Text Analysis of Correlaid

Longhao Chen/Qianhui Rong/Wenjia Xie/Amdrew Zhang

11/3/2018

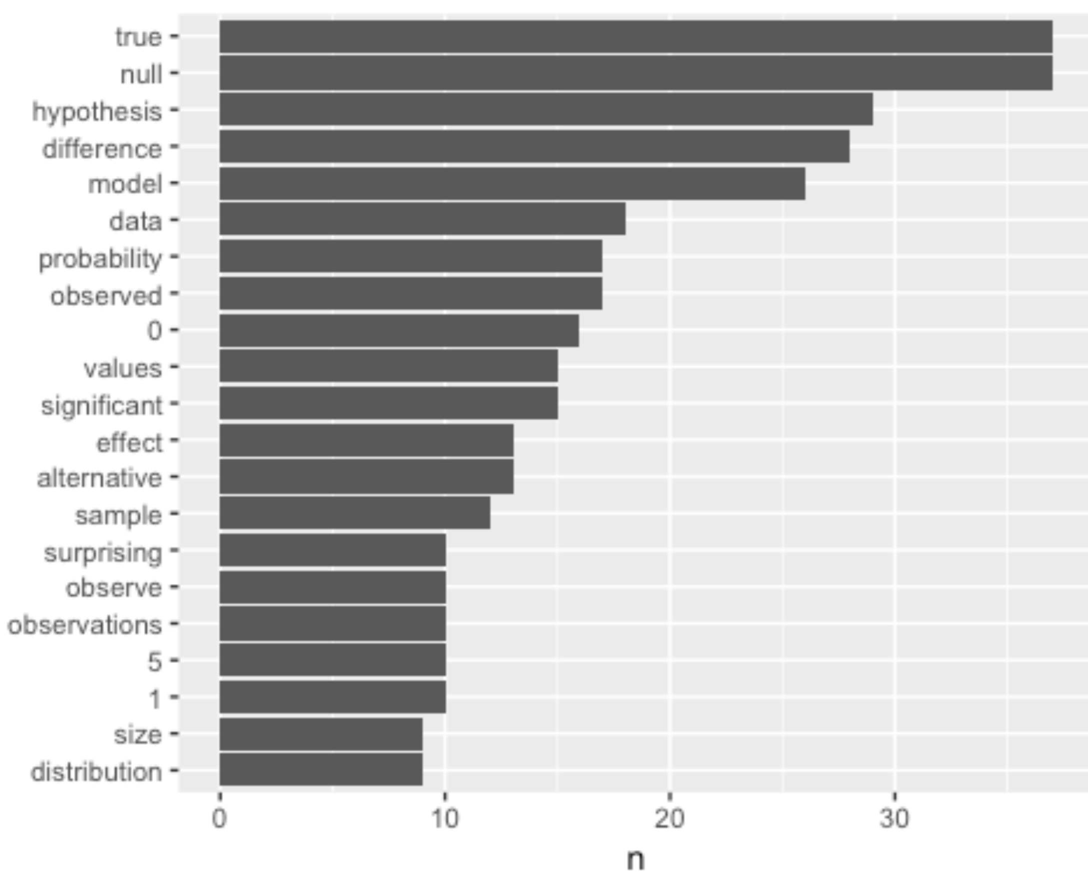
Seperate Analysis on Each Article ## P-Value Article We want to analyze the passage from https://correlaid.org/blog/posts/understand-p-values.

```
library(tidytext)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(stringr)
library(ggplot2)
library(tidyr)

correlaid_txt <- read.delim("correlaid_pvalue.txt")
correlaid_txt <- data.frame(lapply(correlaid_txt, as.character),
stringsAsFactors=FALSE)
colnames(correlaid_txt) <- c("text")
correlaid_txt %>% unnest_tokens(output = word,input = text) ->
token_correlaid

data("stop_words")
token_correlaid %>%
  anti_join(stop_words) -> tidy_correlaid

## Joining, by = "word"

tidy_correlaid %>%
  count(word,sort=TRUE)

## # A tibble: 282 x 2
##    word           n
##    <chr>      <int>
##  1 null          37
##  2 true          37
##  3 hypothesis    29
```

```
## 4 difference     28
## 5 model          26
## 6 data           18
## 7 observed       17
## 8 probability    17
## 9 0              16
## 10 significant   15
## # ... with 272 more rows
```

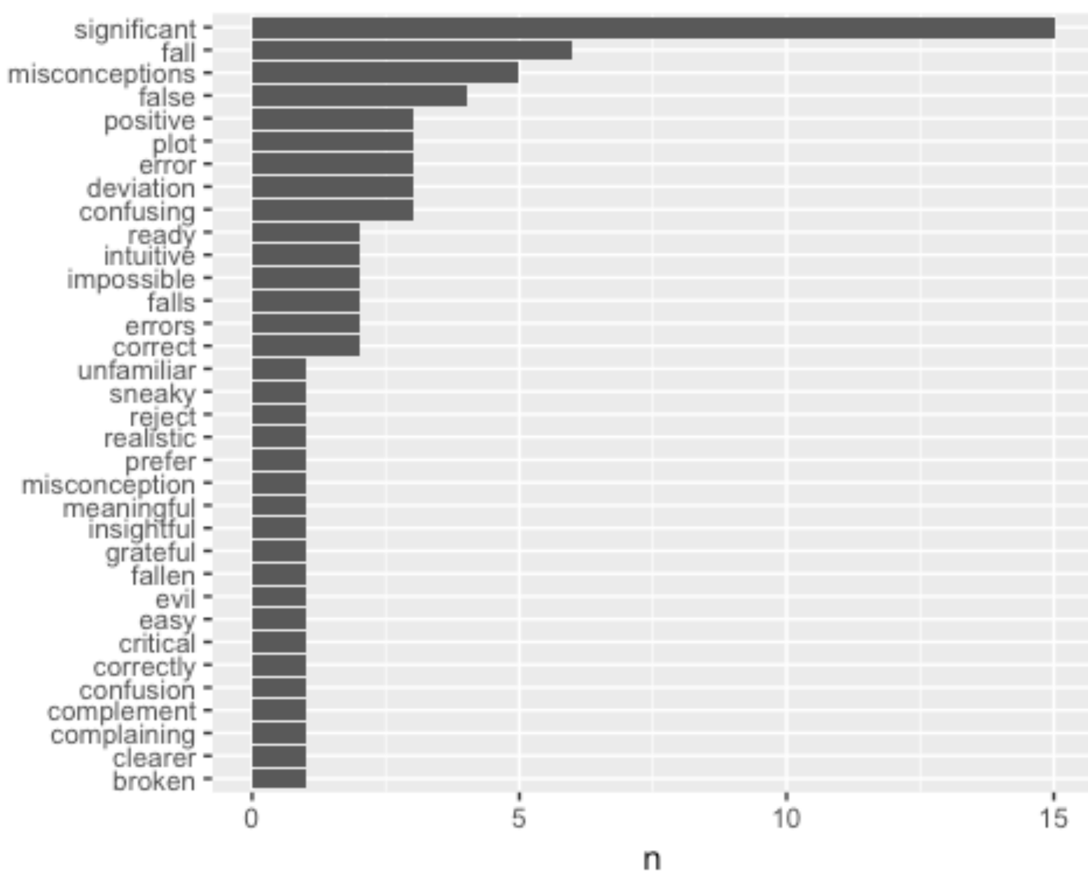
```r
tidy_correlaid %>%
  count(word,sort=TRUE) %>%
  top_n(20) %>%
  mutate(word=reorder(word,n)) %>%  #reorder
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```

