correlaid_text_analysis

Xiang XU, Jing(Mira) Tang, Ningze(Summer) ZU, Jianhao(Miller) Yan

November 3, 2018

Scraping webpages

We see here we scrap three articles from correlaid.com and combine them into aone data frame.

These three articles are :
* understand-p-values; * blockchain-explained; * music-with-r.

```
## 'data.frame':    3 obs. of  2 variables:
##  $ article: chr  "doc1" "doc2" "doc3"
##  $ text   : chr  "Skip to main content\n\nToggle navigation Menu\n\n•\n• Zurück zu Correlaid.org\n• 
```

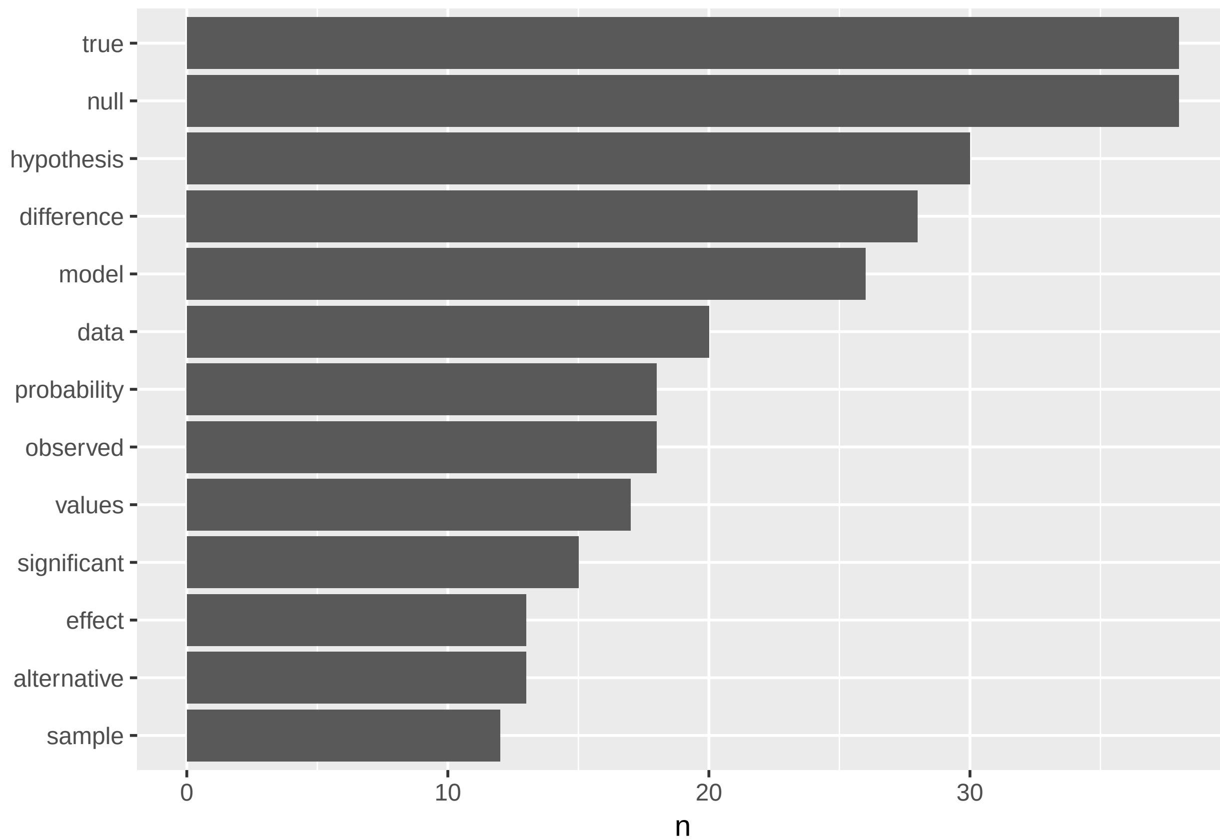
tidy text

We tidy the text by `unnest_tokens` function and filter the useful words, which make sense.

```
##   article       word
## 1    doc1       skip
## 2    doc1       main
## 3    doc1    content
## 4    doc1     toggle
## 5    doc1 navigation
## 6    doc1       menu
```

