# 615 for text mining

Qixuan Zhang,Tingrui Huang

11/4/2018

```
# Read text file
library(tidytext)
library(readr)
library(tidytext)
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────── tidyverse 1.2.1
──
```

```
## ✓ ggplot2 3.0.0     ✓ purrr   0.2.5
## ✓ tibble  1.4.2     ✓ dplyr   0.7.6
## ✓ tidyr   0.8.1     ✓ stringr 1.3.1
## ✓ ggplot2 3.0.0     ✓ forcats 0.3.0
```

```
## ── Conflicts ────────────────────────────────────── tidyverse_conflicts()
──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(tidyr)
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(reshape2)
```

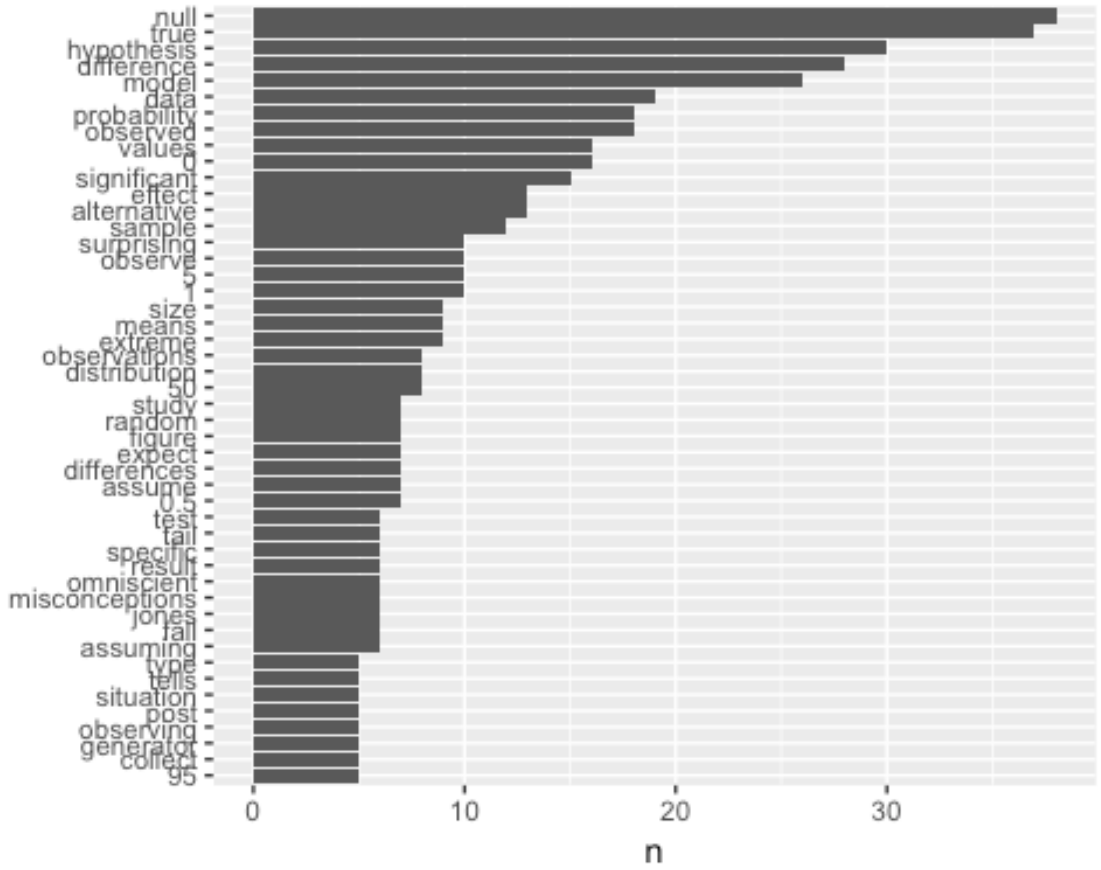
```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths

library(igraph)

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:purrr':
##
##     compose, simplify

## The following object is masked from 'package:tidyr':
##
##     crossing

## The following object is masked from 'package:tibble':
##
##     as_data_frame

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

library(ggraph)
library(rvest)

## Loading required package: xml2

##
## Attaching package: 'rvest'

## The following object is masked from 'package:purrr':
##
##     pluck

## The following object is masked from 'package:readr':
##
##     guess_encoding

url1 <- "https://correlaid.org/blog/posts/understand-p-values"
content <- read_html(url1)
html1<- html_nodes(content,"div.post-content") %>% html_text
text <- str_trim(html1,side = "both")
```

```
text1 <- as.tibble(text)
colnames(text1) <- "text"
text1[,"text"]<-gsub("\n","",text1[,"text"])
```

