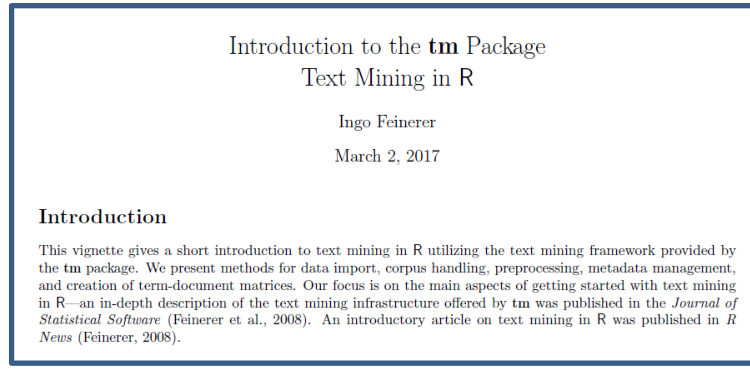
Text mining in R

1. Read Corpus – collection of texts into R.
2. Indexing
   1. Tokenization. Split text into tokens (lowest-level meaningful object of text, typically words).
   2. Pre-processing. Lowercase, remove stopwords, stemming.
   3. Shallow language processing. Term frequency inverse document frequency.
3. Analysis.
   1. Descriptives and classification.

Packages and reading

1. TM
   high-level



Introduction to the **tm** Package
Text Mining in R

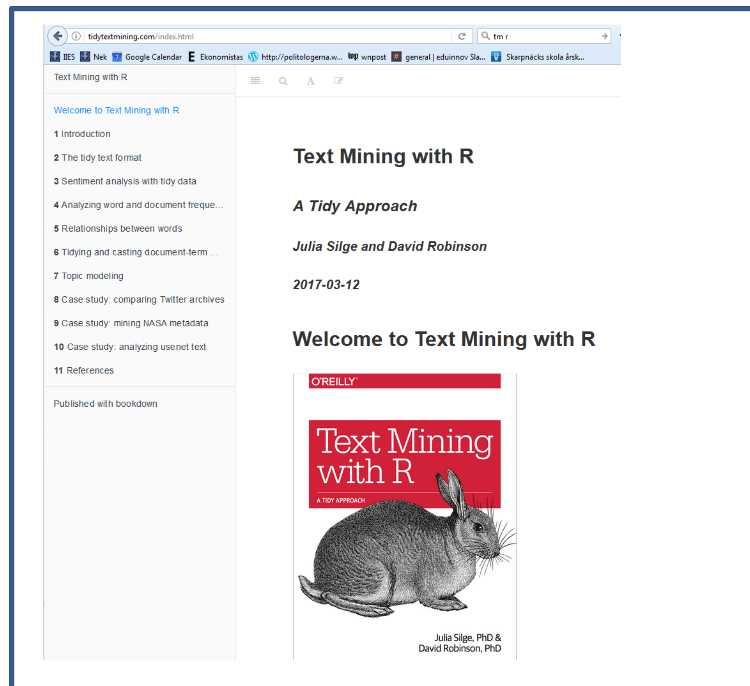
Ingo Feinerer

March 2, 2017

Introduction

This vignette gives a short introduction to text mining in R utilizing the text mining framework provided by the **tm** package. We present methods for data import, corpus handling, preprocessing, metadata management, and creation of term-document matrices. Our focus is on the main aspects of getting started with text mining in R—an in-depth description of the text mining infrastructure offered by **tm** was published in the *Journal of Statistical Software* (Feinerer et al., 2008). An introductory article on text mining in R was published in *R News* (Feinerer, 2008).

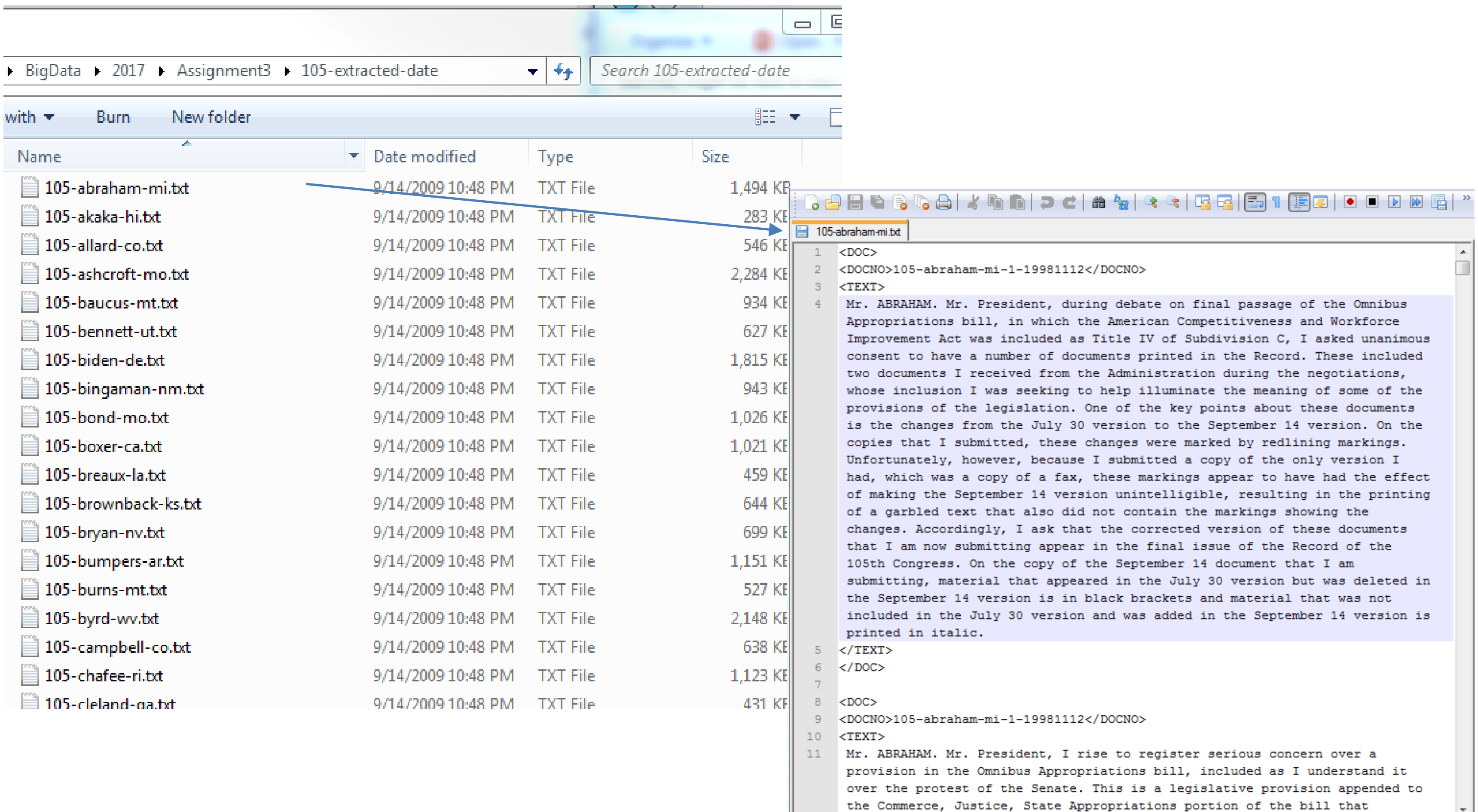
2. Tidytext
   low-level, standard commands.
   https://www.tidytextmining.com



3. Other: stringr, stringi, wordcloud
   dplyr, tidyr
   slam
   SparseM
   e1071

1. Read Corpus Collection of texts into R.

Raw data with documents of senator speeches.



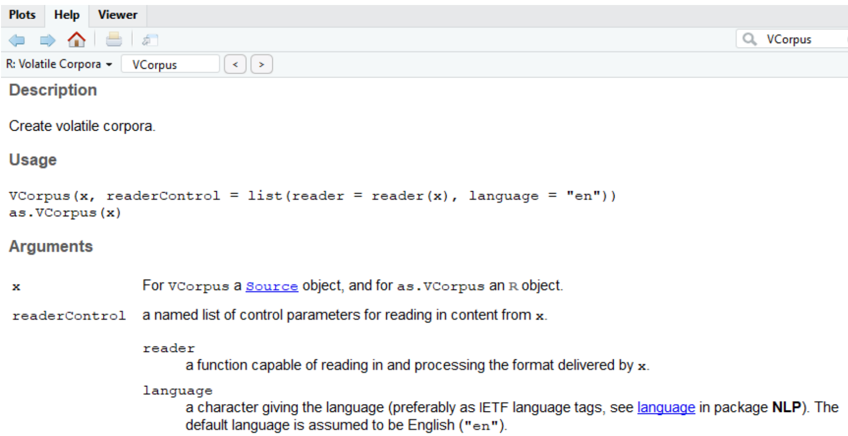
Read Corpus
using VCorpus in tm-package.