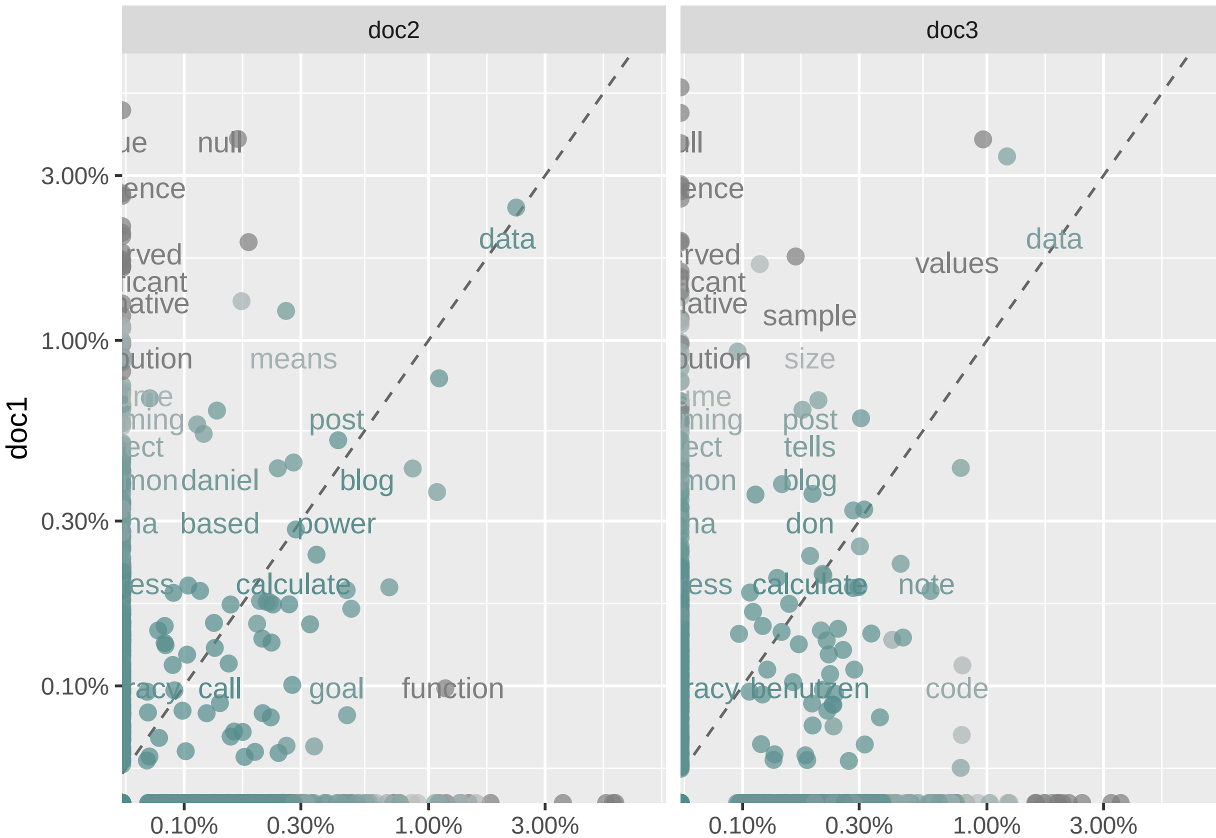
look at single word frequency and visualize

Here look at the first article p-value and cont word frequencies.



Then calculate word frequencies of three articles .