## Chapter1 The tidy text format

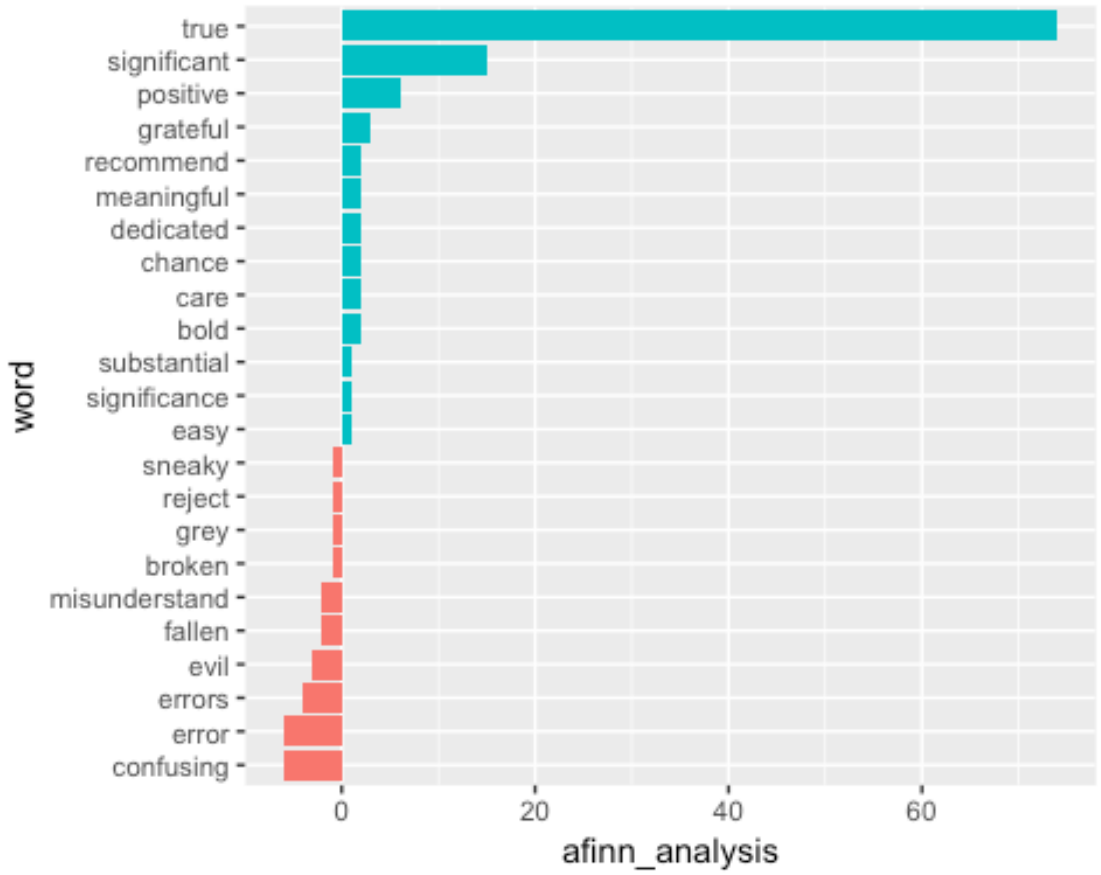
```
#unnest_token function
text1_1<-text1 %>%
  unnest_tokens(word, text)

data(stop_words)
text1_11<- text1_1 %>%
  anti_join(stop_words,by="word")
# Plot the tidy blog
#plot the frenquecy of blog1
text1_11%>% count(word,sort=TRUE) %>%
  filter(n >4) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```



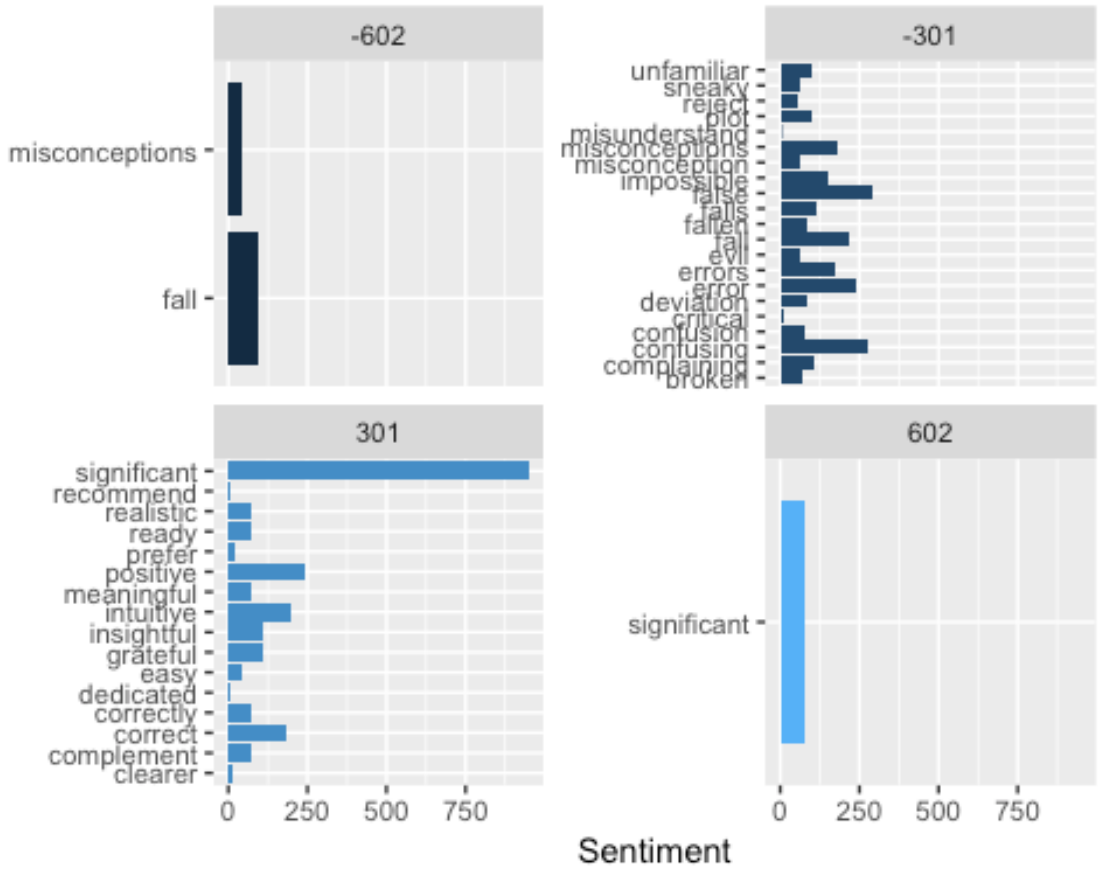

```
# Compare the frenquency of blog1 and blog2
```

## Chapter2 Sentiments words

```
# Sentiments analysis of afinn, bing,nrc
library(tidytext)
sentiments

## # A tibble: 27,314 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 27,304 more rows

afinn <- get_sentiments("afinn")
bing <- get_sentiments("bing")
nrc <- get_sentiments("nrc")

afinn_analysis <- text1_11 %>%
  inner_join(get_sentiments("afinn"), by = "word") %>%
  group_by(word) %>%
  summarize(occurences = n(),
            afinn_analysis = sum(score))
#ggplotafinn_analysis
afinn_analysis %>%
  top_n(25, abs(afinn_analysis)) %>%
  mutate(word = reorder(word, afinn_analysis)) %>%
  ggplot(aes(word, afinn_analysis, fill = afinn_analysis > 0)) +
  geom_col(show.legend = FALSE) +
  coord_flip()
```




```
sentitext <-  text1_11 %>%
  unnest_tokens(sentence, text, token = "sentences") %>%
  group_by(word) %>%
  mutate(linenumber = row_number()) %>%
  ungroup() %>%
  unnest_tokens(word, sentence) %>%
  anti_join(stop_words, by = "word")
#Caculate the sentiment word for p-value article
bing_analysis <- sentitext %>%
  inner_join(bing, by = "word") %>%
  count(word, index = linenumber , sentiment)%>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sentiment = positive - negative)

  ggplot(bing_analysis,aes(x=word,y=index,fill = sentiment)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free_y") +
  labs(y = "Sentiment",
       x = NULL) +
  coord_flip()
```




```
# Compare
```

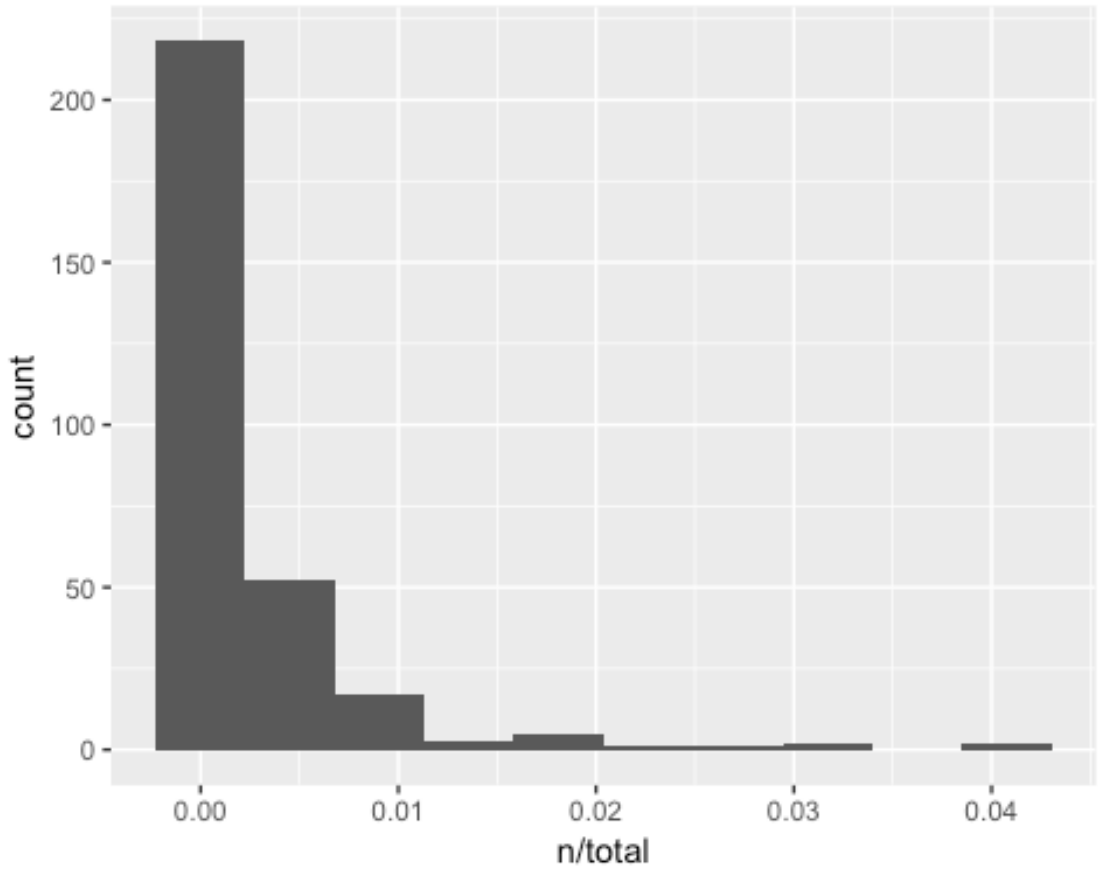
```
# Wordcloud
library(wordcloud)
text1_1 %>%
  anti_join(stop_words,by="word") %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))
```