Read using Vcorpus command in tm-package.

```
#First load all files into a corpus using tm. The file name is in the variable id.
senator_corpus=VCorpus(DirSource(indir))
```

```
senator_corpus        Large VCorpus (100 elements, 128.4 Mb)
  105-abraham-mi.txt :List of 2
  ..$ content: chr [1:7963] "<DOC>" "<DOCNO>105-abraham-mi-1-19981112</DOCNO>" "<TEXT>" " Mr. ABR…
  ..$ meta :List of 7
  .. ..$ author : chr(0)
  .. ..$ datetimestamp: POSIXlt[1:1], format: "2018-09-21 06:48:35"
  .. ..$ description : chr(0)
  .. ..$ heading : chr(0)
  .. ..$ id : chr "105-abraham-mi.txt"
  .. ..$ language : chr "en"
  .. ..$ origin : chr(0)
  .. ..- attr(*, "class")= chr "TextDocumentMeta"
  ..- attr(*, "class")= chr [1:2] "PlainTextDocument" "TextDocument"
  105-akaka-hi.txt :List of 2
  ..$ content: chr [1:1234] "<DOC>" "<DOCNO>105-akaka-hi-1-19981021</DOCNO>" "<TEXT>" "Mr. AKAKA…
  ..$ meta :List of 7
  .. ..$ author : chr(0)
```

Vcorpus command



Plots Help Viewer

R: Volatile Corpora VCorpus

Description

Create volatile corpora.

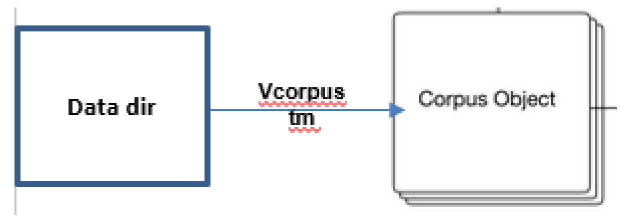
Usage

```
VCorpus(x, readerControl = list(reader = reader(x), language = "en"))
as.VCorpus(x)
```

Arguments

`x`	For `VCorpus` a `Source` object, and for `as.VCorpus` an `R` object.

`readerControl`	a named list of control parameters for reading in content from `x`.
	`reader`
	a function capable of reading in and processing the format delivered by `x`.
	`language`
	a character giving the language (preferably as IETF language tags, see language in package **NLP**). The default language is assumed to be English (`"en"`).



x: DirSource, VectorSource, or DataframeSource.

readerControl:

```
> getReaders()
 [1] "readDataframe"          "readDOC"                     "readPDF"
 [4] "readPlain"              "readRCV1"                    "readRCV1asPlain"
 [7] "readReut21578XML"       "readReut21578XMLasPlain"     "readTagged"
[10] "readXML"
```

2. Tokenization

Raw text. → Text in vector form: one word (token) one row.

105-abraham-mi.txt

```
<DOC>
<DOCNO>105-abraham-mi-1-19981112</DOCNO>
<TEXT>
 Mr. ABRAHAM. Mr. President, during debate on final passage of the Omnibus
 Appropriations bill, in which the American Competitiveness and Workforce
 Improvement Act was included as Title IV of Subdivision C, I asked unanimous
 consent to have a number of documents printed in the Record. These included
 two documents I received from the Administration during the negotiations,
 whose inclusion I was seeking to help illuminate the meaning of some of the
 provisions of the legislation. One of the key points about these documents
 is the changes from the July 30 version to the September 14 version. On the
 copies that I submitted, these changes were marked by redlining markings.
 Unfortunately, however, because I submitted a copy of the only version I
 had, which was a copy of a fax, these markings appear to have had the effect
 of making the September 14 version unintelligible, resulting in the printing
 of a garbled text that also did not contain the markings showing the
 changes. Accordingly, I ask that the corrected version of these documents
 that I am now submitting appear in the final issue of the Record of the
 105th Congress. On the copy of the September 14 document that I am
 submitting, material that appeared in the July 30 version but was deleted in
 the September 14 version is in black brackets and material that was not
 included in the July 30 version and was added in the September 14 version is
 printed in italic.
</TEXT>
</DOC>

<DOC>
<DOCNO>105-abraham-mi-1-19981112</DOCNO>
<TEXT>
 Mr. ABRAHAM. Mr. President, I rise to register serious concern over a
 provision in the Omnibus Appropriations bill, included as I understand it
 over the protest of the Senate. This is a legislative provision appended to
 the Commerce, Justice, State Appropriations portion of the bill that
```

	id	word	row
1	abraham-mi	doc	1
2	abraham-mi	docno	2
3	abraham-mi	105	3
4	abraham-mi	abraham	4
5	abraham-mi	mi	5
6	abraham-mi	1	6
7	abraham-mi	19981112	7
8	abraham-mi	docno	8
9	abraham-mi	text	9
10	abraham-mi	mr	10
11	abraham-mi	abraham	11
12	abraham-mi	mr	12
13	abraham-mi	president	13
14	abraham-mi	during	14
15	abraham-mi	debate	15

From Corpus to word vector.

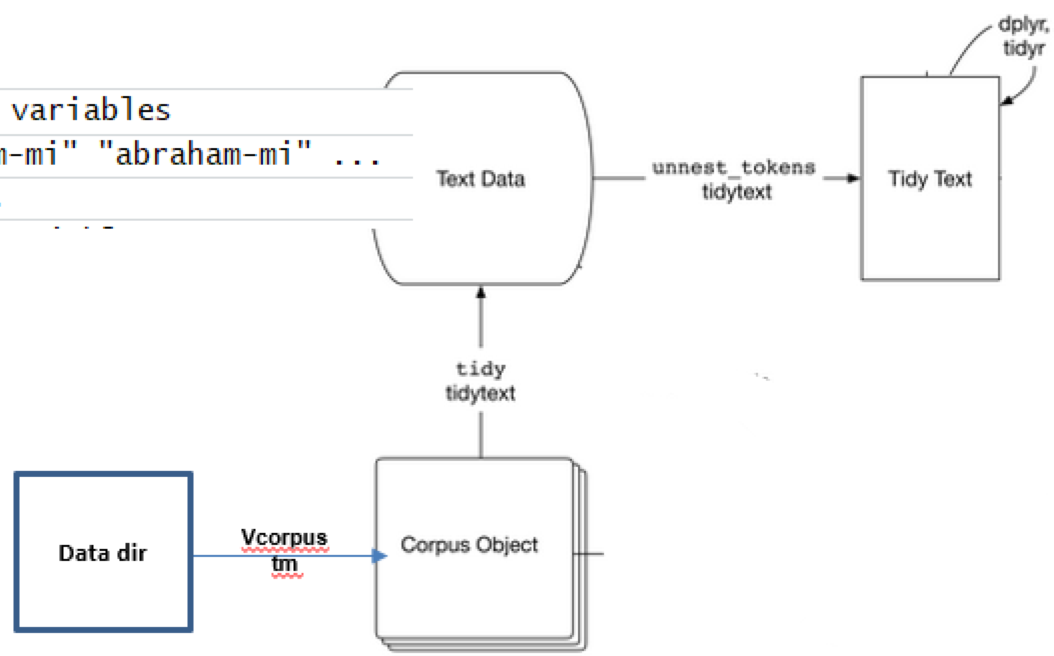
- tidy(): tidytext package
  - constructs a table (tibble) with one row per document, including the metadata (such as id) as columns alongside the text (in variable called “text”).
  - tibble is data frame format (in dplyr) that do not convert strings to factors.
- unnest_tokens(word,text)
  - splits texts into one-token-per row.
  - punctuation stripped.
  - converts tokens to lowercase by default.