```
## Selecting by n
```



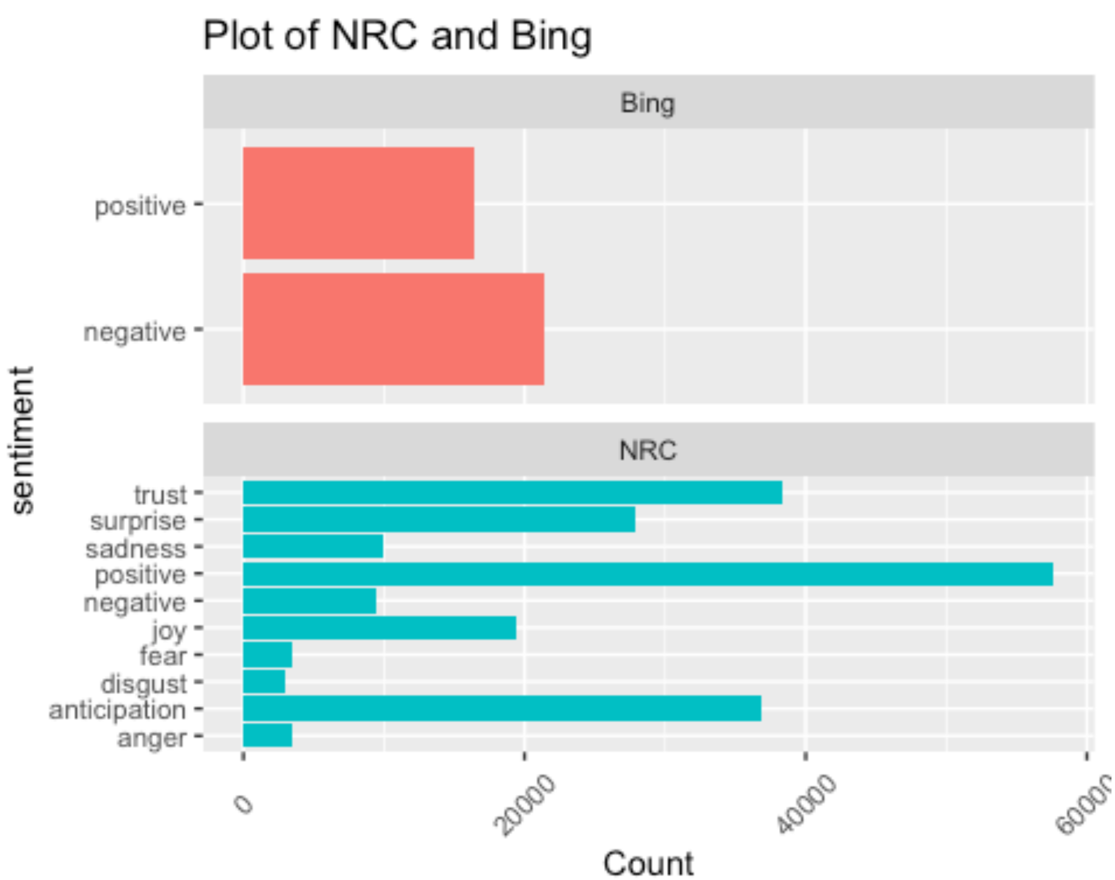
After eliminating the stop words in the article, we order the words appeared in the passage by frequency and we made a ggplot to show the 20 most frequent words appear in the article.

```
sentiment_correlaid <- tidy_correlaid %>%  #With Bing
  inner_join(get_sentiments("bing"))%>%
  mutate(method = "Bing")

## Joining, by = "word"

sentiment_correlaid %>%
  count(word,sort=TRUE) %>%
  top_n(20) %>%
  mutate(word=reorder(word,n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()