Plotting and comparing the three articles



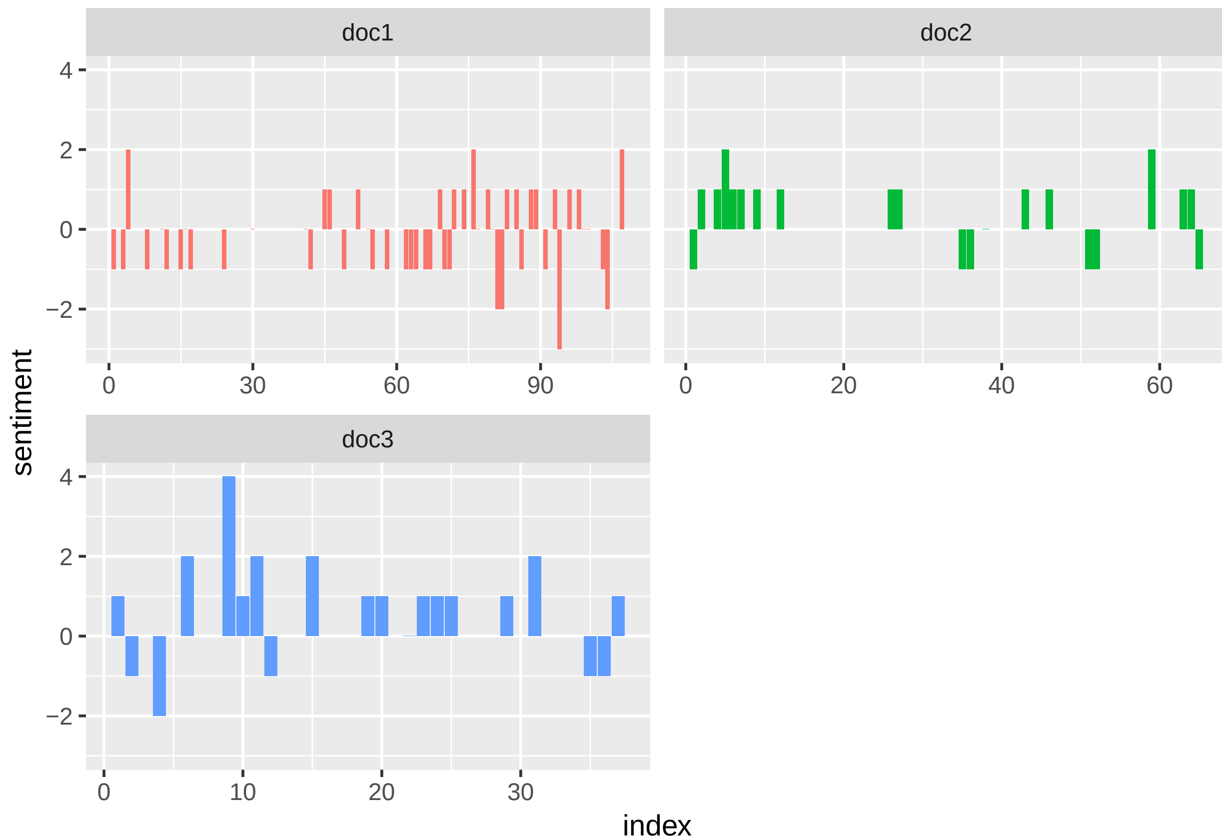

```
##
##  Pearson's product-moment correlation
##
## data:  proportion and doc1
## t = -0.90635, df = 800, p-value = 0.365
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.10103135  0.03728257
## sample estimates:
##         cor
## -0.03202772
```

```
##
##  Pearson's product-moment correlation
##
## data:  proportion and doc1
## t = -1.3419, df = 800, p-value = 0.18
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.11623528  0.02191054
## sample estimates:
##         cor
## -0.04738898
```