```
## Warning in wordcloud(word, n, max.words = 100): hypothesis could not be
fit
## on page. It will not be plotted.
```

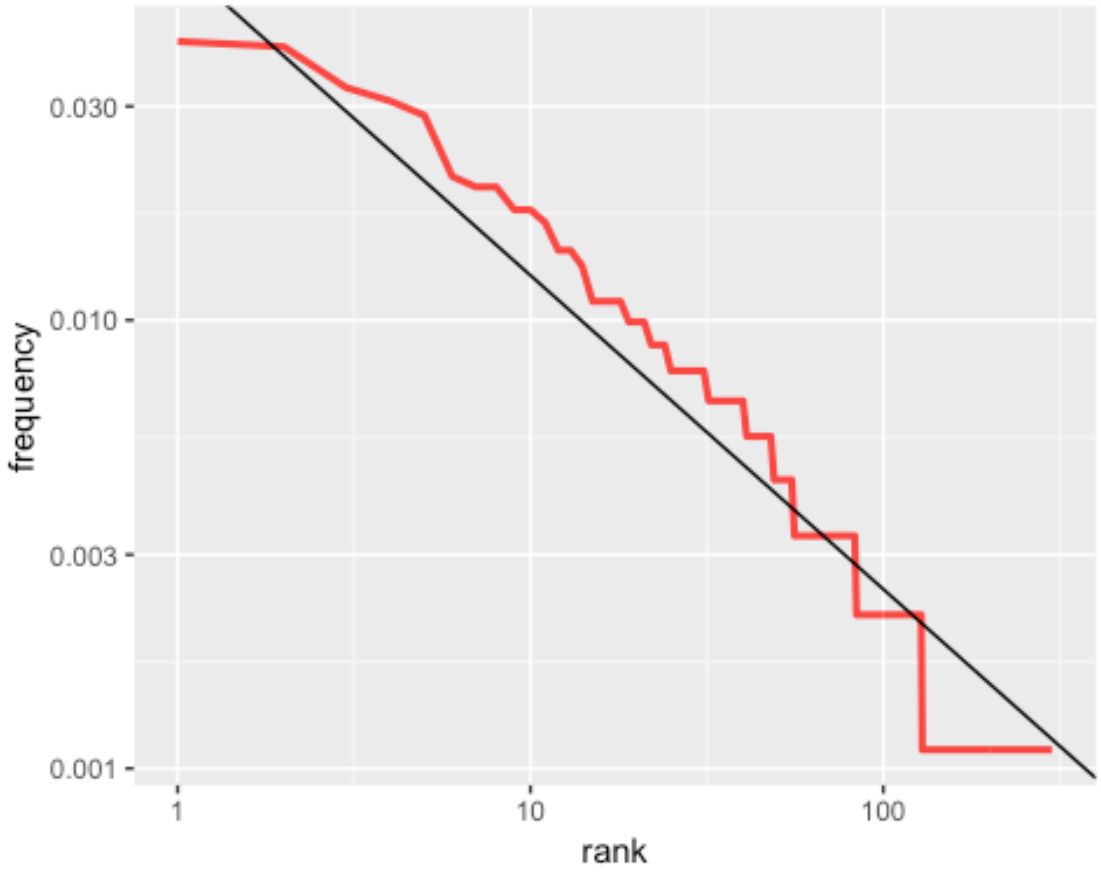
```
## Warning in wordcloud(word, n, max.words = 100): probability could not be
## fit on page. It will not be plotted.
```


##Chapter3

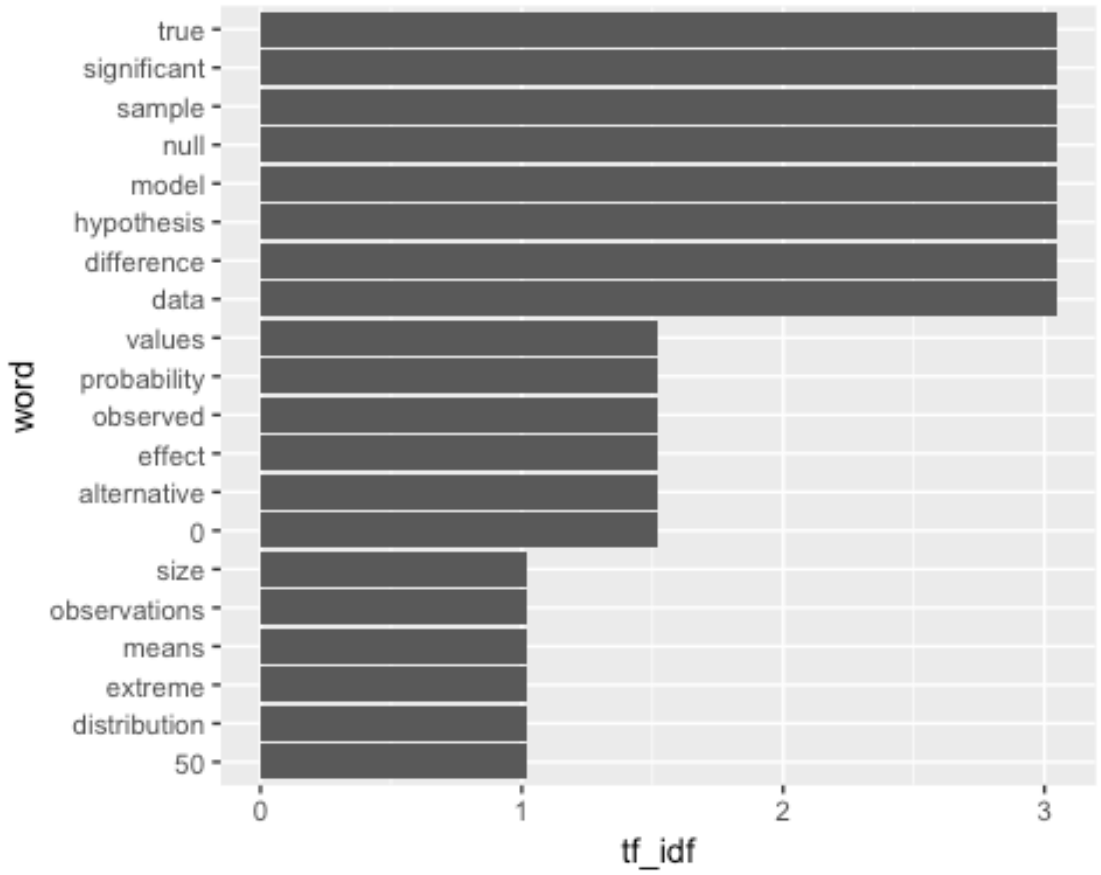
```
#Total word frequency calculate and plot
total_words<-text1_11 %>% summarize(total=n())
book_word <- text1_11 %>%
  count(word, sort = TRUE) %>% mutate(total=rep(908,301))
ggplot(book_word) + aes(n/total) + geom_histogram(bins = 10)
```