## Selecting by n
```



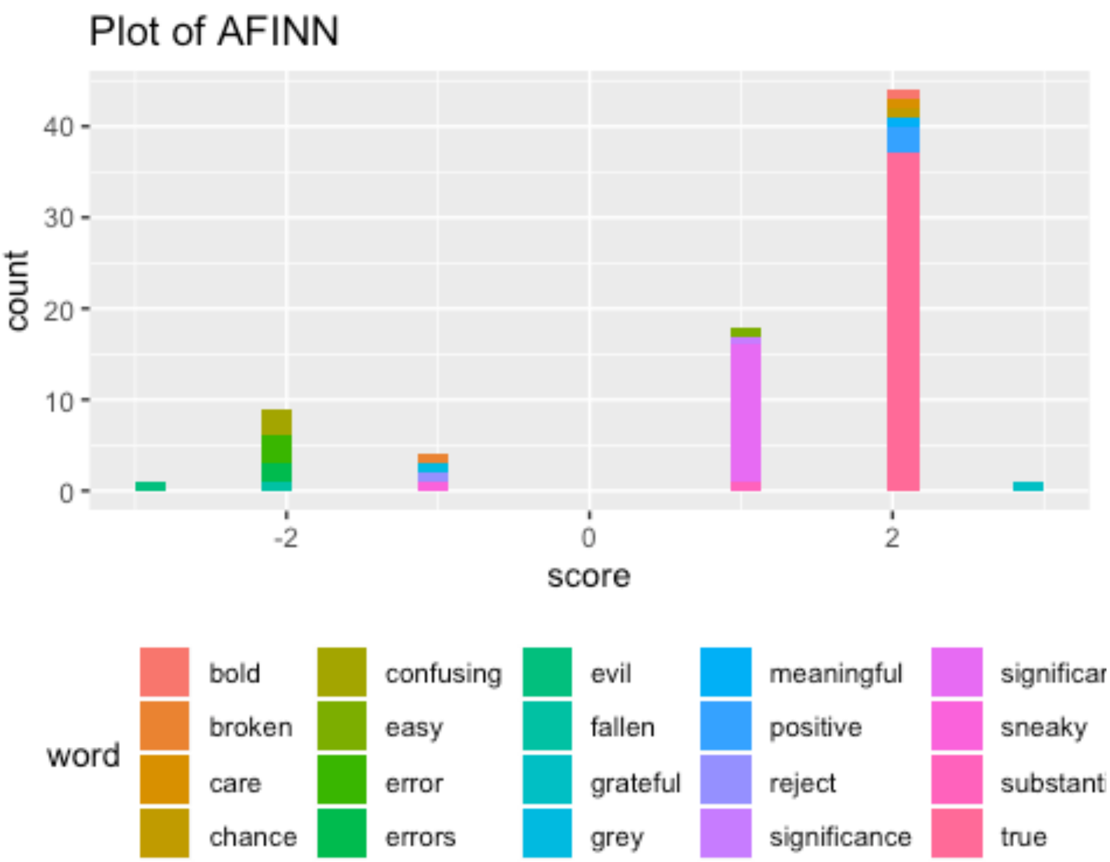
In order to get an odea of the passage's sentiment on P-Value, we applied Bing sentiment package, and made a ggplot of the top 20 sentimental words in the article. The first one "significant" is about 3 times more frequent than the second word in order. That should be due to the term "statistically significant". Then we want to compare the results from the other two packages of sentimenal words: AFINN and NRC.

```r
sentiment_correlaid_AF <- tidy_correlaid %>%
  inner_join(get_sentiments("afinn"))%>%
  mutate(method = "AFINN")

## Joining, by = "word"

sentiment_correlaid_NRC <- tidy_correlaid %>%
  inner_join(get_sentiments("nrc"))%>%
  mutate(method = "NRC")

## Joining, by = "word"

bind_rows(sentiment_correlaid,
          sentiment_correlaid_NRC) %>%
  mutate(Count=n()) -> three_pack
ggplot(aes(sentiment, Count,fill = method),data=three_pack) +
  geom_col(show.legend = FALSE)+
  facet_wrap(~method, ncol = 1, scales = "free_y")+
  theme(axis.text.x = element_text(angle = 45, hjust = 0.5, vjust = 0.5))+
  coord_flip()+
  ggtitle("Plot of NRC and Bing")
```




```r
#Because AFINN's results are in numerical continuous scale, so we draw a
seperate plot for it.
```

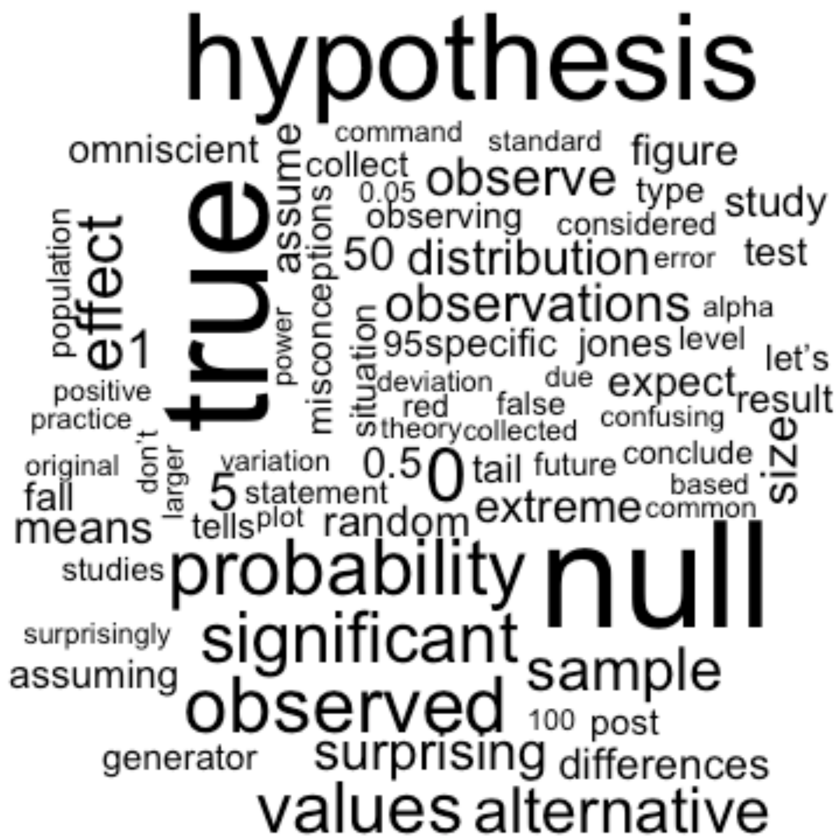
```
sentiment_correlaid_AF %>%
  ggplot()+
  geom_histogram(aes(x=score,fill=word),stat="bin",show.legend = TRUE)+
  theme(legend.position="bottom")+
  scale_alpha_discrete(breaks=c(-3,-2,-1,0,1,2,3))+
  ggtitle("Plot of AFINN")

## Warning: Using alpha for a discrete variable is not advised.

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We want to also see the wordcloud.

```
library(wordcloud)

## Loading required package: RColorBrewer

tidy_correlaid %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))

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

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

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




```r
library(reshape2)

##
## Attaching package: 'reshape2'

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

tidy_correlaid %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("black", "red"),
                   max.words = 100)

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

From the Data to the Story Article

We want to analyze the passage from https://correlaid.org/blog/posts/journocode-workflow.