From Corpus to word vector.

```
# Unnest tokens is a tokenizer which splits sentences to words.
senators_td2 = senator_corpus %>%
  tidy() %>%
  select(id, text) %>%
  mutate(id=str_match(id,"-(.*).txt")[,2]) %>%
  unnest_tokens(word, text)  %>%
  group_by(id) %>%
  mutate(row=row_number()) %>%
  ungroup()
```

```
senators_td       19247013 obs. of 2 variables
  id : chr "abraham-mi" "abraham-mi" "abraham-mi" "abraham-mi" ...
  word: chr "doc" "docno" "105" "abraham" ...
```

	id	word
1	abraham-mi	doc
2	abraham-mi	docno
3	abraham-mi	105
4	abraham-mi	abraham
5	abraham-mi	mi
6	abraham-mi	1



Pre-processing.

```r
# First load the senator party labels.
sen105_party <- read.csv("../sen105_party.csv", stringsAsFactors=FALSE)

# Create a data frame with senator names in lower case.
names = sen105_party %>%
  mutate(word=tolower(lname)) %>%
  select(word)

# Create a data frame with state names in lower case.
states = as.data.frame(c(tolower(state.abb),tolower(state.name)))
colnames(states) <- "word"
```

	party	id
1	200	sessions-al
2	200	shelby-al
3	200	murkowski-ak

	word
1	sessions
2	shelby
3	murkowski

	word
1	al
2	ak
3	az

```r
# Remove non-alphabetic characters, stopwords, senator and state names
droplist=c("text","doc","docno")
senators_td2 = senators_td2 %>%
  mutate(word = str_extract(word, "[a-z']+")) %>%
  drop_na(word)   %>%
  filter(!(word %in% droplist)) %>%
  anti_join(stop_words) %>%
  anti_join(names) %>%
  anti_join(states)
```

	word	lexicon
1	a	SMART
2	a's	SMART
3	able	SMART
4	about	SMART
5	above	SMART
6	according	SMART

Other tokens: bigrams and trigrams

```
# Create bigrams
senators_bigram = senators_td2 %>%
  arrange(id,row)  %>%
  group_by(id)  %>%
  mutate(bigram=str_c(lag(word,1),word,sep=" ")) %>%
  filter(row==lag(row,1)+1)  %>%
  select(-word)  %>%
  ungroup()

# Create trigrams
senators_trigram = senators_td2 %>%
  arrange(id,row)  %>%
  group_by(id)  %>%
  mutate(trigram=str_c(lag(word,2),lag(word,1),word,sep=" ")) %>%
  filter(row==lag(row,1)+1 & lag(row,1)==lag(row,2)+1)  %>%
  select(-word)  %>%
  ungroup()
```

Keep adjacent observations within senator

3. Shallow language processing.

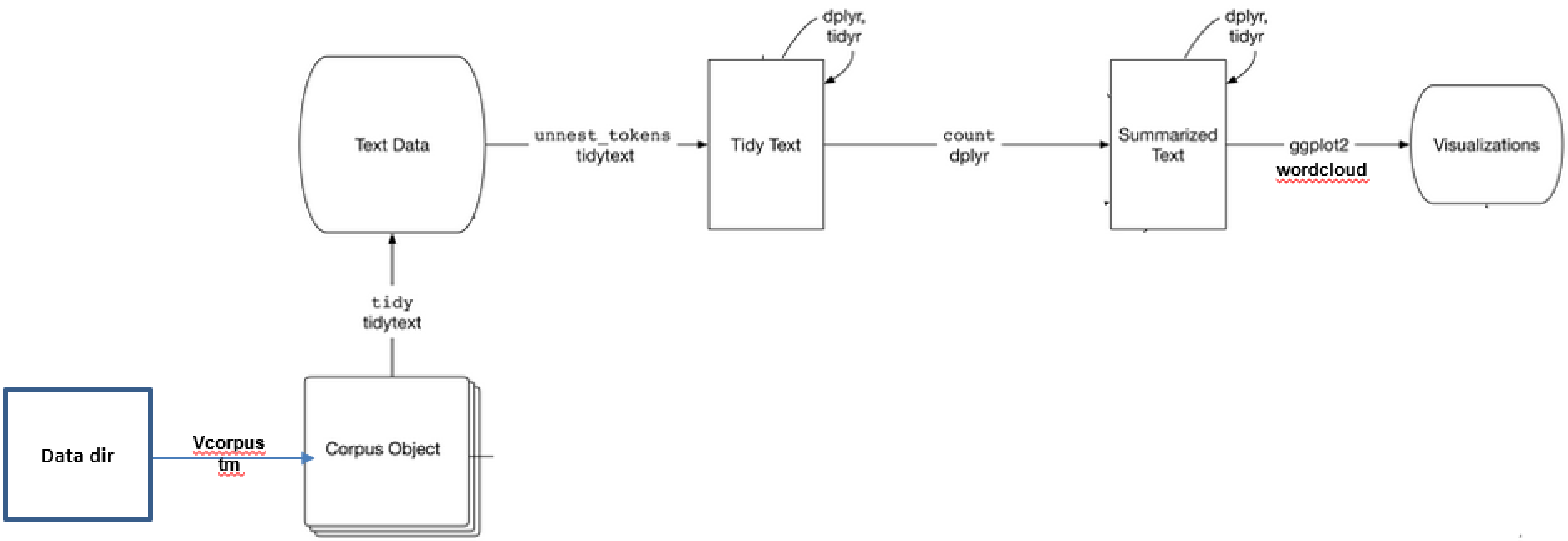
Total word frequencies

```
> # Create an overall word-frequency list
> wordlist= senators_td2 %>%
+   count(word,sort=TRUE)
> wordlist
# A tibble: 65,137 x 2
         word     n
        <chr> <int>
1   president 89492
2     senator 59391
3        bill 55967
4   amendment 45208
5      senate 38915
6        time 38797
7      people 38275
8     federal 27341
9 legislation 27267
10  committee 24882
# ... with 65,127 more rows
```

```
> bigramlist= senators_bigram %>%
+   count(bigram,sort=TRUE)
> bigramlist
# A tibble: 741,550 x 2
   bigram                  n
   <chr>               <int>
 1 unanimous consent   13278
 2 social security      6384
 3 health care          5777
 4 federal government   5245
 5 american people      5115
 6 balanced budget      4977
 7 madam president      3845
 8 majority leader      2938
 9 appropriations bill  2721
10 child care           2568
# ... with 741,540 more rows
```

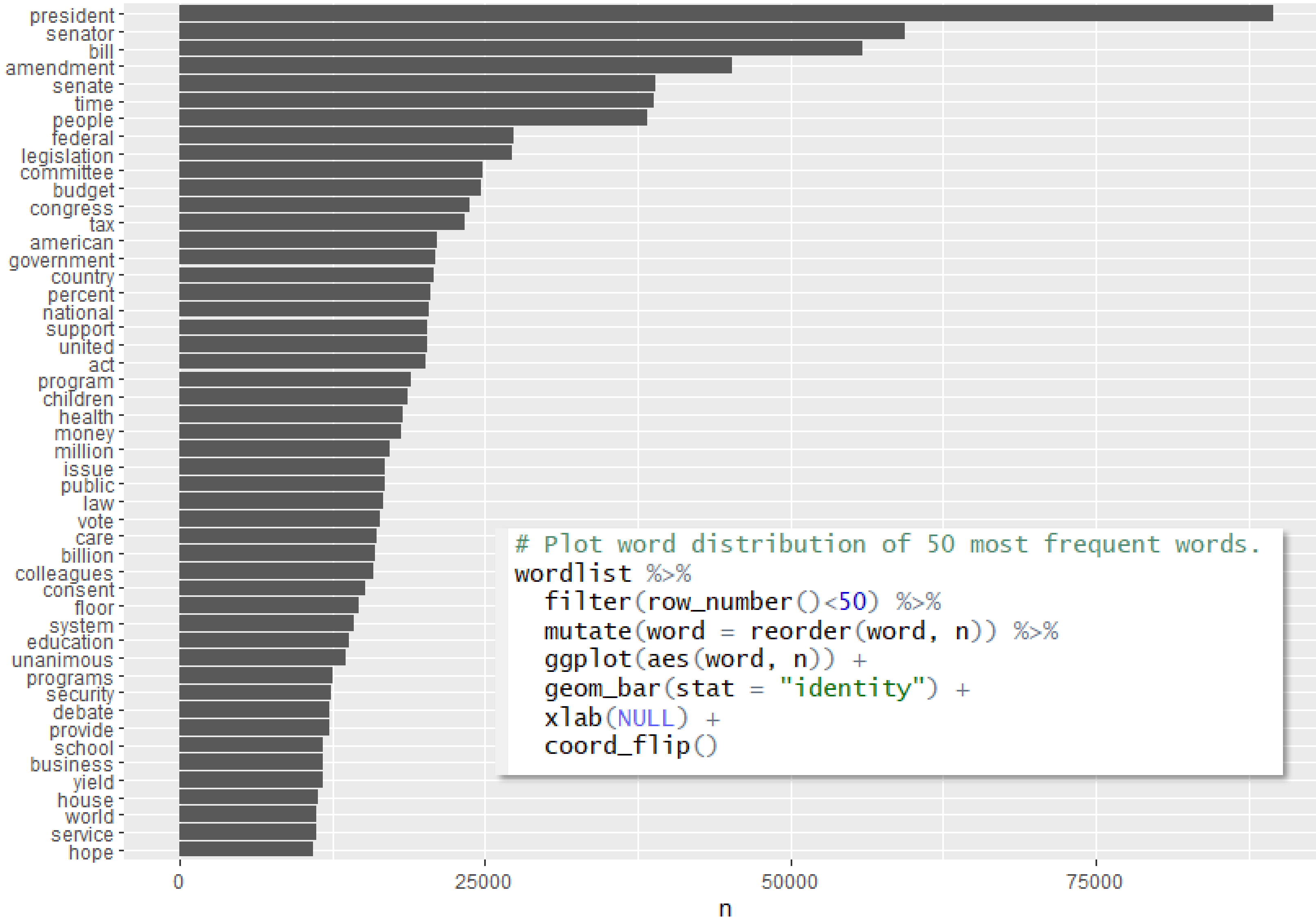
```
> trigramlist= senators_trigram %>%
+   count(trigram,sort=TRUE)
> trigramlist
# A tibble: 458,269 x 2
   trigram                        n
   <chr>                      <int>
 1 balanced budget amendment   1974
 2 campaign finance reform     1525
 3 federal debt stood          1115
 4 world war ii                 930
 5 armed services committee     921
 6 partial birth abortion       713
 7 internal revenue service     709
 8 social security trust        696
 9 line item veto               695
10 foreign relations committee  665
# ... with 458,259 more rows
```