```
#Frequecy and rank of p-value
freq_by_rank <- book_word %>% mutate(rank=row_number(), frequency=n/total)
ggplot(freq_by_rank) + aes(rank, frequency) + geom_line(size=1.1,
alpha=0.8,color="red") + scale_x_log10() + scale_y_log10() +
  geom_abline(intercept = -1.2, slope = -0.7)
```




```
# TF-IDF
ds_tf_idf <- book_word %>% bind_tf_idf(word,n,total) %>%
arrange(desc(tf_idf))
# Visualization
ds_tf_idf %>% top_n(15) %>% mutate(word = reorder(word, tf_idf)) %>% ggplot()
+ aes(word, tf_idf) + geom_col() + coord_flip()

## Selecting by tf_idf
```

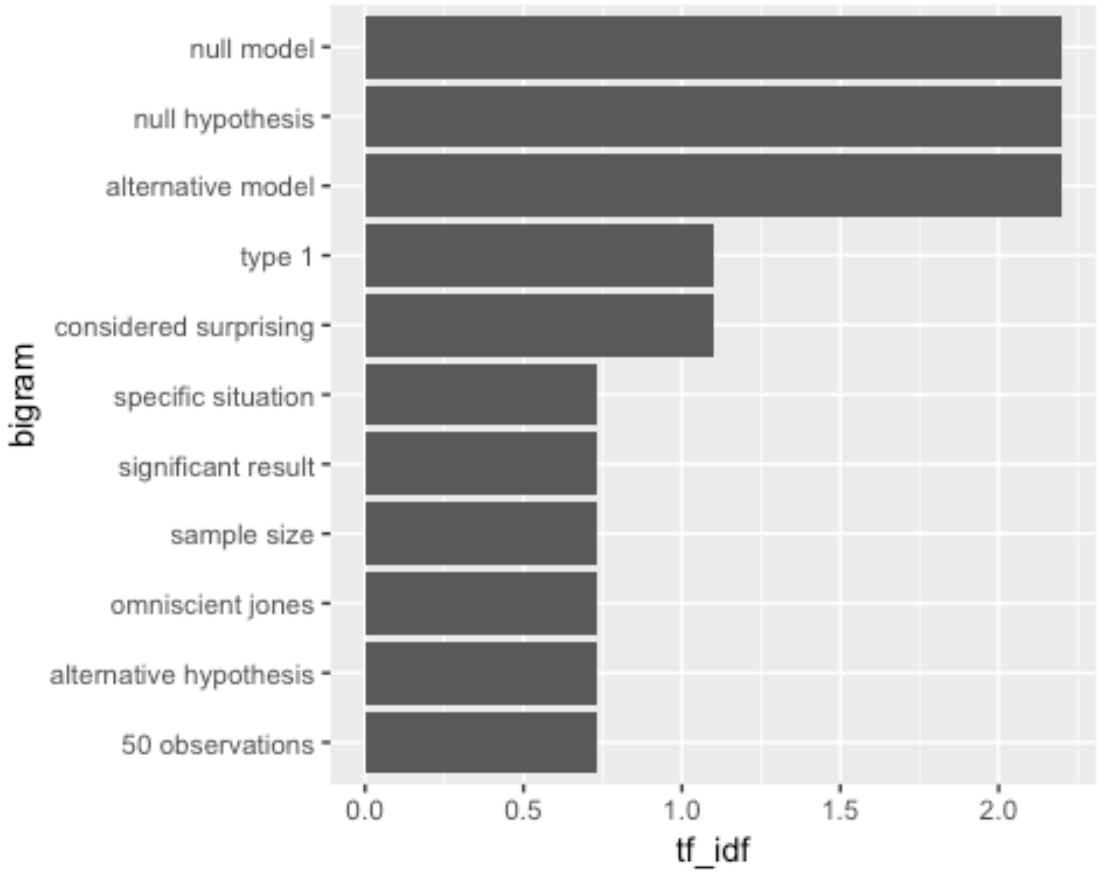



```
#Compute the rank coefficient
rank_subset <- freq_by_rank %>%
  filter(rank < 100,
         rank > 10)
rankreg <- lm(log10(`frequency`) ~ log10(rank), data = rank_subset)
rankreg$coef

## (Intercept) log10(rank)
##  -0.7863396  -0.9258345
```

## Chapter4

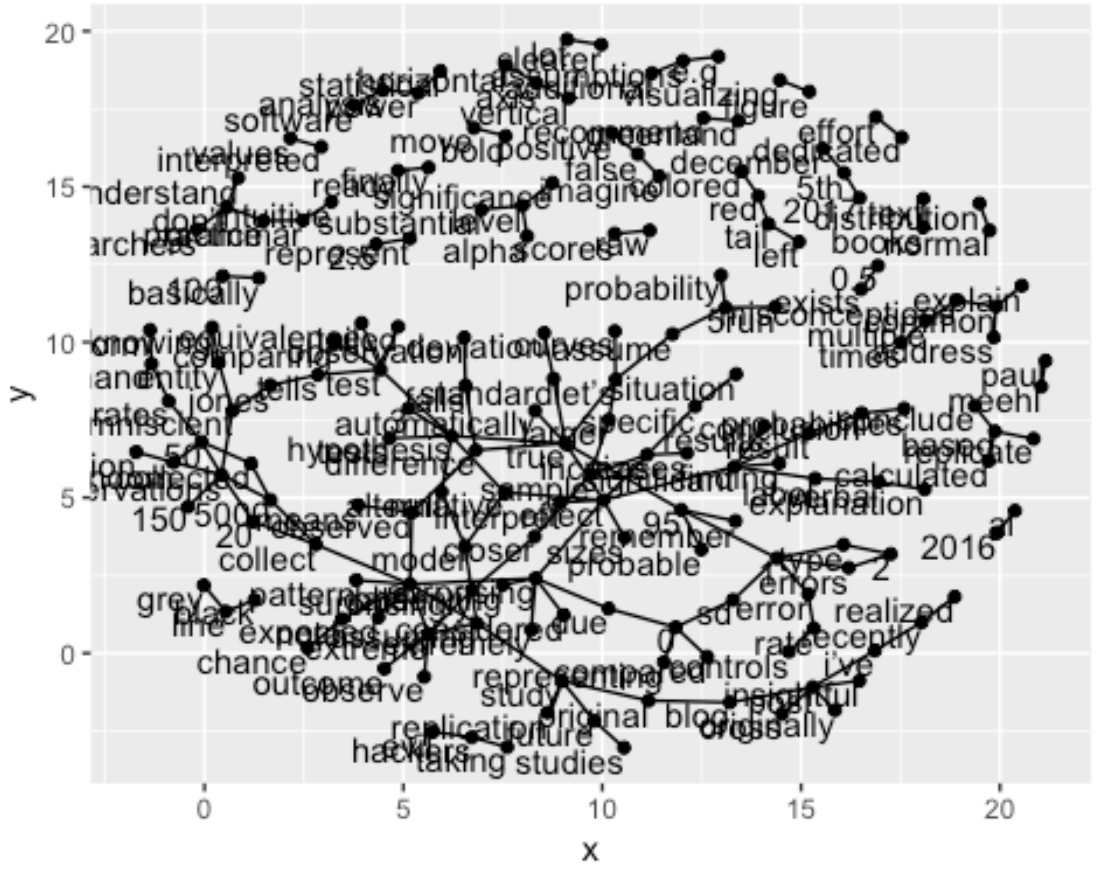
```
# P-value tokenizing by 2 grams
p_value <- text1_11 %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)
#4.1.1 Counting,filtering n-grams and remove stopwords
sum_count<-p_value %>%
  count(bigram, sort = TRUE)
bigrams_separated <- p_value %>%
  separate(bigram, c("word1", "word2"), sep = " ")

p_value_bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
```

```
  filter(!word2 %in% stop_words$word) %>%
   count(word1, word2, sort = TRUE) %>%
  unite(bigram, word1, word2, sep = " ")