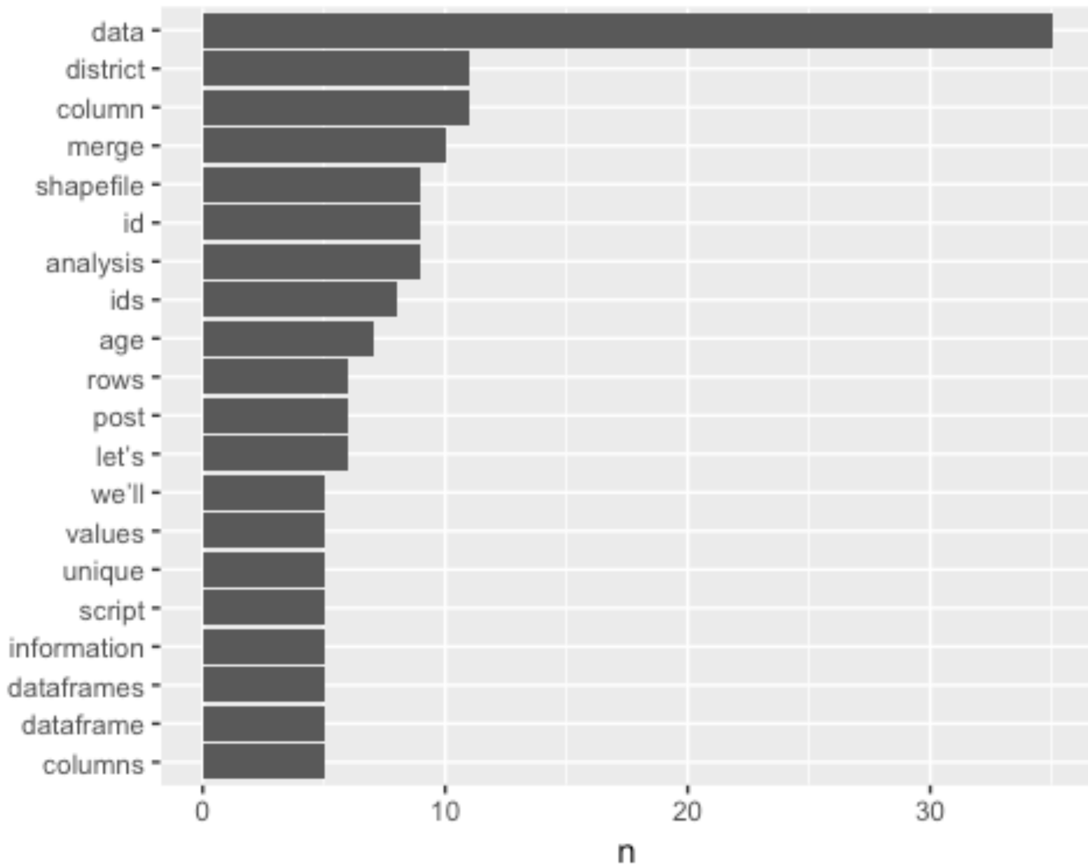
```
library(tidytext)
library(dplyr)
library(stringr)
library(ggplot2)

correlaid_txt2 <- read.delim("correlaid_fromdatatostory.txt")
correlaid_txt2 <- data.frame(lapply(correlaid_txt2, as.character),
stringsAsFactors=FALSE)
colnames(correlaid_txt2) <- c("text")
correlaid_txt2 %>% unnest_tokens(output = word,input = text) ->
token_correlaid2

data("stop_words")
token_correlaid2 %>%
  anti_join(stop_words) -> tidy_correlaid2

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

```r
tidy_correlaid2 %>%
  count(word,sort=TRUE)
```

```
## # A tibble: 285 x 2
##    word          n
##    <chr>     <int>
##  1 data         35
##  2 column       11
##  3 district     11
##  4 merge        10
##  5 analysis      9
##  6 id            9
##  7 shapefile     9
##  8 ids           8
##  9 age           7
## 10 let’s         6
## # ... with 275 more rows
```

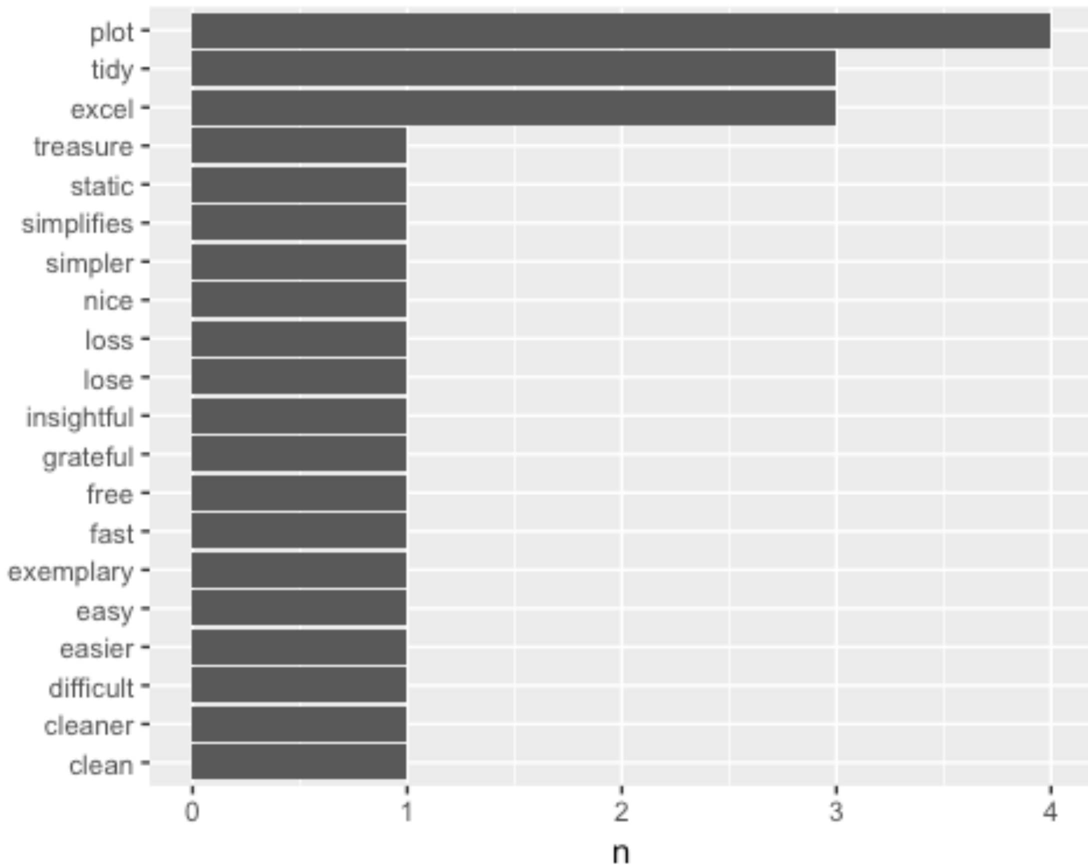
```r
tidy_correlaid2 %>%
  count(word,sort=TRUE) %>%
  top_n(20) %>%
  mutate(word=reorder(word,n)) %>%  #reorder
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```

```
## Selecting by n
```



After eliminating the stop words in the article, we order the words appeared in the passage by frequency and we made a ggplot to show the 20 most frequent words appear in the article.