3. Shallow language processing.



Zipf’s law: word frequency approx 1/n.

Words like president is not very informative since every document contains it.




```r
# Plot word distribution of 50 most frequent words.
wordlist %>%
  filter(row_number()<50) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_bar(stat = "identity") +
  xlab(NULL) +
  coord_flip()
```

Tf-idf

```
#Compute word frequency, by senator
wordlist_s <- senators_td2 %>%
  inner_join(sen105_party) %>%
  count(id, party, word, sort=TRUE) %>%
  ungroup()

#Compute tf-idf, each senator is a "document"
wordlist_s <- wordlist_s %>%
  bind_tf_idf(word, id, n)
```

	id	party	word	n	share	tf	idf	tf_idf
1	lott-ms	200	president	3030	0.020277460	0.020277460	0	0
2	lott-ms	200	senate	2780	0.018604402	0.018604402	0	0
3	lott-ms	200	senator	2560	0.017132111	0.017132111	0	0
4	wellstone-mn	100	people	2355	0.016220796	0.016220796	0	0

”president” used by all senators: idf=0.

	id	party	word	n	share	tf	idf	tf_idf
824885	akaka-hi	100	hawaii's	45	0.0021577559	0.0021577559	3.21887582	0.006945548
824884	dewine-oh	200	haitian	196	0.0024831501	0.0024831501	2.12026354	0.005264933
824883	conrad-nd	100	forks	166	0.0025595165	0.0025595165	1.83258146	0.004690522
824882	wellstone-mn	100	blanca	177	0.0012191426	0.0012191426	3.50655790	0.004274994
824881	levin-mi	100	atr	105	0.0010871704	0.0010871704	3.91202301	0.004253035
824880	akaka-hi	100	monk	31	0.0014864541	0.0014864541	2.81341072	0.004182006

3. Analysis

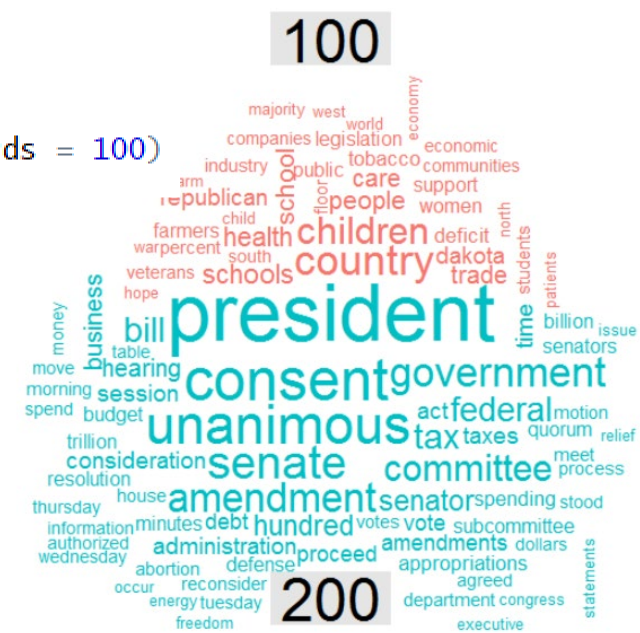
- Descriptive: frequency by party.
- Sentiment analysis.
- Classification (SVM).

Descriptive: Frequencies by party

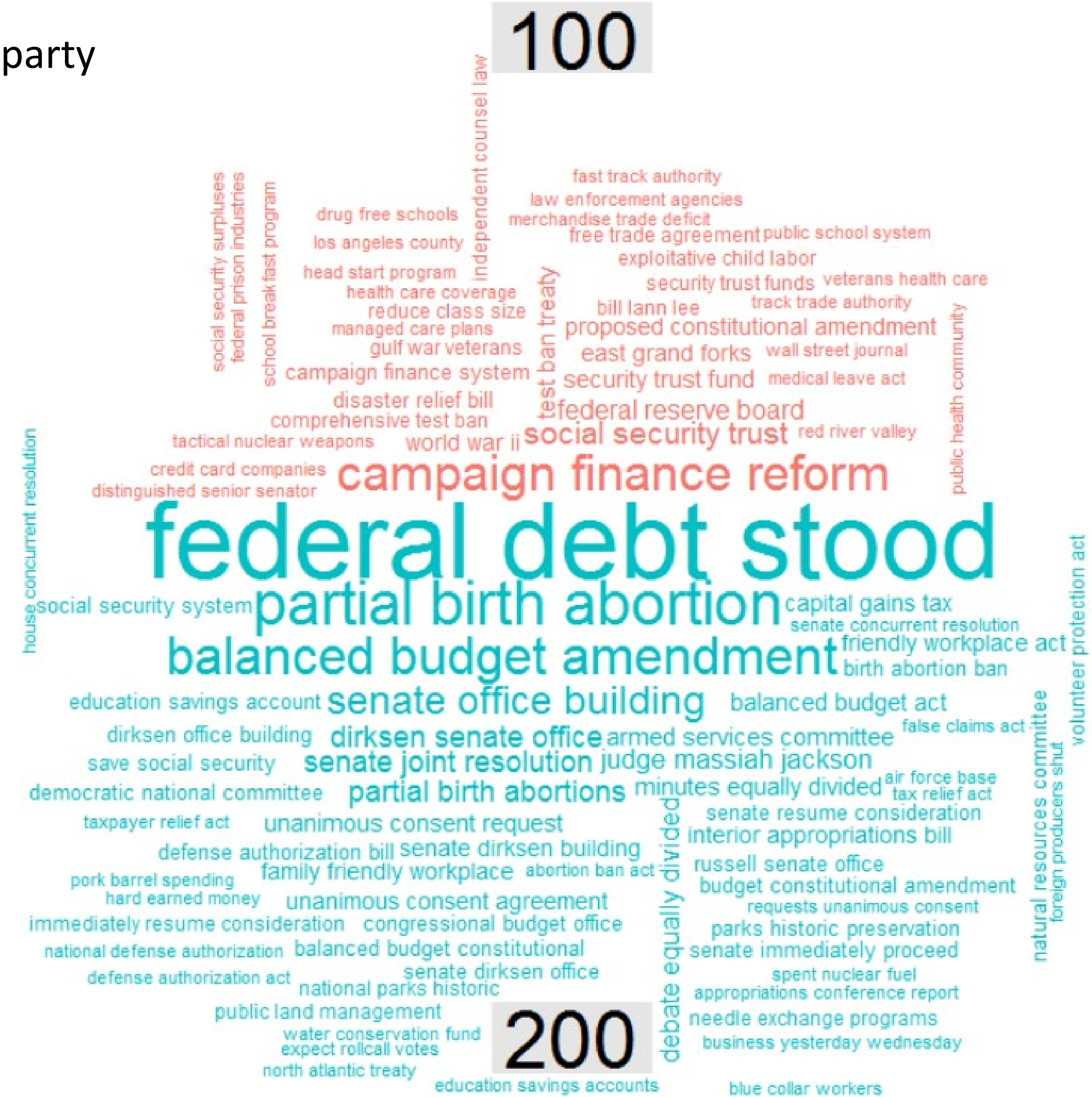
```r
wordlist_p <- senators_td2 %>%
  inner_join(sen105_party) %>%
  rename(word=trigram) %>%
  count(party, word, sort=TRUE) %>%
  group_by(party) %>%
  mutate(share = n / sum(n), rank=row_number())  %>%
  ungroup()
```

```
   party      word     n       share  rank
   <int>     <chr> <int>       <dbl> <int>
1    200 president 51613 0.014386891     1
2    100 president 37879 0.011503566     1
3    200   senator 32291 0.009000971     2
4    200      bill 30986 0.008637208     3
5    100   senator 27100 0.008230065     2
6    200 amendment 25756 0.007179369     4
7    100      bill 24981 0.007586541     3
8    200    senate 22907 0.006385223     5
9    200      time 21165 0.005899648     6
10   100 amendment 19452 0.005907425     4
# ... with 98,724 more rows
```

```r
#Wordcloud, by party
library(reshape2)
wordlist_p %>%
  select(word,party, n)  %>%
  acast(word ~ party, value.var = "n", fill = 0)  %>%
  comparison.cloud(colors = c("#F8766D", "#00BFC4"),max.words = 100)
```