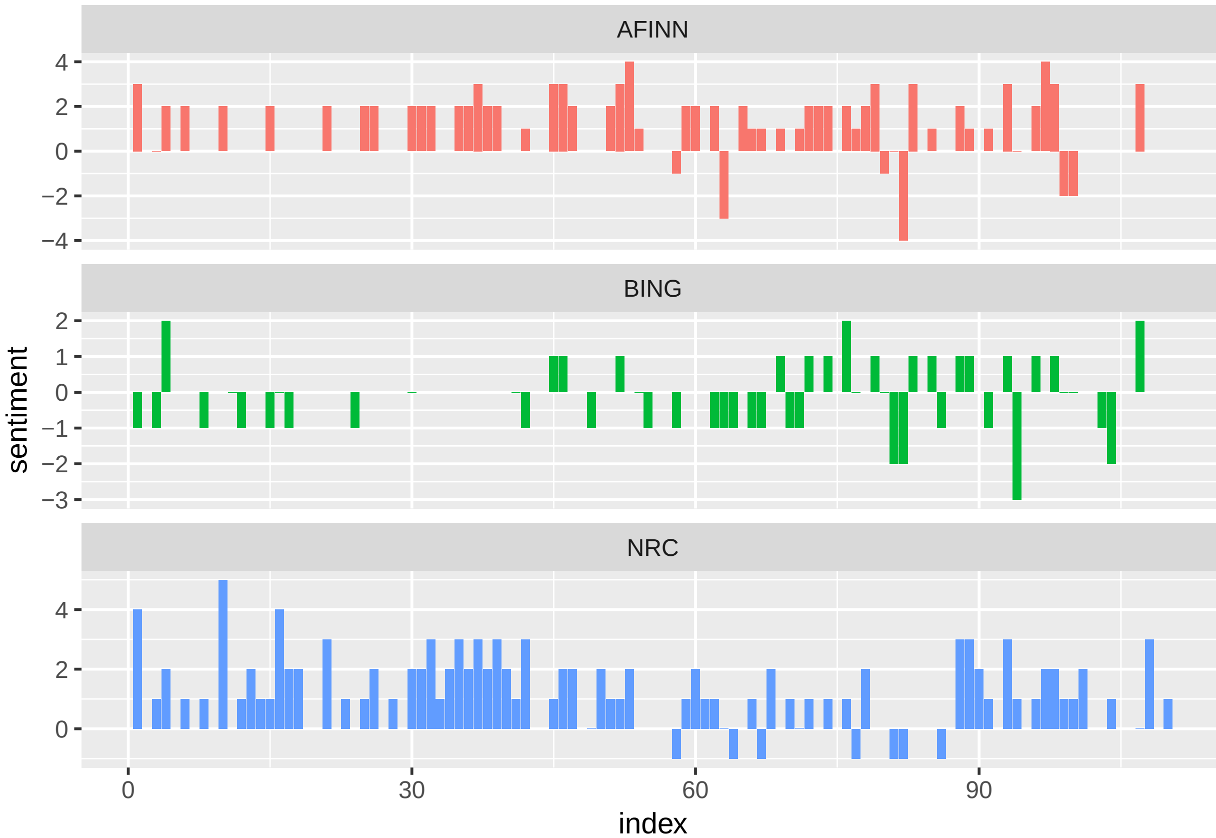
As we saw in the plots, the word frequencies have little frequencies in three articles.