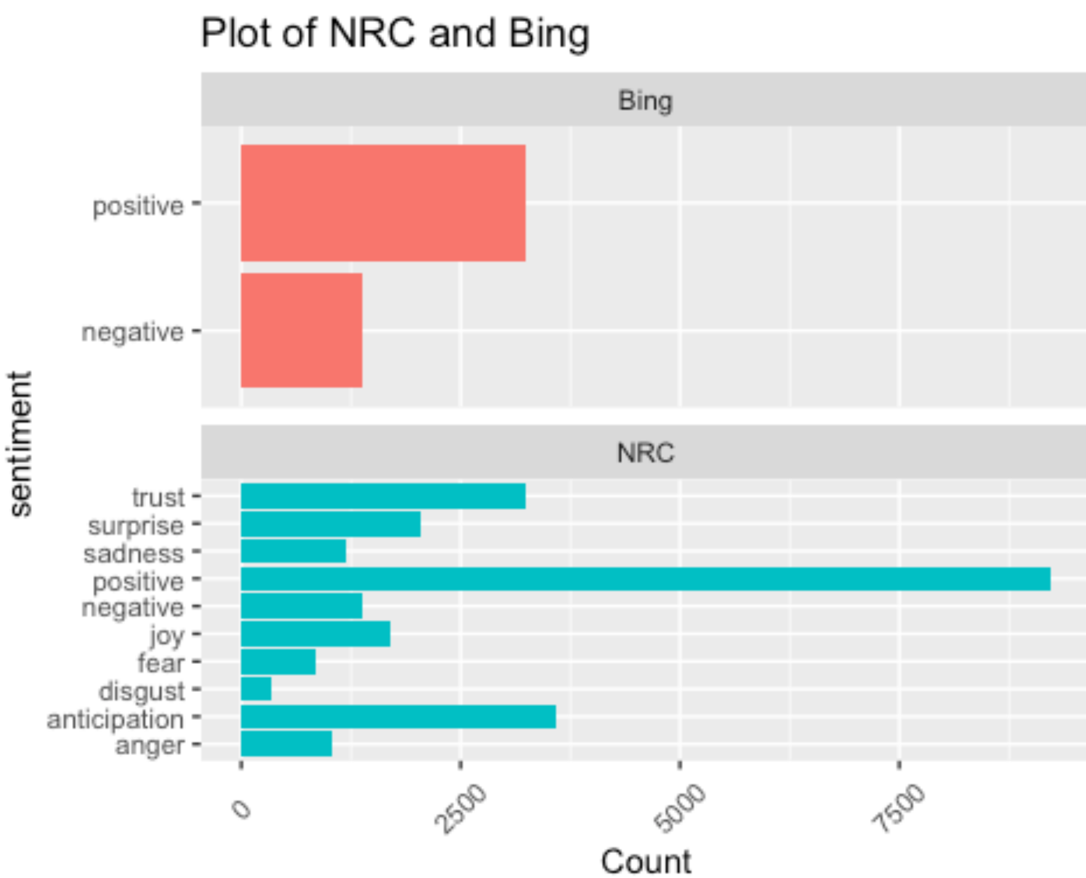
```
sentiment_correlaid2 <- tidy_correlaid2 %>%  #With Bing
  inner_join(get_sentiments("bing"))%>%
  mutate(method = "Bing")

## Joining, by = "word"

sentiment_correlaid2 %>%
  count(word,sort=TRUE) %>%
  top_n(20) %>%
  mutate(word=reorder(word,n)) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()

## Selecting by n
```



In order to get an odea of the passage's sentiment on this article, we applied Bing sentiment package, and made a ggplot of the top 20 sentimental words in the article. The first three are "plot", "excel" and "tidy", which is reasonable because this is a tutorial of R. Then we want to compare the results from the other two packages of sentimenal words: AFINN and NRC.

```
sentiment_correlaid_AF2 <- tidy_correlaid2 %>%
  inner_join(get_sentiments("afinn"))%>%
  mutate(method = "AFINN")

## Joining, by = "word"

sentiment_correlaid_NRC2 <- tidy_correlaid2 %>%
  inner_join(get_sentiments("nrc"))%>%
  mutate(method = "NRC")

## Joining, by = "word"

bind_rows(sentiment_correlaid2,
          sentiment_correlaid_NRC2) %>%
  mutate(Count=n()) -> three_pack2
ggplot(aes(sentiment, Count,fill = method),data=three_pack2) +
  geom_col(show.legend = FALSE)+
  facet_wrap(~method, ncol = 1, scales = "free_y")+
```

```r
  theme(axis.text.x = element_text(angle = 45, hjust = 0.5, vjust = 0.5))+
  coord_flip()+
  ggtitle("Plot of NRC and Bing")
```

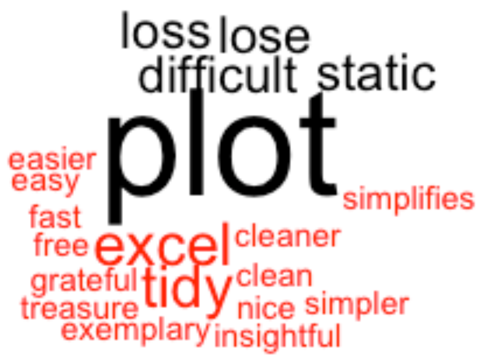



```r
#Because AFINN's results are in numerical continuous scale, so we draw a
seperate plot for it.

sentiment_correlaid_AF2 %>%
  ggplot()+
  geom_histogram(aes(x=score,fill=word),stat="bin",show.legend = TRUE)+
  theme(legend.position="bottom")+
  scale_alpha_discrete(breaks=c(-3,-2,-1,0,1,2,3))+
  ggtitle("Plot of AFINN")

## Warning: Using alpha for a discrete variable is not advised.

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

We want to also see the wordcloud.