# tf-idf totao=908
ds_bigram_tf_idf <- p_value_bigrams_filtered %>% mutate(total = rep(908,
164)) %>% bind_tf_idf(bigram,n,total)
# Visualization
ds_bigram_tf_idf %>% top_n(10) %>% mutate(bigram = reorder(bigram, tf_idf))
%>%
  ggplot() + aes(bigram, tf_idf) + geom_col() + coord_flip()

## Selecting by tf_idf
```




```
# Visualizing a network of bigrams
library(igraph)
bigram_count <- p_value_bigrams_filtered  %>% separate(bigram,
c("word1","word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  count(word1, word2, sort = TRUE)
bigram_graph <- bigram_count %>% graph_from_data_frame()
```

```r
#ggraph
library(ggraph)
set.seed(2017)
ggraph(bigram_graph, layout = "fr") +
geom_edge_link() +
geom_node_point() +
geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```




```r
library(ggraph)
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))
b<-ggraph(bigram_graph,layout = "fr") +
  geom_edge_link(aes(), arrow = a, end_cap=circle(.07, "inches")) +
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label=name),vjust=1,hjust=1) +
  theme_void()

### Step 1. Load text file into R
# "Data Science is not just about Data Science" - ds
library(rvest)
library(stringr)
datascience <- "https://correlaid.org/blog/posts/data-science-books-to-read"
content <- read_html(datascience)
ds <- html_nodes(content,"div.post-content") %>% html_text
```

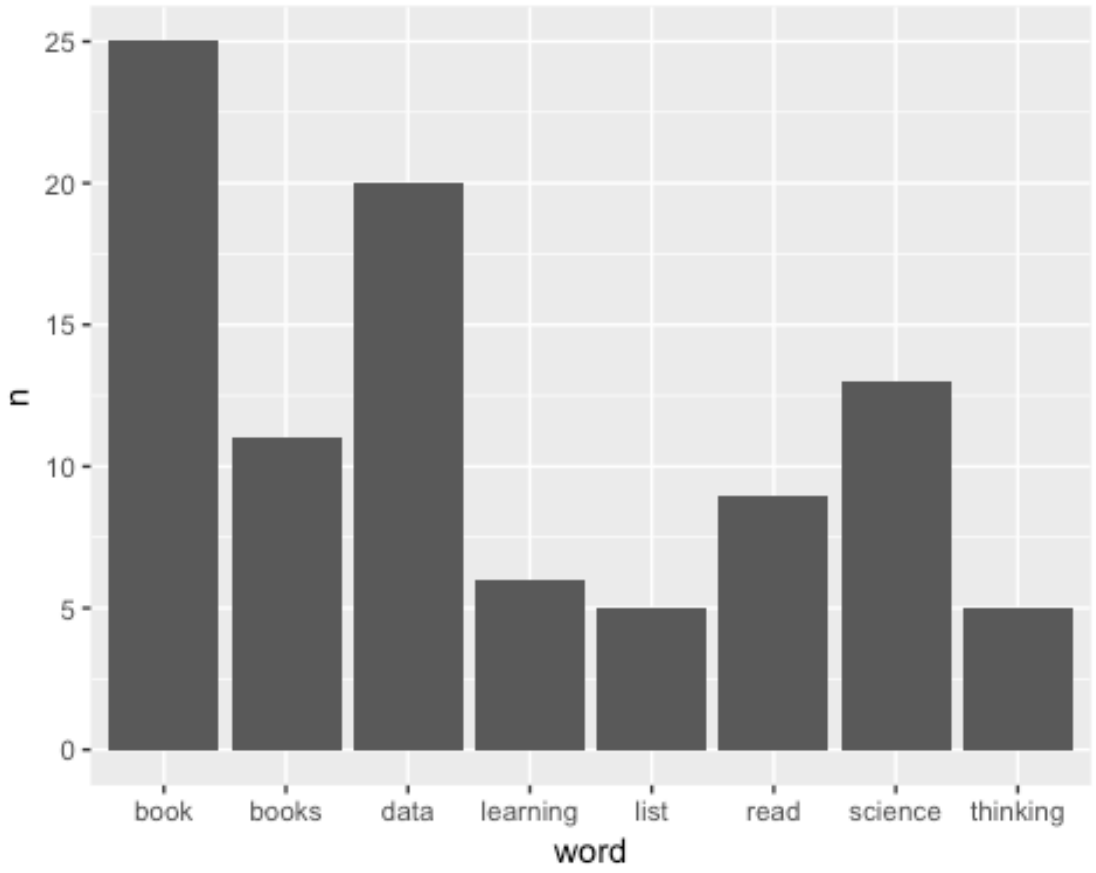
```
ds_2 <- str_trim(ds,side = "both")
ds_3 <- as.tibble(ds_2)
colnames(ds_3) <- "text"
ds_3["text"]<-gsub("\n","",ds_3["text"])
```

### Step 2. Clean data

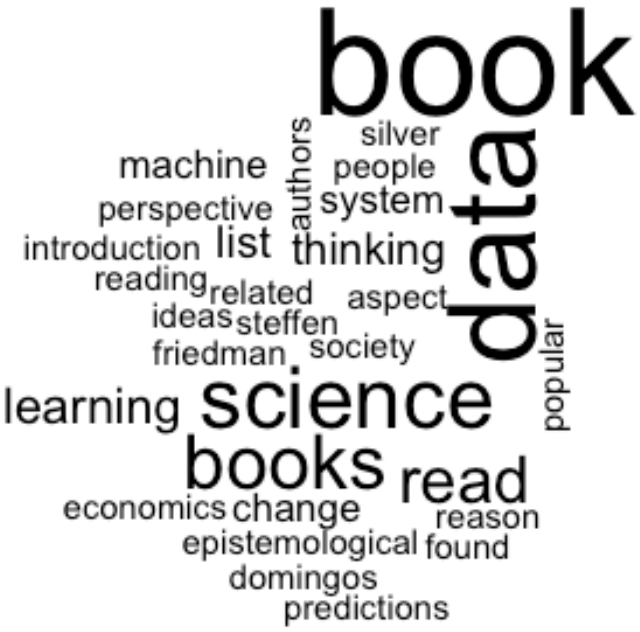
```
library(tidyverse)
library(tidytext)
#One token per row
ds_tidy <- ds_3 %>% unnest_tokens(word, text)
# Remove stop words
data(stop_words)
ds_tidy <- ds_tidy %>% anti_join(stop_words)

## Joining, by = "word"
```

### Step 3. EDA

```
# Most common words in the blog
ds_tidy %>% count(word, sort = TRUE)

## # A tibble: 352 x 2
##    word         n
##    <chr>    <int>
##  1 book        25
##  2 data        20
##  3 science     13
##  4 books       11
##  5 read         9
##  6 learning     6
##  7 list         5
##  8 thinking     5
##  9 change       4
## 10 machine      4
## # ... with 342 more rows

# Visualize the common words in the blog
ds_tidy %>% count(word, sort = TRUE) %>% filter(n > 4) %>%
ggplot(aes(word,n)) + geom_col()
```