Sentiment analysis

bing sentiment analysis

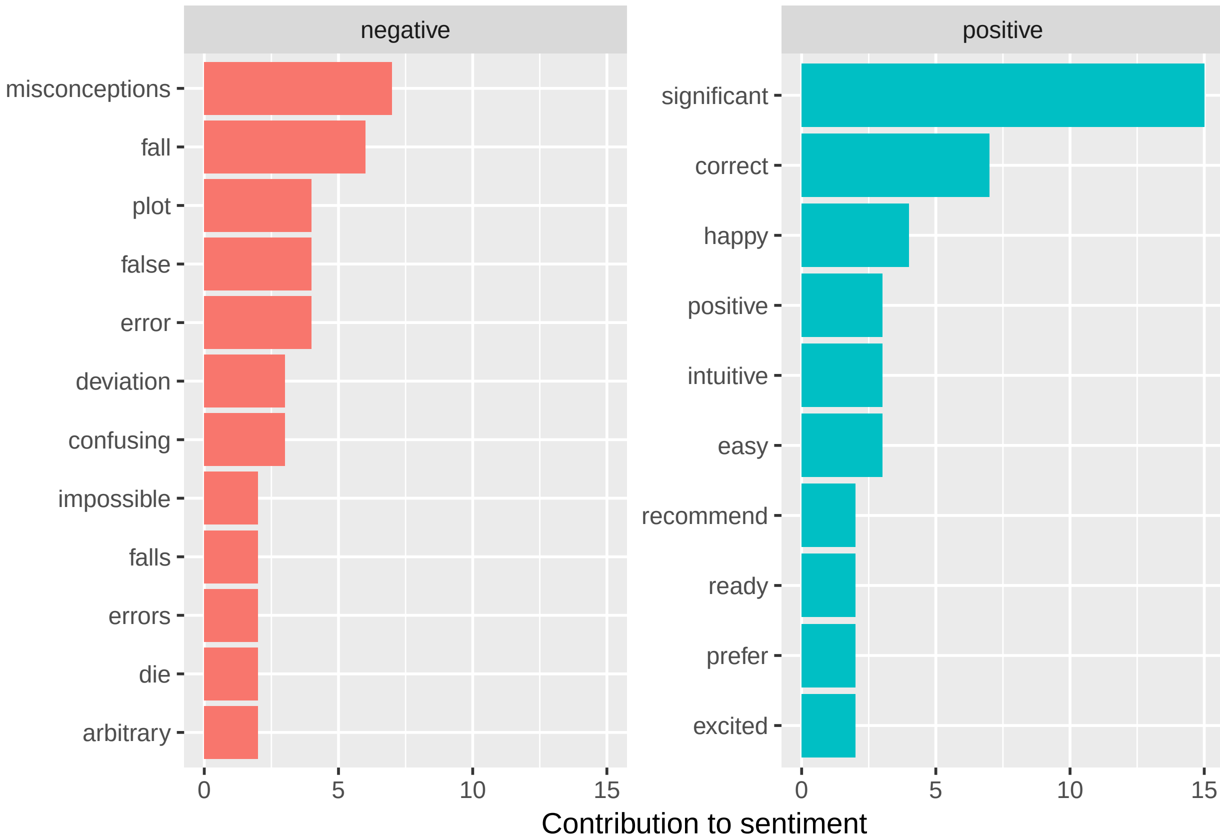


Comparing the three sentiment dictionaries

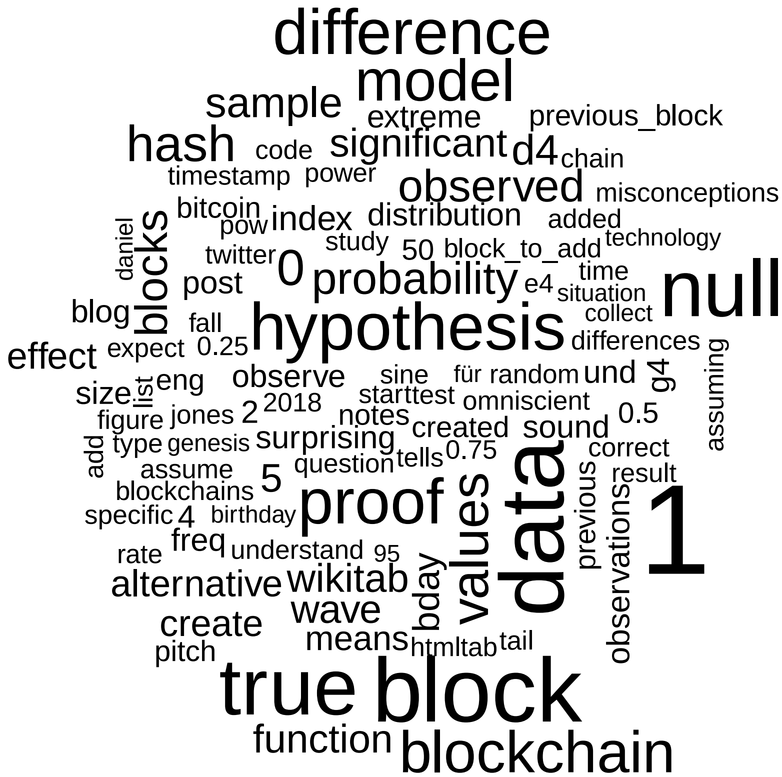


Most common sentiment words

```
## # A tibble: 79 x 3
##    word           sentiment     n
##    <chr>          <chr>     <int>
##  1 significant    positive     15
##  2 correct        positive      7
##  3 misconceptions negative      7
##  4 fall           negative      6
##  5 error          negative      4
##  6 false          negative      4
##  7 happy          positive      4
##  8 plot           negative      4
##  9 confusing      negative      3
## 10 deviation      negative      3
## # ... with 69 more rows
```