```
library(wordcloud)
tidy_correlaid2 %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))
```

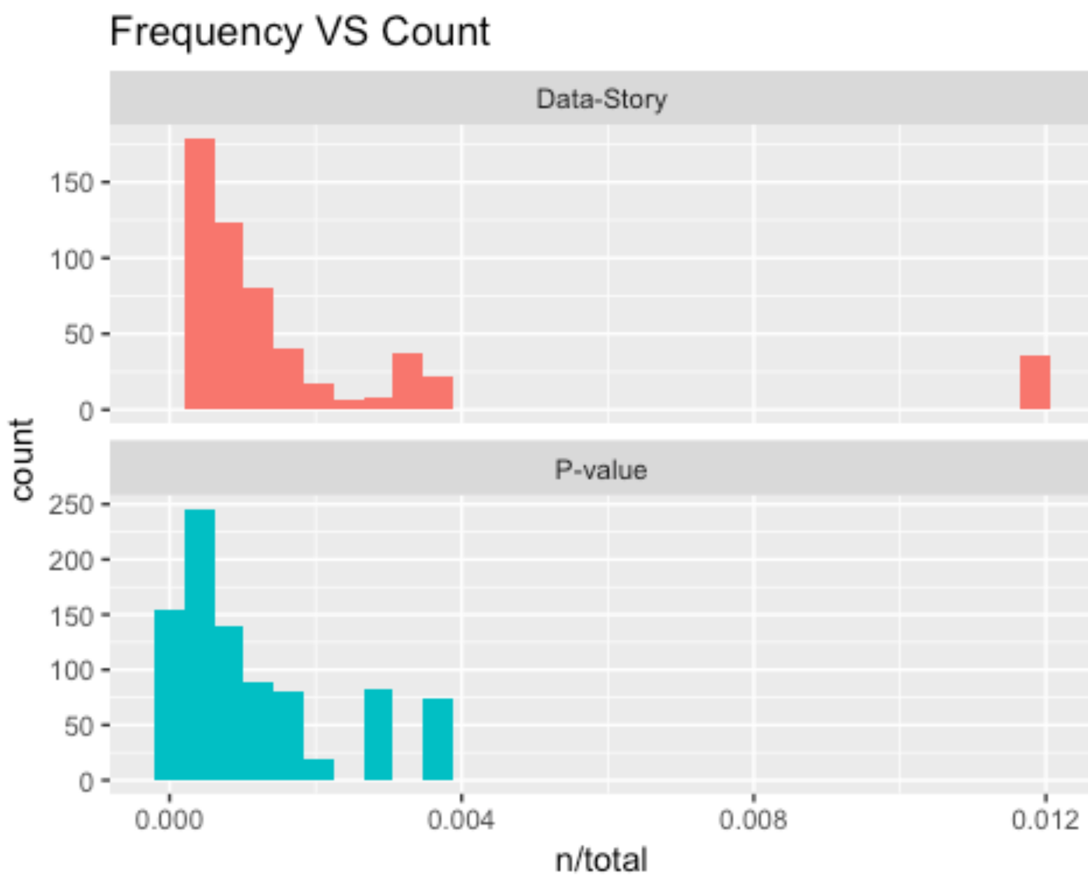

```r
library(reshape2)
tidy_correlaid2 %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, sentiment, sort = TRUE) %>%
  acast(word ~ sentiment, value.var = "n", fill = 0) %>%
  comparison.cloud(colors = c("black", "red"),
                   max.words = 100)

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



Combined Analysis on Two Articles To find important words for the context by decreasing the weight for commonly used words, we apply bind_tf_idf function for these two article.

```
tidy_correlaid %>% group_by(word) %>%
  mutate(n=n()) %>%
  mutate(article="P-value") %>%
  arrange(n)-> correlaid_words
tidy_correlaid2 %>% group_by(word) %>%
  mutate(n=n()) %>%
  mutate(article="Data-Story") %>%
  arrange(n) -> correlaid_words2
total_words <- rbind(correlaid_words,correlaid_words2)
total_words %>% group_by(article) %>%
  mutate(total=sum(n)) %>%
  mutate(rank=row_number(), `term frequency`= n/total) %>%
  arrange(desc(`term frequency`))-> all_words
#all_words has all information we need to do tf_idf analysis.

ggplot(all_words, aes(n/total, fill = article)) +
  geom_histogram(show.legend = FALSE) +
  facet_wrap(~article, nrow = 2, scales = "free_y")+
  ggtitle("Frequency VS Count")
```