Trigrams by party



Sentiment analysis:
Wordcount using the sentiments lexicons in tidytext.

```
> # Sentiments, word count
> library(tidytext)
> sentiments
# A tibble: 23,165 x 4
          word sentiment lexicon score
         <chr>     <chr>   <chr> <int>
1       abacus     trust     nrc    NA
2      abandon      fear     nrc    NA
3      abandon  negative     nrc    NA
4      abandon   sadness     nrc    NA
5    abandoned     anger     nrc    NA
6    abandoned      fear     nrc    NA
7    abandoned  negative     nrc    NA
8    abandoned   sadness     nrc    NA
9  abandonment     anger     nrc    NA
10 abandonment      fear     nrc    NA
# ... with 23,155 more rows
```

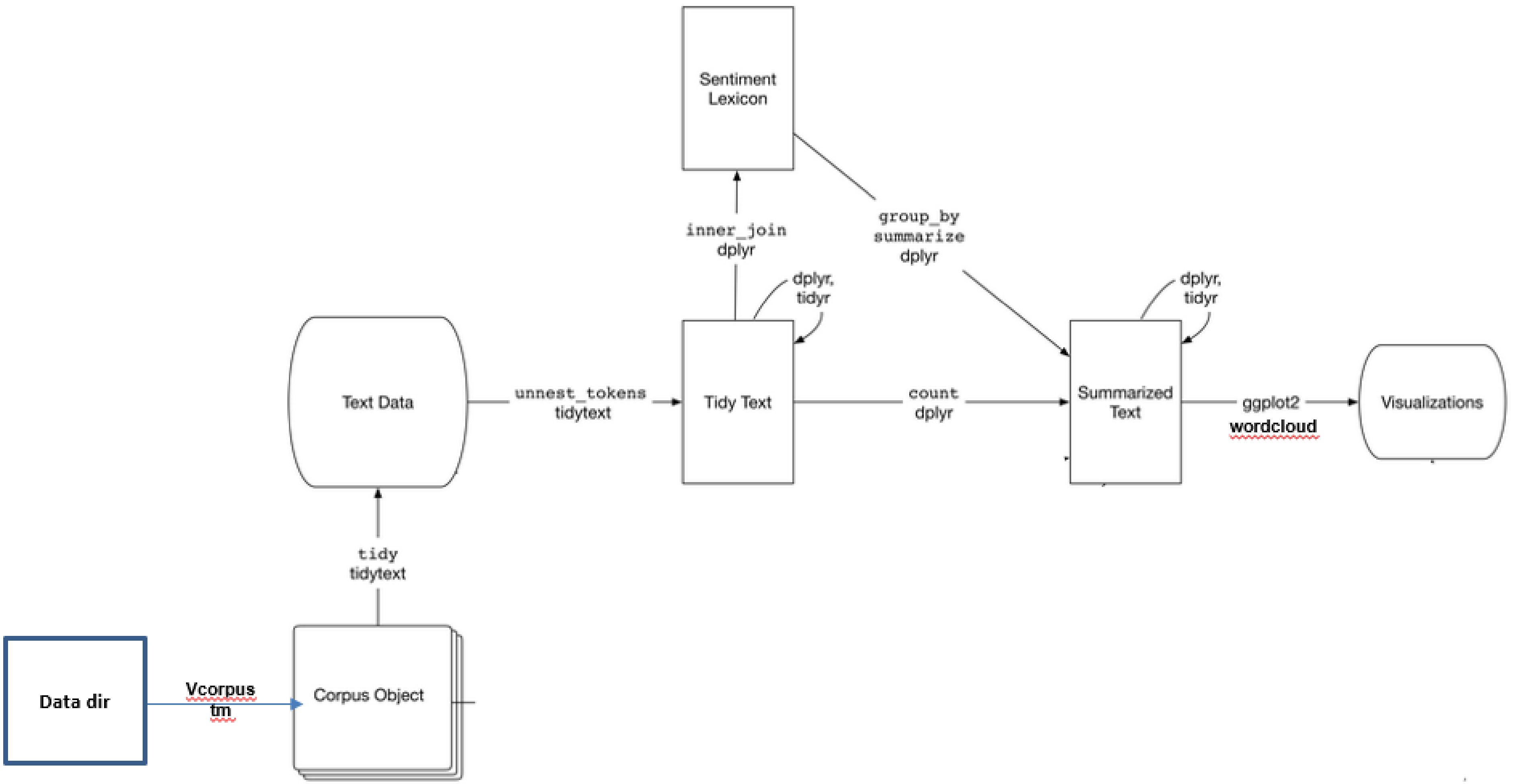
```
> table(lexicon)
lexicon
AFINN  bing   nrc
 2476  6788 13901
> table(sentiment[lexicon=="nrc"])

       anger anticipation      disgust         fear
        1247          839         1058         1476
         joy     negative     positive      sadness
         689         3324         2312         1191
    surprise        trust
         534         1231
> table(sentiment[lexicon=="bing"])

negative positive
    4782     2006
> table(score[lexicon=="AFINN"])

 -5  -4  -3  -2  -1   0   1   2   3   4   5
 16  43 264 965 309   1 208 448 172  45   5
```

Sentiment analysis: implement by merge

```
> get_sentiments("nrc")[1:10,]
# A tibble: 10 x 2
          word sentiment
         <chr>     <chr>
1       abacus     trust
2      abandon      fear
3      abandon  negative
4      abandon   sadness
5    abandoned     anger
6    abandoned      fear
7    abandoned  negative
8    abandoned   sadness
9  abandonment     anger
10 abandonment      fear
```

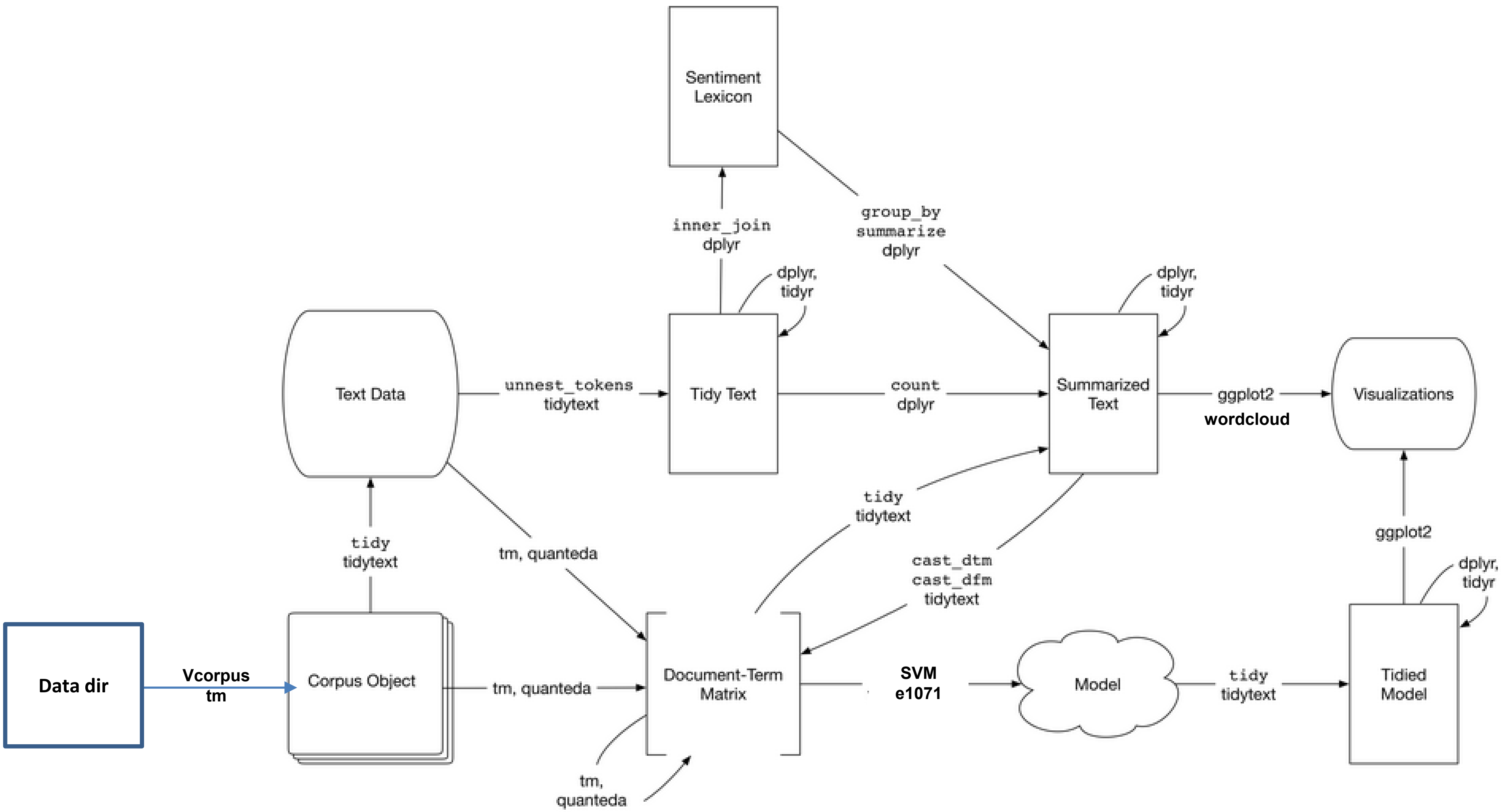
```
> wordlist_s %>%
+   inner_join(get_sentiments("nrc")) %>%
+   group_by(party) %>%
+   mutate(total=sum(n)) %>%
+   group_by(party, sentiment) %>%
+   summarise(n2=sum(n/total)) %>%
+   spread(party, n2)
Joining, by = "word"
# A tibble: 10 x 3
      sentiment      `100`      `200`
*         <chr>      <dbl>      <dbl>
1         anger 0.05536695 0.05335390
2  anticipation 0.10183271 0.10087415
3       disgust 0.03113449 0.02888124
4          fear 0.06904045 0.06815292
5           joy 0.06772064 0.06639535
6      negative 0.11874833 0.11591421
7      positive 0.26543140 0.27106587
8       sadness 0.05392743 0.05254787
9      surprise 0.03358197 0.03209497
10        trust 0.20321562 0.21071953
```