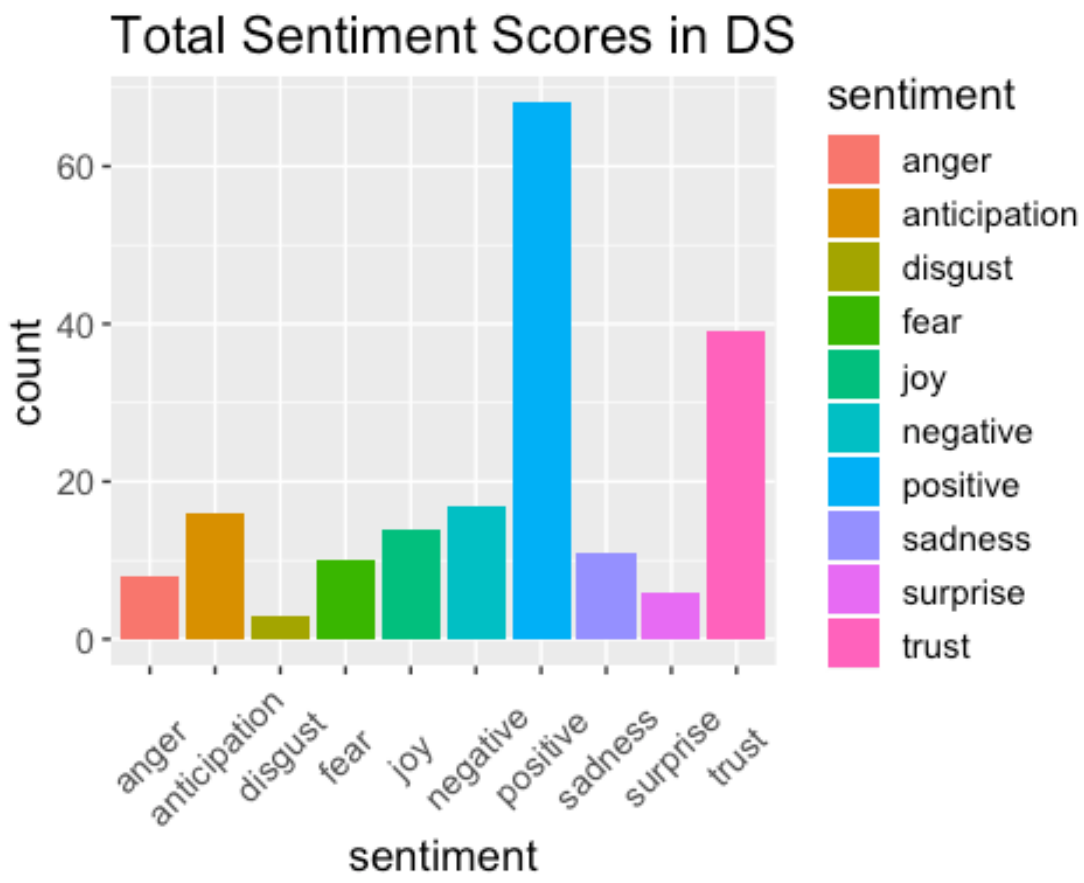



```r
# Wordcloud
library(wordcloud)
library(RColorBrewer)
ds_tidy %>% count(word, sort = TRUE) %>% with(wordcloud(word,n))
```

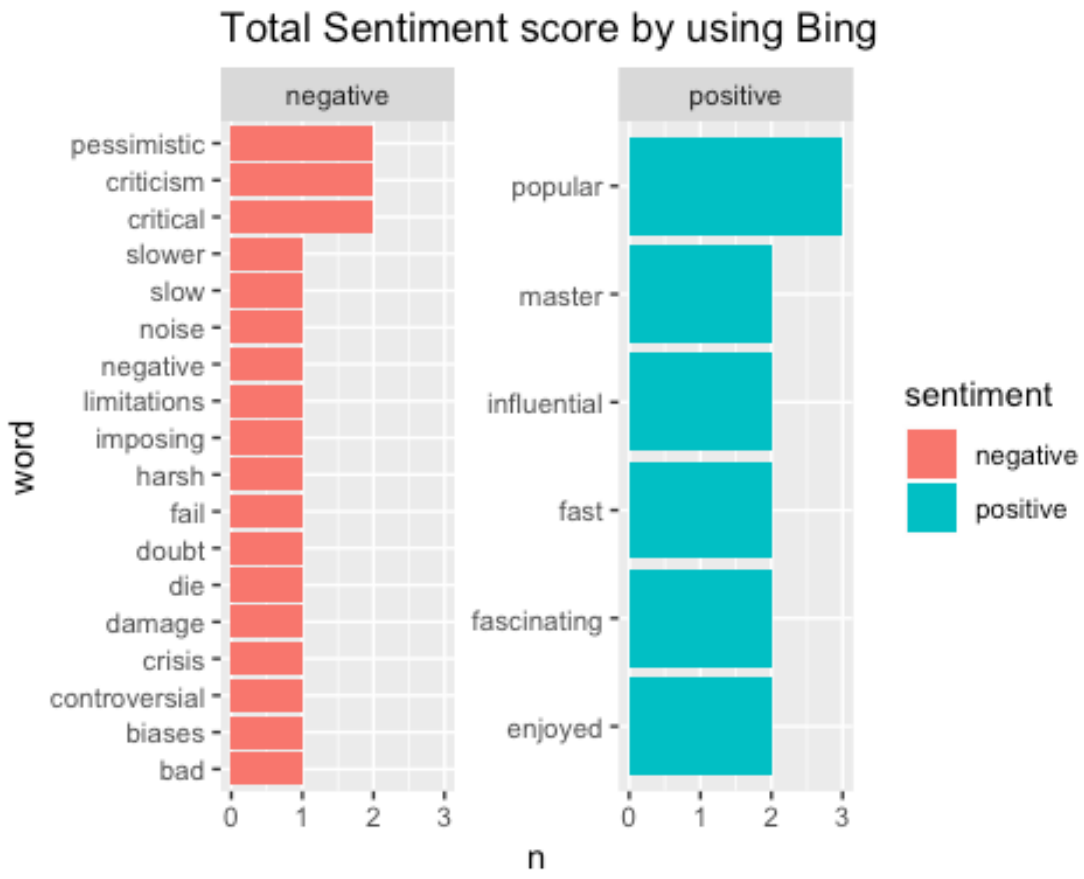



## Step 4. Sentiment Analysis

```
# Using "afinn"
ds_affin <- ds_tidy %>% inner_join(get_sentiments("afinn")) %>%
summarise(sentiment = sum(score))

## Joining, by = "word"

# Using "nrc"
ds_nrc <- ds_tidy %>% inner_join(get_sentiments("nrc")) %>% count(word,
sentiment)

## Joining, by = "word"

ggplot(ds_nrc) + aes(sentiment, n) + geom_bar(aes(fill=sentiment), stat =
"identity") +
  theme(text = element_text(size=14), axis.text.x = element_text(angle = 45,
vjust = 0.5)) +
  ylab("count") + ggtitle("Total Sentiment Scores in DS")
```