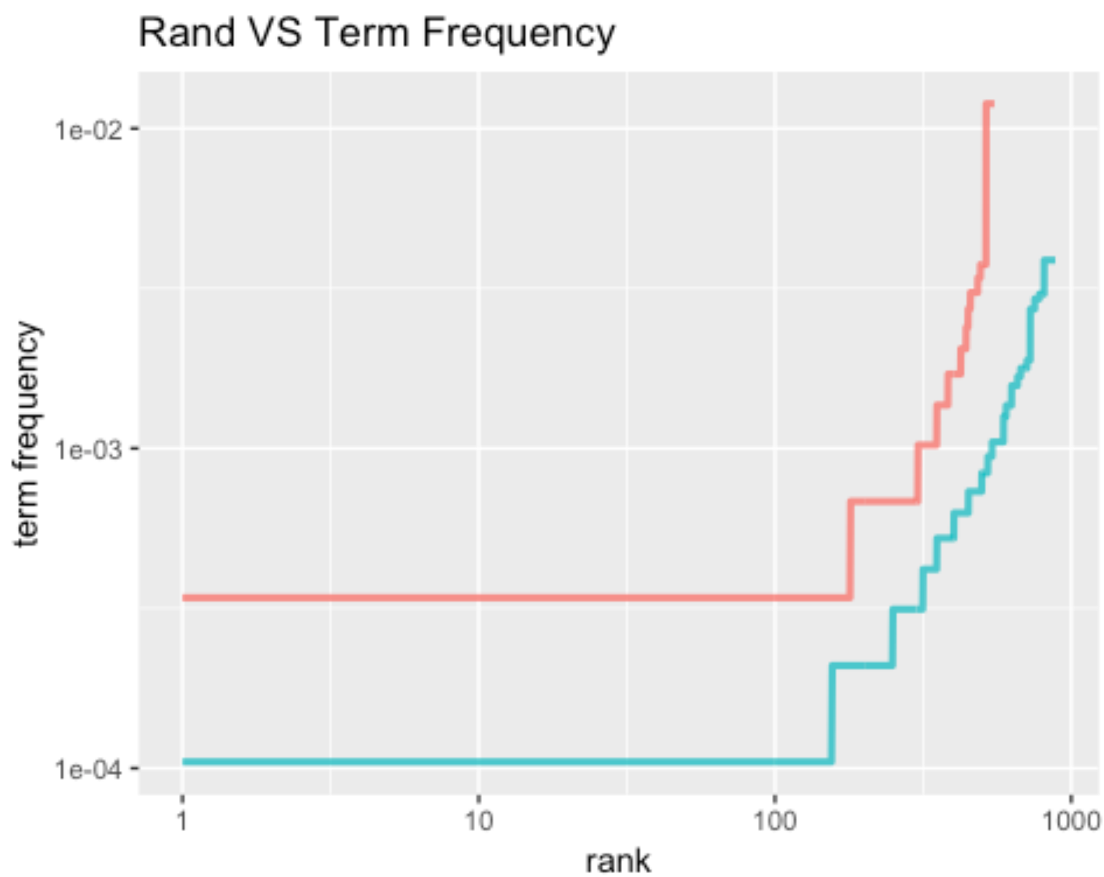
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
correlaid_words <- correlaid_words %>% mutate(proportion=n/sum(n))
correlaid_words2 <- correlaid_words2 %>% mutate(proportion=n/sum(n))
```

We can see that the tails are not so long and these two article exhibit similar distribution. Their peaks are at similar points.

```
ggplot(all_words,aes(rank, `term frequency`, color = article)) +
geom_line(size = 1.1, alpha = 0.8, show.legend = FALSE) +
  scale_x_log10() +
  scale_y_log10()+
  ggtitle("Rand VS Term Frequency")
```



The result is totally opposite to the Zipf's Law, which states that a word appears is inversely proportional to its rank.

Then we apply bind_tf_idf function to find the important words for the content of each document by decreasing the weight for commonly used words and increasing the weight for words that not used very much.

```
all_words <- all_words %>% bind_tf_idf(word, article, n)
all_words

## # A tibble: 1,433 x 9
## # Groups:   article [2]
##    word      n article    total  rank `term frequency`     tf   idf
tf_idf
##    <chr> <int> <chr>      <int> <int>            <dbl>  <dbl> <dbl>
<dbl>
##  1 data     35 Data-Story  2930   516           0.0119 0.0119 -3.28 -
0.0391
##  2 data     35 Data-Story  2930   517           0.0119 0.0119 -3.28 -
0.0391
##  3 data     35 Data-Story  2930   518           0.0119 0.0119 -3.28 -
0.0391
##  4 data     35 Data-Story  2930   519           0.0119 0.0119 -3.28 -
0.0391
```

```
## 5 data     35 Data-Story  2930   520          0.0119 0.0119 -3.28 -
0.0391
## 6 data     35 Data-Story  2930   521          0.0119 0.0119 -3.28 -
0.0391
## 7 data     35 Data-Story  2930   522          0.0119 0.0119 -3.28 -
0.0391
## 8 data     35 Data-Story  2930   523          0.0119 0.0119 -3.28 -
0.0391
## 9 data     35 Data-Story  2930   524          0.0119 0.0119 -3.28 -
0.0391
## 10 data    35 Data-Story  2930   525          0.0119 0.0119 -3.28 -
0.0391
## # ... with 1,423 more rows
```

N-grams and Correlations We want to check the words as bigrams from now on.

```
correlaid_bigrams <- correlaid_txt %>% unnest_tokens(bigram, text, token =
"ngrams", n = 2) #for p-value article
correlaid_bigrams2 <- correlaid_txt2 %>% unnest_tokens(bigram, text, token =
"ngrams", n = 2) #for data-story article

#Seperate the bigrams into two words
bigrams_separated <- correlaid_bigrams %>%
separate(bigram, c("word1", "word2"), sep = " ")
bigrams_separated2 <- correlaid_bigrams2 %>%
separate(bigram, c("word1", "word2"), sep = " ")
#Eliminate stop words
bigrams_filtered <- bigrams_separated %>% filter(!word1 %in% stop_words$word)
%>% filter(!word2 %in% stop_words$word)
bigrams_filtered2 <- bigrams_separated2 %>% filter(!word1 %in%
stop_words$word) %>% filter(!word2 %in% stop_words$word)
#Then unite them into bigrams
bigrams_united <- bigrams_filtered %>% unite(bigram, word1, word2, sep = " ")
bigrams_united2 <- bigrams_filtered2 %>% unite(bigram, word1, word2, sep = "
")
bigrams_united %>% mutate(article="P-Value") -> bigrams_united
bigrams_united2 %>% mutate(article="Data-Story") -> bigrams_united2
```

Then we apply bind_tf_idf function to find the important bigrams.