Sentiment analysis: implement by merge



3. Analysis

- Lasso-logit
  - glmnet
- (Support Vector Machines in R.)
  - e1071 library: svm() function
    - y-variable must be coded as factor.
    - We will specify the x-variables
      as a document (senator) – term matrix
    - Parameters
      - kernel="linear”
      - cost argument: selected by tune() that performs ten-fold cross-validation on a set of models



Sentiment Lexicon
inner_join
dplyr
group_by
summarize
dplyr
dplyr, tidyr
dplyr, tidyr
Text Data
unnest_tokens
tidytext
Tidy Text
count
dplyr
Summarized Text
ggplot2
wordcloud
Visualizations
tidy
tidytext
tidy
tidytext
tm, quanteda
cast_dtm
cast_dfm
tidytext
ggplot2
dplyr, tidyr
Data dir
Vcorpus
tm
Corpus Object
tm, quanteda
Document-Term Matrix
SVM
e1071
Model
tidy
tidytext
Tidied Model
tm, quanteda

Document-Term Matrix Conversion

tibble – DocumentTermMatrix – sparse matrix

```
#Load a DocumentTermMatrix
data("AssociatedPress",package="topicmodels")
AssociatedPress

# 1. dtm -> tibble
#Convert this spart matrics (DocumentTermMatrix in the tm package)
# into a tibble.
ap_td <-tidy(AssociatedPress)
ap_td
```

```
# A tibble: 302,031 x 3
   document term       count
      <int> <chr>      <dbl>
 1        1 adding         1
 2        1 adult          2
 3        1 ago            1
 4        1 alcohol        1
 5        1 allegedly      1
 6        1 allen          1
 7        1 apparently     2
 8        1 appeared       1
 9        1 arrested       1
10        1 assault        1
# ... with 302,021 more rows
```

```
# 3. tibble -> sparse matrix
m <- ap_td %>%
  cast_sparse(document,term, count)
```

```
# 2. tibble -> dfm document term matrix
ap_td %>%
  cast_dtm(document, term, count)
```

```
<<DocumentTermMatrix (documents: 2246, terms: 10473)>>
Non-/sparse entries: 302031/23220327
Sparsity           : 99%
Maximal term length: 18
Weighting          : term frequency (tf)
```

4. tm/Corpus - tidy

```
senator_corpus=VCorpus(DirSource(indir))

#Then turn the data into a tidy text document.
# Unnest tokens is a tokenizer which splits sentences to words.
senators_td = senator_corpus %>%
  tidy() %>%
```

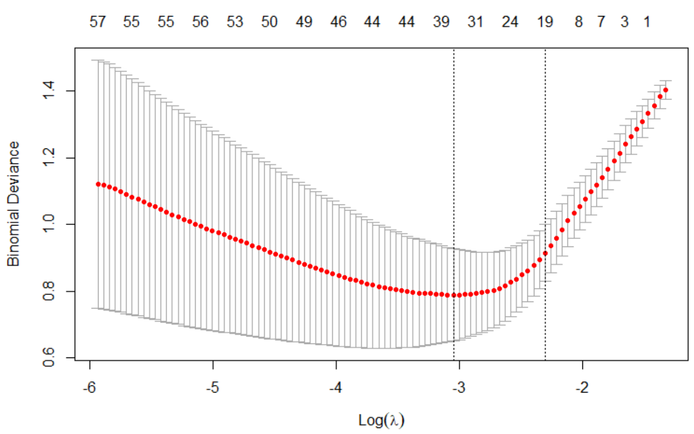
Prepare x-matrix as sparse and y as factor

```
#Compute trigram frequency, by senator
wordlist_s3 <- senators_trigram %>%
  rename(word=trigram) %>%
  inner_join(sen105_party) %>%
  count(id, party, word, sort=TRUE) %>%
  ungroup()
```

```
# For SVM analysis
# Cast text into a Matrix object
s <- wordlist_s3 %>%
  cast_sparse(id, word, n)
class(s)

# Order rows by row names "abraham-mi", "akaka-hi",... to match ordering in y
s=s[order(rownames(s)),]


#generate dependent var
y=sen105_party[order(sen105_party$id),]
y <- as.matrix(y$party)
y <- as.factor(y)
```

Estimate Lasso logit

```
#lasso
library(glmnet)

# Choosing lambda that minimizes MSE:
cv_lasso <- cv.glmnet(s_train,y_train, alpha = 1, family="binomial")
plot(cv_lasso)
```




```
# # Using whole data with lambda chosen above, and saving coefficients:
lasso_pred <- predict(cv_lasso, newx=s_test, s ="lambda.min" )
lasso_pred <- ifelse(lasso_pred<0,0,1)
table(predict =lasso_pred , truth= y_test )
```

```
        truth
predict 100 200
      0  10   2
      1   0   8
```

Trigrams most predictive of party

```
# Using whole data with lambda chosen above, and saving coefficients:
cv_lasso <- cv.glmnet(s,y, alpha = 1, family="binomial")
lasso_best <- predict(cv_lasso, s ="lambda.min", type = "coefficients")
lasso_coef <- as.matrix(coef(cv_lasso, s = "lambda.min"))
coef_lasso <- data.frame(names = lasso_best@Dimnames[[1]][lasso_best@i+1], coefficients = lasso_best@x)
```

names	coefficients
nuclear weapons nuclear	1.520254e+00
anticipate rollcall votes	1.433015e+00
conference committee deliberations	1.367938e+00
unfunded federal mandates	1.304774e+00
clinton tax increase	1.040175e+00
weekly policy luncheons	9.502650e-01
executive items cleared	9.046558e-01
russian arms control	8.810411e-01
requests unanimous consent	8.692695e-01
supported credit unions	8.679496e-01
dirksen office building	7.330411e-01
life threatening health	6.746818e-01
income tax treated	5.045973e-01
clinger cohen act	4.999213e-01
majority leader trent	2.993568e-01
campaign financing issues	2.712431e-01
federal retirement benefits	2.488765e-01