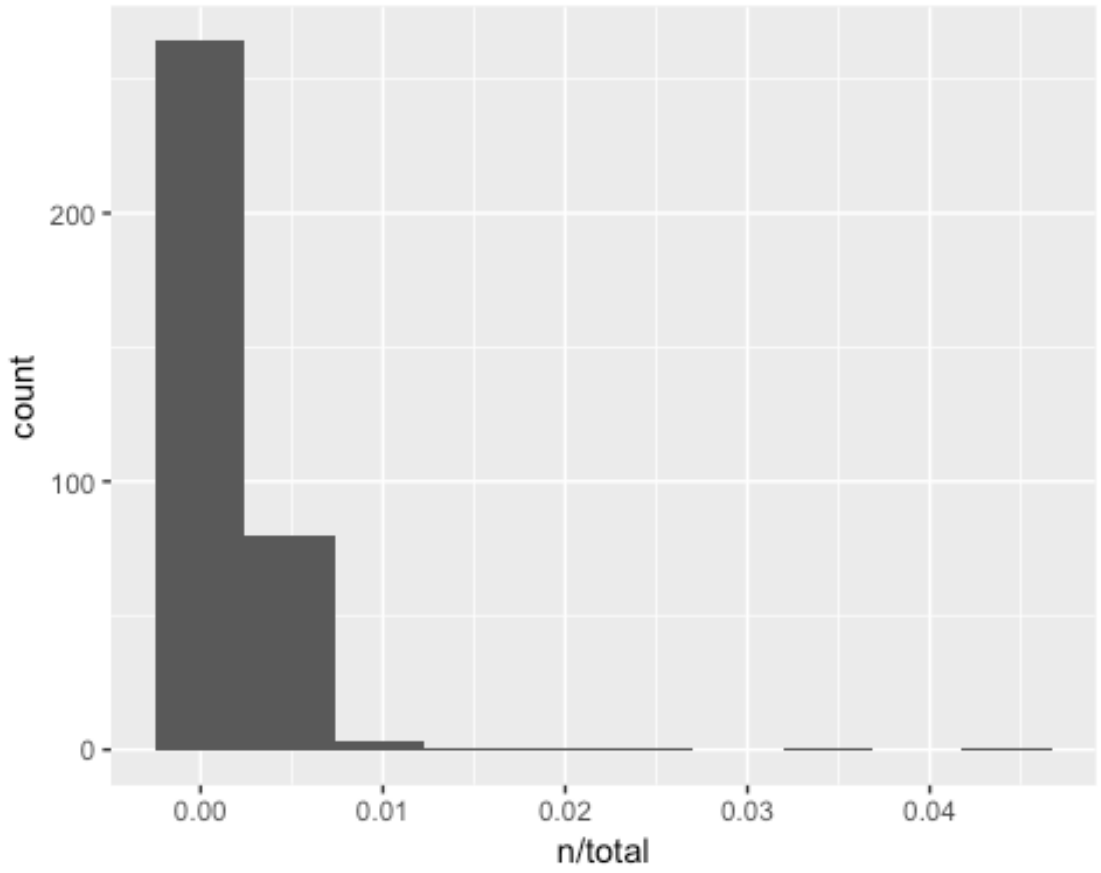



```
# Using "bing"
ds_bing <- ds_tidy %>% inner_join(get_sentiments("bing")) %>% count(word,
sentiment, sort = TRUE)

## Joining, by = "word"

ds_bing_group <- ds_bing %>% group_by(sentiment) %>% top_n(5) %>% ungroup()
%>% mutate(word=reorder(word, n))

## Selecting by n

ggplot(ds_bing_group, aes(word, n, fill=sentiment)) + geom_col() +
facet_wrap(~sentiment, scales = "free_y") +
  ggtitle("Total Sentiment score by using Bing")  +coord_flip()
```