```
total_bigrams <- rbind(bigrams_united,bigrams_united2)
total_bigrams %>%
  mutate(n=n()) %>%
  bind_tf_idf(bigram, article, n)

##                        bigram    article   n          tf        idf
## 1             null hypothesis    P-Value 439 0.003891051 -2.3025851
## 2             null hypothesis    P-Value 439 0.003891051 -2.3025851
## 3             null hypothesis    P-Value 439 0.003891051 -2.3025851
## 4                  text books    P-Value 439 0.003891051  0.6931472
## 5              power analysis    P-Value 439 0.003891051  0.6931472
```

```
## 6           analysis software    P-Value 439 0.003891051  0.6931472
## 7             horizontal axis    P-Value 439 0.003891051  0.6931472
## 8            calculated based    P-Value 439 0.003891051  0.6931472
## 9         normal distribution    P-Value 439 0.003891051  0.6931472
## 10                sample size    P-Value 439 0.003891051 -1.0986123
```

```
## 258            standard steps Data-Story 439 0.005494505  0.6931472
## 259               data driven Data-Story 439 0.005494505  0.6931472
## 260            driven project Data-Story 439 0.005494505  0.6931472
## 261            exemplary data Data-Story 439 0.005494505  0.6931472
## 262           data journalism Data-Story 439 0.005494505  0.6931472
## 263            workflow we'll Data-Story 439 0.005494505  0.6931472
## 264              bbsr germany Data-Story 439 0.005494505  0.6931472
## 265            commented code Data-Story 439 0.005494505  0.6931472
## 266               github page Data-Story 439 0.005494505  0.0000000
## 267       makes collaboration Data-Story 439 0.005494505  0.6931472
```

```
##           tf_idf
## 1   -0.008959475
## 2   -0.008959475
## 3   -0.008959475
## 4    0.002697071
## 5    0.002697071
## 6    0.002697071
## 7    0.002697071
## 8    0.002697071
## 9    0.002697071
## 10  -0.004274756

## 258  0.003808501
## 259  0.003808501
## 260  0.003808501
## 261  0.003808501
## 262  0.003808501
## 263  0.003808501
## 264  0.003808501
## 265  0.003808501
## 266  0.000000000
## 267  0.003808501
```

Using bigrams to do sentiments analysis. If we do seperate analysis on both article about "not" words.

```
AFINN <- get_sentiments("afinn")
not_words <- bigrams_separated %>%
filter(word1 == "not") %>%
inner_join(AFINN, by = c(word2 = "word")) %>% count(word2, score, sort =
```

```
TRUE) %>% ungroup()

not_words %>%
  mutate(contribution = n * score) %>%
  arrange(desc(abs(contribution)))
```

```
## # A tibble: 1 x 4
##   word2 score     n contribution
##   <chr> <int> <int>        <int>
## 1 true      2     3            6
```

In this P-Value article, only one word is follwed by “not”.

```
not_words2 <- bigrams_separated2 %>%
  filter(word1 == "not") %>%
  inner_join(AFINN, by = c(word2 = "word")) %>%
  count(word2, score, sort = TRUE) %>% ungroup()

not_words2 %>%
  mutate(contribution = n * score) %>%
  arrange(desc(abs(contribution)))
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: word2 <chr>, score <int>, n <int>,
## #   contribution <int>
```

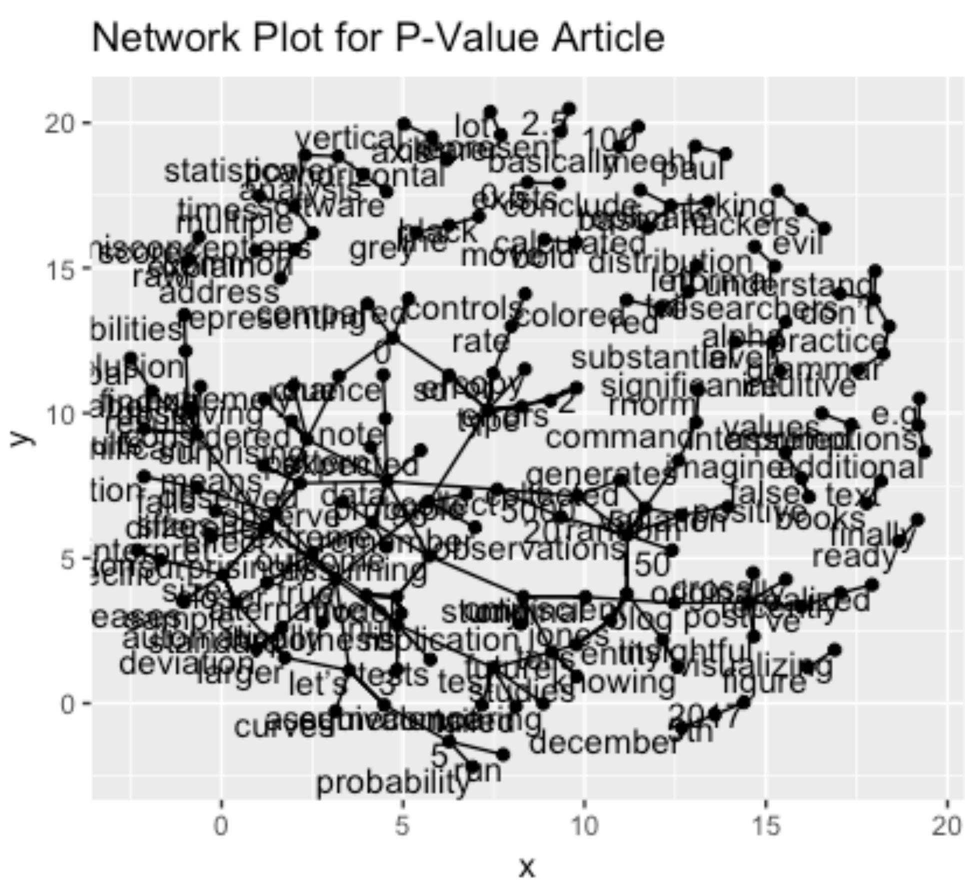
And in this Data to Story article, on word is followed by “not”.

Network of Bigrams

```
library(igraph)
```

```
##
## Attaching package: 'igraph'

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

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

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

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

```r
library(ggraph)

#Network for P-Value Article
bigrams_separated <-
  correlaid_bigrams %>%
  mutate(n=n()) %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)
bigrams_separated %>% graph_from_data_frame() -> bigram_graph
set.seed(2018)
ggraph(bigram_graph, layout = "fr") +
  geom_edge_link() +
  geom_node_point() +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1)+
  coord_fixed(0.8)+
  ggtitle("Network Plot for P-Value Article")
```




```r
#Network for Data to Story Article
bigrams_separated2 <-
  correlaid_bigrams2 %>%
  mutate(n=n()) %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
```

```r
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)
bigrams_separated2 %>% graph_from_data_frame() -> bigram_graph2
set.seed(2018)
ggraph(bigram_graph2, layout = "fr") +
  geom_edge_link() +
  geom_node_point() +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1)+
  coord_fixed(0.8)+
  ggtitle("Network Plot for Data to Story Article")
```