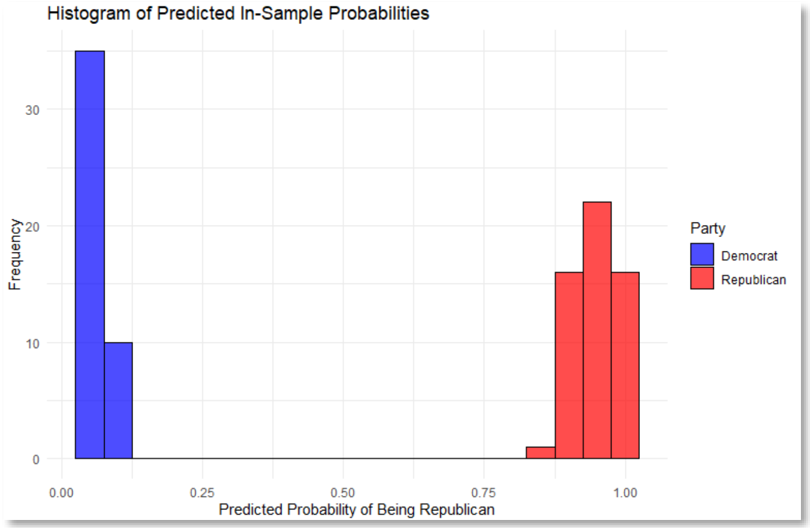
names	coefficients
blue ribbon panel	-2.926350e-01
federal campaign finance	-2.914492e-01
public health research	-2.603931e-01
low inflation low	-2.223530e-01
american chemical companies	-1.990252e-01
sewage treatment plants	-1.852173e-01
chief executive officers	-1.661542e-01
single republican vote	-1.014517e-01
senate floor debating	-6.621919e-02
comprehensive campaign finance	-6.374862e-02
tobacco control legislation	-5.538079e-02
civil rights movement	-4.554250e-02
democratic leader senator	-3.749014e-02
minority owned businesses	-3.492129e-02
day care center	-2.008378e-02
democratic national convention	-1.210135e-02
patient protection act	-9.209256e-03

Pr(Republican|language)

```
# Predicted probability of senator being Republican
lasso_pred <- predict(cv_lasso, newx=s, s ="lambda.min", type="response")
sen_lasso <- as.data.frame(lasso_pred) %>%
  rename(PrRep=lambda.min) %>%
  merge(sen105_party, by.x="row.names", by.y="id")
```

Row.names	PrRep	party
levin-mi	0.03080366	100
dorgan-nd	0.03248301	100
hollings-sc	0.03808736	100
moynihan-ny	0.03819922	100
leahy-vt	0.04210428	100
dodd-ct	0.04228984	100
lautenberg-nj	0.04236770	100
kerrey-ne	0.04266700	100
johnson-sd	0.04679090	100
wellstone-mn	0.04993091	100
conrad-nd	0.05055045	100
kennedy-ma	0.05587330	100
feingold-wi	0.05627624	100
boxer-ca	0.05668981	100
ford-ky	0.05670182	100
harkin-ia	0.05673786	100

Row.names	PrRep	party
lott-ms	1.00000000	200
coverdell-ga	0.99999279	200
chafee-ri	0.99999103	200
stevens-ak	0.99999059	200
jeffords-vt	0.99994165	200
domenici-nm	0.99984318	200
gorton-wa	0.99959099	200
coats-in	0.99907610	200
roth-de	0.99795665	200
mccain-az	0.99786834	200
grams-mn	0.98888677	200
cochran-ms	0.98364616	200
thurmond-sc	0.98297003	200
craig-id	0.98280564	200
ashcroft-mo	0.98273647	200
sessions-al	0.98252005	200



Estimate SVM

```
> svmfit=svm(s,y,kernel="linear", cost=.1)
> summary(svmfit)

Call:
svm.default(x = s, y = y, kernel = "linear", cost = 0.1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1
      gamma:  2.182124e-06

Number of Support Vectors:  93

 ( 50 43 )


Number of Classes:  2

Levels:
 100 200
```

Set tuning parameter

```
> set.seed(1)
> tune.out=tune(svm ,s,y ,kernel ="linear", ranges =list(cost=c(0.00001, 0.0001, 0.001 , 0.01, 0.1, 1) ))
> summary(tune.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
  cost
 0.001

- best performance: 0.28

- Detailed performance results:
   cost error dispersion
1 1e-05  0.45  0.1581139
2 1e-04  0.32  0.1475730
3 1e-03  0.28  0.1475730
4 1e-02  0.28  0.1475730
5 1e-01  0.28  0.1475730
6 1e+00  0.28  0.1475730

> bestmod =tune.out$best.model
> ypred=predict(bestmod,s)
> table(predict =ypred , truth= y )
       truth
predict 100 200
    100  45   0
    200   0  55
```

Retrieve beta-coefficients

$$\widehat{\beta} = \sum_{i=1}^{n} \widehat{\alpha}_i y_i x_i$$

```
#svmfit$coefs: the svm alpha's (signed by yi)
#svmfit$coefs the indices of the observations to which the alphas belong
#beta = sum x_i alpha_i

beta=drop(t(bestmod$coefs)%*%as.matrix(s)[bestmod$index,])
beta=as.data.frame(beta)
```

	beta
campaign finance reform	-0.009416556
world war ii	-0.007504979
test ban treaty	-0.006341214
el camino real	-0.005978151
social security trust	-0.005900607

senate dirksen office	0.006196373
debate equally divided	0.006375586
social security system	0.007054818
capital gains tax	0.008592982
senate office building	0.008645145
partial birth abortion	0.010126529

Senators with most ideological language

```
#Get distance from hyperplane for each senator.
pred <- predict(bestmod, s, decision.values = TRUE)
dist<-as.data.frame(attr(pred, "decision.values"))
sen_dist<-arrange(sen105_party,id)
sen_dist <- merge(sen_dist,dist,by.x = "row.names", by.y = "row.names")
```

party	id	200/100
100	dorgan-nd	-1.7261673
100	feingold-wi	-1.5034754
100	ford-ky	-1.0002059
100	bryan-nv	-1.0001969

200	ashcroft-mo	1.6163157
200	santorum-pa	1.6557853
200	hatch-ut	2.0381536
200	lott-ms	2.0437390

Text analysis packages in R

1: tm

Introduction to the **tm** Package
Text Mining in R

Ingo Feinerer

March 2, 2017

Introduction

This vignette gives a short introduction to text mining in R utilizing the text mining framework provided by the **tm** package. We present methods for data import, corpus handling, preprocessing, metadata management, and creation of term-document matrices. Our focus is on the main aspects of getting started with text mining in R—an in-depth description of the text mining infrastructure offered by **tm** was published in the *Journal of Statistical Software* (Feinerer et al., 2008). An introductory article on text mining in R was published in *R News* (Feinerer, 2008).

```
library(slam)
library(data.table)
library(e1071)
library(tm)
library(dplyr)
library(wordcloud)

rm(list = ls())

setwd('E:/c_old/DavidD/Courses/BigData/OtherMaterial/tm')
sendir <- 'E:/c_old/David/Projects/Religion/Data/Sen_text/text/105-extracted-date'
sen <- Corpus(DirSource(sendir))
summary(sen)

# Remove extra whitespace, lowercase, stopwords, stem:
sen <- tm_map(sen, stripWhitespace)
sen <- tm_map(sen, tolower)
sen <- tm_map(sen, removeWords, stopwords("english"))
sen <- tm_map(sen, stemDocument)

# Create term-document matrix
dtm <- DocumentTermMatrix(sen)
inspect(dtm[1:2,100:105])

# Read senator data (one variable has file names, e.g. "105-abraham-mi.txt"
pcafile <- "E:/c_old/DavidD/Courses/BigData/2016/Rearranged/Part3_MachineLearning/L6/ProblemSet/pca/sen105kh_pc1.txt"
senators <- read.csv(pcafile)

#generate dependent var. First extract column names, then add values.
rows=as.data.frame(rownames(dtm))
names(rows)<-c("doc")
sen_p=merge(rows,senators, by = "doc", all.x=TRUE)
y <- as.matrix(sen_p$party)
y <- as.factor(y)
```

Topic Models

Unsupervised learning: Motivating questions:

- What are the topics that a document is about?
- How do topics change over time (Hansen et al., 2018)?
- How can we reduce the dimensionality when describing documents?

References:

- D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. Journal of Machine Learning Research, 3:993–1022, January 2003.
- D. Blei and J. Lafferty. Topic Models. In A. Srivastava and M. Sahami, editors, Text Mining: Theory and Applications. Taylor and Francis, 2009.
- Hansen, Stephen, Michael McMahon, and Andrea Prat. "Transparency and deliberation within the FOMC: a computational linguistics approach." The Quarterly Journal of Economics 133.2 (2018): 801-870.
- https://www.tidytextmining.com/topicmodeling.html

Latent Dirichlet Allocation: DGP

We have D documents, a vocabulary of V words, and K topics.