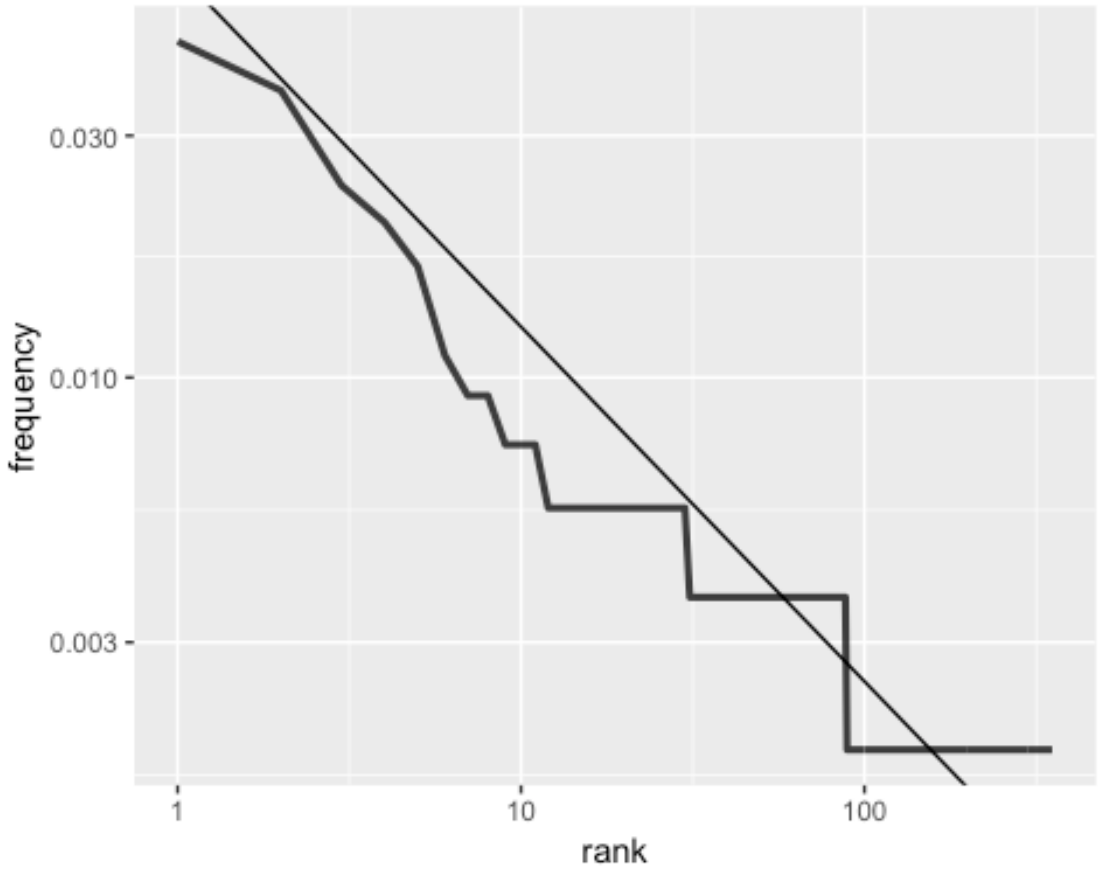



## Step 5. Term Frequency and Inverse Document Frequency

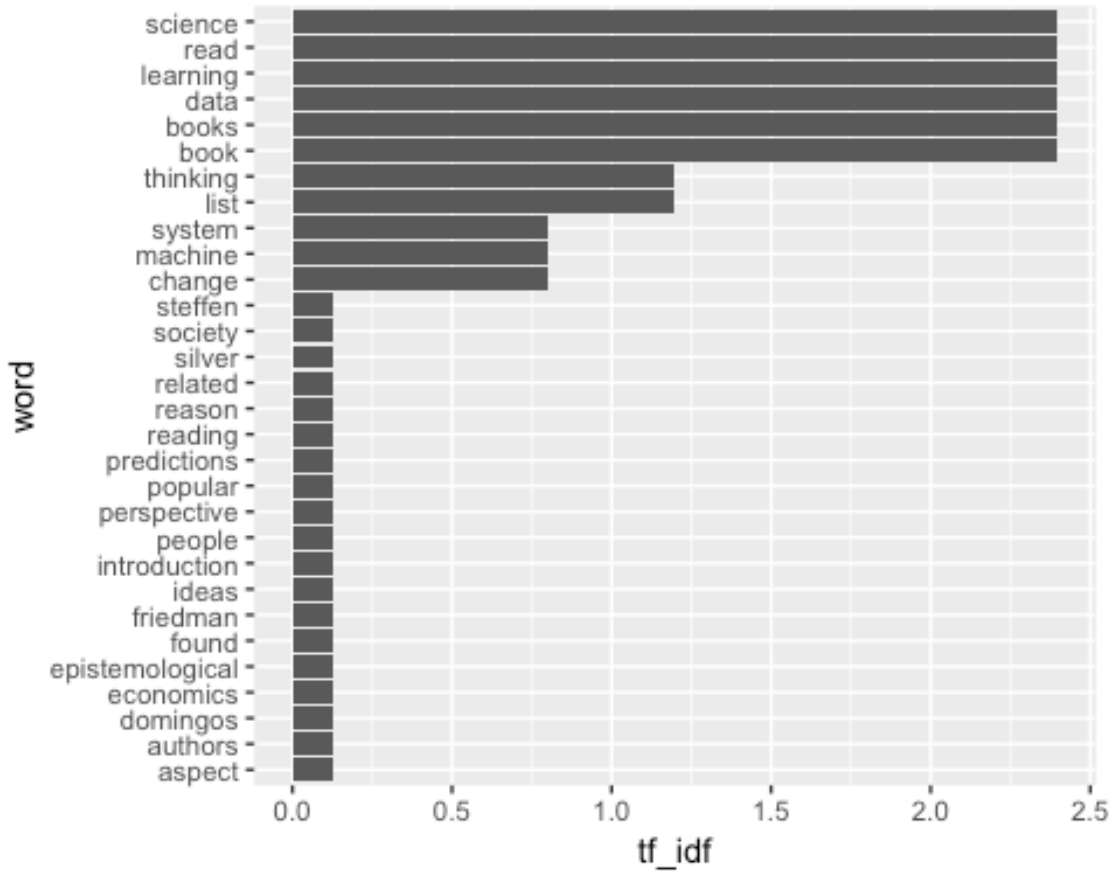
```
# Total words
tt_words <- ds_tidy %>% count(word, sort = TRUE) %>% summarise(total=sum(n))
ds_words <- ds_tidy %>% count(word, sort = TRUE) %>% mutate(total = rep(543,
352))
# Term Frequency
ggplot(ds_words) + aes(n/total) + geom_histogram(bins = 10)
```




```
# Term Frequency and Rank
freq_by_rank <- ds_words %>% mutate(rank=row_number(), frequency=n/total)
ggplot(freq_by_rank) + aes(rank, frequency) + geom_line(size=1.1, alpha=0.8)
+ scale_x_log10() + scale_y_log10() +
  geom_abline(intercept = -1.2, slope = -0.7)
```

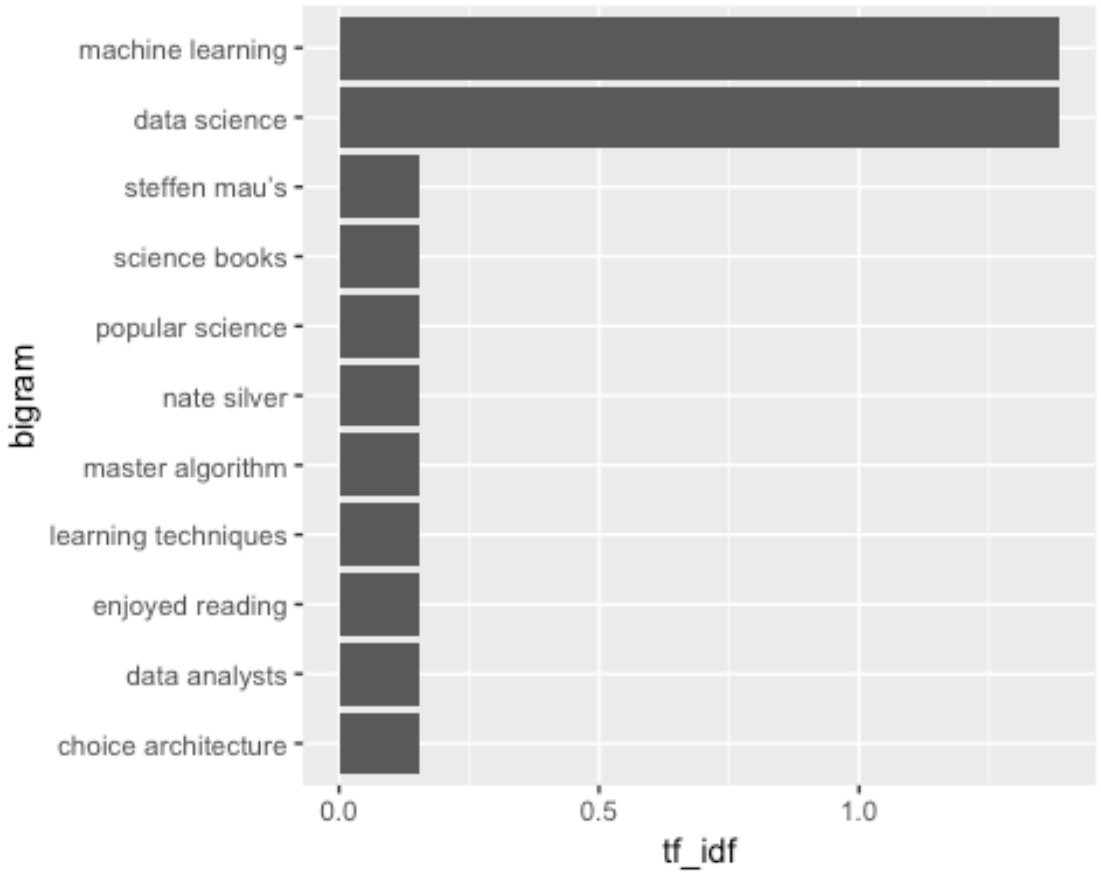



```
# TF-IDF
ds_tf_idf <- ds_words %>% bind_tf_idf(word,n,total) %>% arrange(desc(tf_idf))
# Visualization
ds_tf_idf %>% top_n(15) %>% mutate(word = reorder(word, tf_idf)) %>% ggplot()
+ aes(word, tf_idf) + geom_col() + coord_flip()

## Selecting by tf_idf
```




## Step 6. n-grams and correlations

```
# Tokenizing by 2 grams
ds_bigrams <- ds_3 %>% unnest_tokens(bigram, text, token = "ngrams", n=2)
# Summarize
sum_count <- ds_bigrams %>% count(bigram, sort = TRUE)
# Remove stop words
ds_bigrams_tidy <- ds_bigrams %>% separate(bigram, c("word1","word2"), sep =
" ") %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  count(word1, word2, sort = TRUE) %>%
  unite(bigram, word1, word2, sep = " ")
# tf-idf totao=181
ds_bigram_tf_idf <- ds_bigrams_tidy %>% mutate(total = rep(181, 162)) %>%
bind_tf_idf(bigram,n,total)
# Visualization
ds_bigram_tf_idf %>% top_n(10) %>% mutate(bigram = reorder(bigram, tf_idf))
%>%
  ggplot() + aes(bigram, tf_idf) + geom_col() + coord_flip()

## Selecting by tf_idf
```




```
# Visualizing a network of bigrams
library(igraph)
bigram_count <- ds_bigrams_tidy <- ds_bigrams %>% separate(bigram,
c("word1","word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  count(word1, word2, sort = TRUE)
bigram_graph <- bigram_count %>% graph_from_data_frame()
library(ggraph)
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))
ggraph(bigram_graph,layout = "fr") +
  geom_edge_link(aes(edge_alpha=n), arrow = a, end_cap=circle(.07, "inches"))
+
  geom_node_point(color = "lightblue", size = 5) +
  geom_node_text(aes(label=name),vjust=1,hjust=1) +
  theme_void()
```


```
# correlation and pairs
library(widyr)
word_pairs <- ds_3 %>% mutate(section=row_number() %/% 10) %>%
unnest_tokens(word, text) %>%
  filter(!word %in% stop_words$word) %>% pairwise_count(word, section,
sort=TRUE)
word_cors <- ds_3 %>% mutate(section=row_number()) %>% unnest_tokens(word,
text) %>%
  filter(!word %in% stop_words$word) %>% group_by(word) %>%
  pairwise_cor(word,section, sort = TRUE)
```

```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```