  0.6706 0.16855
## text11  2.6486 0.00771
## text12 -0.5137 0.13555
## text13 -0.2456 0.16221
## text14 -0.8363 0.09828
## text15 -1.0017 0.06164
## text16 -0.8938 0.06310
## text17  1.4389 0.16431
## text18  0.2916 0.16904
## text19 -0.6119 0.19919
## text20  0.4029 0.22027
## text21  0.7384 0.23935
## text22  0.7430 0.19529
## text23  0.1829 0.15036
## text24 -0.7036 0.07123
## text25 -0.6264 0.07955
## text26  0.2709 0.15910
## text27 -0.4084 0.14281
## text28 -0.4938 0.09461
## text29 -0.4061 0.13137
## text30  0.1496 0.16848
## text31 -1.6645 0.03547
## text32 -0.0455 0.13387
## text33  0.0356 0.14296
## text34 -0.4718 0.11330
## text35  0.6522 0.21316
## text36  0.6553 0.18028
## text37 -1.3389 0.03265
## text38  0.0159 0.18165
## text39 -0.4412 0.11256
##
## Estimated Feature Scores:
##         man   shot   kill    one  block   away chicago  polic depart headquart
## beta -0.641 -0.173  0.152  0.161 -0.331  0.309   0.299 -0.163  -0.31     -0.51
## psi   0.944  1.291 -0.289 -1.126  1.362 -3.021  -0.889  1.015  -1.55     -2.65
##      bronzevill neighborhood friday  night walter  neeli   found abdomen    p.m
## beta     -0.496      0.00676  -2.25 -0.572  -0.51 -0.525  0.0969   -1.38 -0.228
## psi      -3.052      0.19329  -1.58 -0.572  -2.65 -1.955 -0.4420   -1.54  0.425
##       south indiana accord   cook counti medic examin  offic someon approach
## beta -0.364  -0.817 -0.367 -0.495 -0.490 -0.63 -0.455 -0.419 -0.329   -0.234
## psi   0.907  -2.174  0.753  0.649  0.599  0.99  0.733  0.909 -0.977   -2.154
```

```
##         foot
## beta -0.461
## psi  -3.745
```

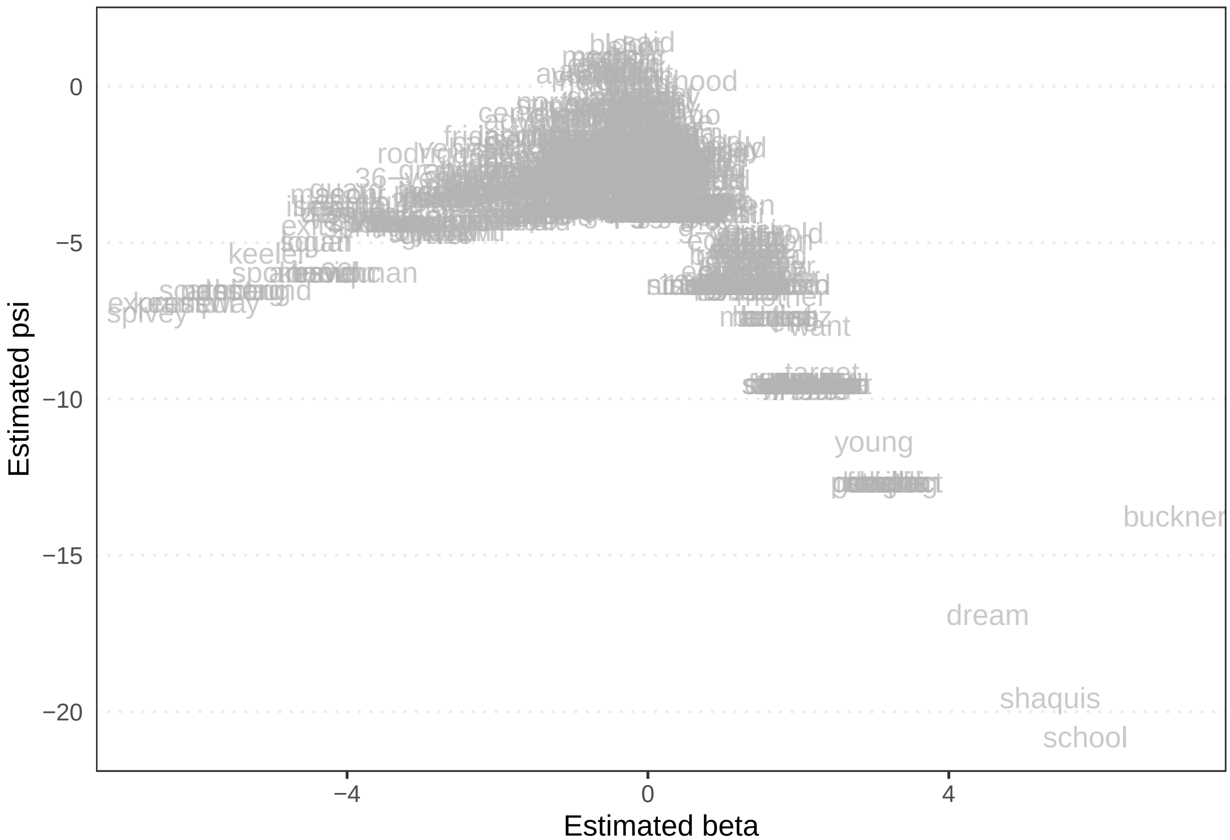
```r
# Store the theta document estimates and se's #
sumwf <- summary(wf)$estimated.document.positions

# Merge the scores and the text together (real good) #
text_scaling <- data.frame(sumwf, nc$victim_text)
colnames(text_scaling)[3] <- "victim_text"

# Sort the data frame (nice and good) #
text_final <- text_scaling[order(text_scaling[["theta"]]),]

# Take a look at the distribution #
hist(text_final$theta)
```




```r
# Look at the distribution more formally
textplot_scale1d(wf)
```




```r
# Then look at the words on either end that pop out
textplot_scale1d(wf, margin = "features")
```


Estimated psi
Estimated beta
0
−5
−10
−15
−20
−4
0
4
young
buckner
dream
shaquis
school
keeler
spivey
want