- Every document d is a mixture of topics.
  - A speech is $\theta_{d,k}$ =80% about inflation and 20% about employment.
- A topic k is a probability distribution over words v, $\beta_{k,v}$, e.g.

	price	increase	wage	employ
Inflation	1/3	1/3	1/6	1/6
Employment	1/6	1/6	1/3	1/3

- For each topic 1...K, draw a multinomial over words $\beta_k \sim Dir(\eta)$.
- For each document 1...D, draw a multinomial over topics $\theta_d \sim Dir(\alpha)$.

Example: Senator speeches

Speeches as the unit of observation (instead of senator).

```
# Gen id variable = senator + docno
senators_td2 = senators_td2[!is.na(senators_td2$word),]
senators_td2 = senators_td2 %>%
  mutate(d = cumsum(word=="docno"))

# Remove particular words and missing values
droplist=c("text","doc","docno", "")
senators_td2 = senators_td2[!(senators_td2$word %in% droplist),]

# Generate speech indicator.
senators_td2 = senators_td2 %>%
  mutate(x=ifelse(d!=lag(d,1) | id!=lag(id,1), 1,0)) %>%
  mutate(speech = cumsum( ifelse(is.na(lag(d,1)),0,x)) )
```

	id	word	row	d	x	speech
86	abraham-mi	printed	251	2	0	0
87	abraham-mi	italic	253	2	0	0
88	abraham-mi	president	268	4	1	1
89	abraham-mi	rise	270	4	0	1

Compute word frequencies per speech

```
#Compute word frequency, by speech
wordlist_s <- senators_td2 %>%
  count(speech, word, sort=TRUE) %>%
  ungroup()

# Remove rarely used words
wordlist= senators_td2 %>%
count(word,sort=TRUE)
wordlist

wordlist_m50 <- wordlist %>%
  filter(n>50) %>%
  select(word)

wordlist_s <- wordlist_s  %>%
  inner_join(wordlist_m50)

# Cast text into a Matrix object
s <- wordlist_s %>%
  cast_sparse(speech, word, n)
class(s)
```

Run topic model and extract beta_k

```
# Run a topic model with k=10 topics.
# set a seed so that the output of the model is predictable
ap_lda10 <- LDA(s, k = 10, control = list(seed = 1234))
ap_lda10

#The tidytext package provides this method for extracting the per-topic-per-word probabi
ap_topics <- tidy(ap_lda10, matrix = "beta")
ap_topics
```

```
# We could use dplyr's slice_max() to find the 10 terms that are most common within each topic
# As a tidy data frame, this lends itself well to a ggplot2 visualization (Figure 6.2).

ap_top_terms <- ap_topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta)

ap_top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()
```

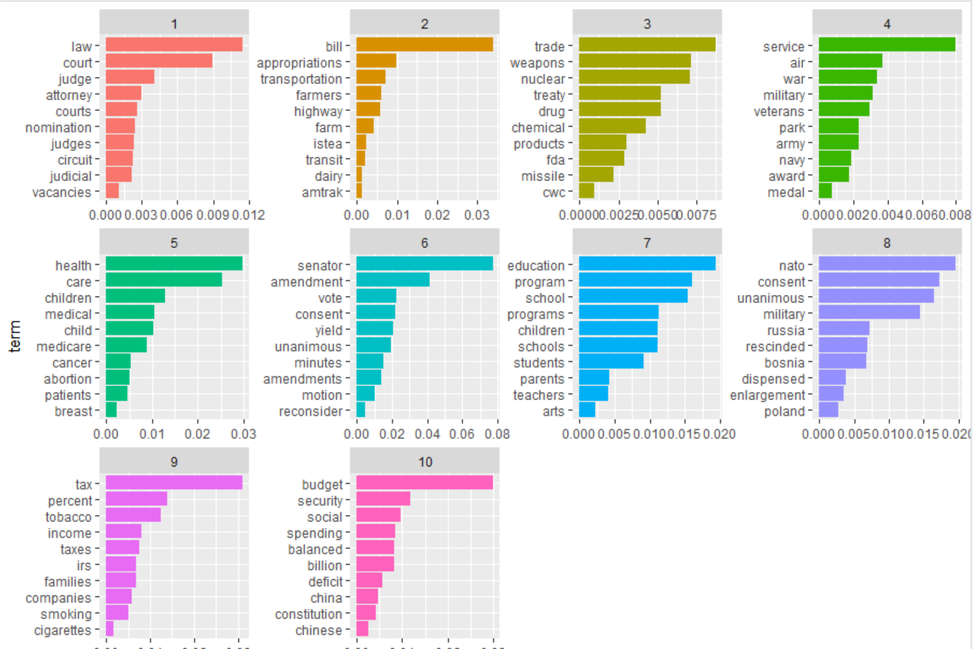
Compute term score for term v in topic k

Term scores downweight term probabilities by how likely they are to be generated by any topic.

$$
\begin{aligned}
term\_score_{k,v} &= \widehat{\beta}_{k,v} \log\left(\frac{\widehat{\beta}_{k,v}}{\left(\Pi_{k=1}^{K}\widehat{\beta}_{j,v}\right)^{\frac{1}{K}}}\right) \\
&= \widehat{\beta}_{k,v}\left(\log\left(\widehat{\beta}_{k,v}\right) - \frac{1}{K}\sum_{j=1}^{K}\ln\left(\widehat{\beta}_{j,v}\right)\right)
\end{aligned}
$$

```r
# As an alternative, use the term-score measure of relative use
sumlogbeta  <- ap_topics %>%
  mutate(log_beta = log(beta)) %>%
  group_by(term)  %>%
  summarize(s_log_beta=sum(log_beta))

ap_top_terms2 <- ap_topics %>%
  inner_join(sumlogbeta)   %>%
  mutate(log_beta = log(beta))  %>%
  mutate(term_score = beta * (log(beta)-(s_log_beta)/10)) %>%
  group_by(topic)  %>%
  slice_max(term_score, n = 10) %>%
  ungroup() %>%
  arrange(topic, -term_score)

ap_top_terms2 %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  scale_y_reordered()
```

Plot term scores



Documents most about a topic

```
# Document-topic probabilities
# We can examine the per-document-per-topic probabilities with the matrix = "gamma" argument to

ap_documents <- tidy(ap_lda10, matrix = "gamma")
ap_documents

ap_top_documents <- ap_documents %>%
  group_by(topic)  %>%
  slice_max(gamma, n = 10) %>%
  ungroup() %>%
  arrange(topic, -gamma)
```

	document	topic	gamma
40	53448	4	0.9953655
41	57603	5	0.9981488

Topic 5

<DOCNO>105-snowe-me-1-19980313</DOCNO>
<TEXT>
Ms. SNOWE.
Mr. President, I rise today to introduce legislation which will authorize breast cancer research funding at a record level.
Over the past seven years, Congress has demonstrated an increased commitment to the fight against breast cancer. Back in 1991, less than $100 million dollars was spent on breast cancer research. Since then, Congress has steadily increased this allocation. These increases have stimulated new and exciting research that has begun to unravel the mysteries of this devastating disease and is moving us closer to a cure. Today, we must send a message through our authorization level to scientists and research policy makers that we are committed to continued funding for this important research.
This increase in funding is necessary because breast cancer has reached crisis levels in America. In 1998, it is estimated that 178,700 new cases of breast cancer will be diagnosed in this country, and 43,500 women will die from this disease. Breast cancer is the most common form of cancer and the second leading cause of cancer deaths among American women. Today, over 2.6 million American women are living with this disease. In my home state of Maine, it is the most commonly-diagnosed cancer among women, representing more than 30 percent of all new cancers in Maine women.

Text as data task

The files in the folder 105-extracted-date contains all speeches by U.S. senators in the 105th Congress (1997-1998). The name of each file shows the congress-name-state abbreviation. For example, the file "105-akaka-hi.txt" contains all speeches by senator Akaka from Hawaii in the 105th congress (1997-1998).

The file sen105_party.csv contains the senator name, state abbreviation and party (100=Democrat, 200=Republican).

1. Load data. Read all speech-files into a corpus using the tm command VCorpus. Turn the data into a tibble (data frame) with columns containing the name of file containing text, the word and row number.

2. Pre-processing. Remove non-alphabetic characters, stopwords and other words that you find to be uninformative. Also generate variables with bigrams and trigrams for each senator.

3. Simple analysis.

a. Compute overall frequency lists for bigrams and trigrams. What are the most frequent bigrams and trigrams?

b. Merge in party information. Compute frequency lists for bigrams and trigrams by party. Plot a wordcloud for the 50 words most frequently used by each party.

4. Analysis. Estimate a Lasso logit model predicting the party of the senator based on bigrams. What bigrams are most important in predicting the party of the senator?

5. **LDA**. Estimate al LDA topic model with 5 topic based on the speeches by the senators. What ten words are most characteristic of each topic?