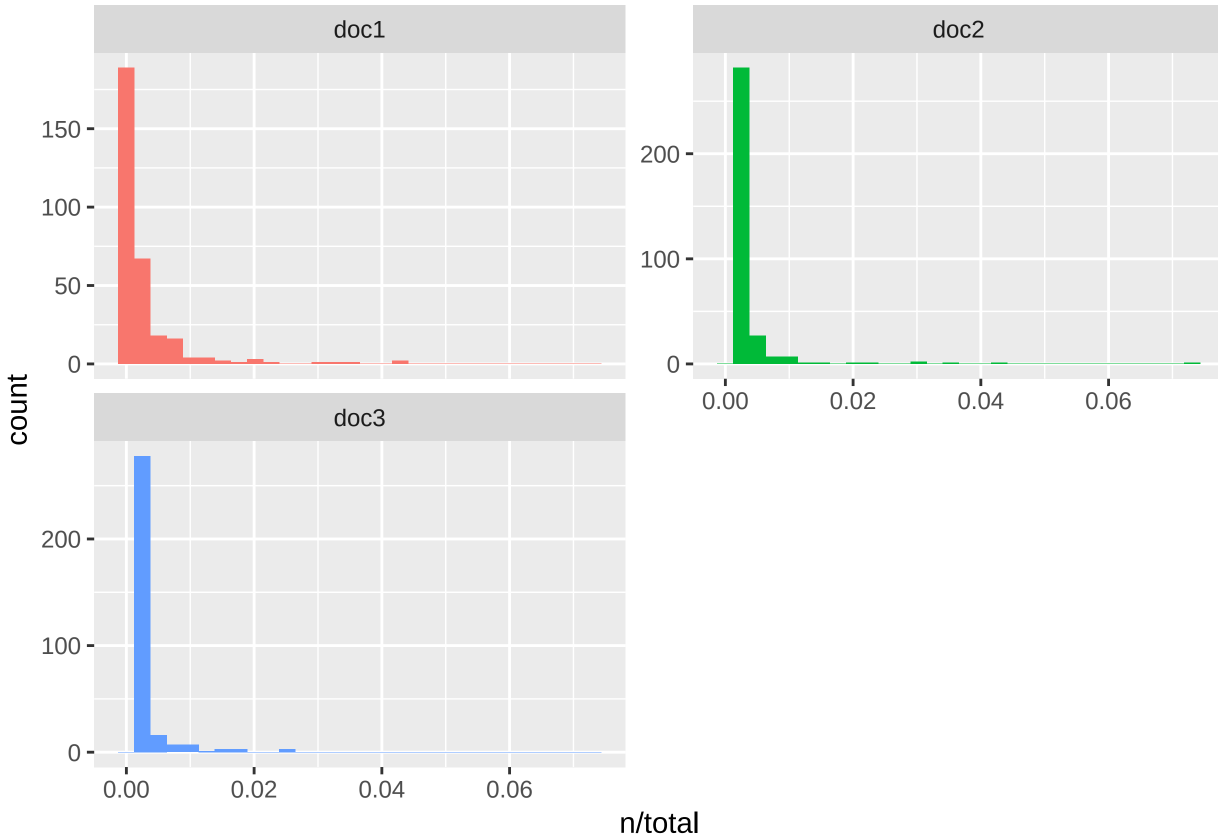

negative
positive
misconceptions
fall
plot
false
error
deviation
confusing
impossible
falls
errors
die
arbitrary
significant
correct
happy
positive
intuitive
easy
recommend
ready
prefer
excited
0
5
10
15
Contribution to sentiment

Wordclouds

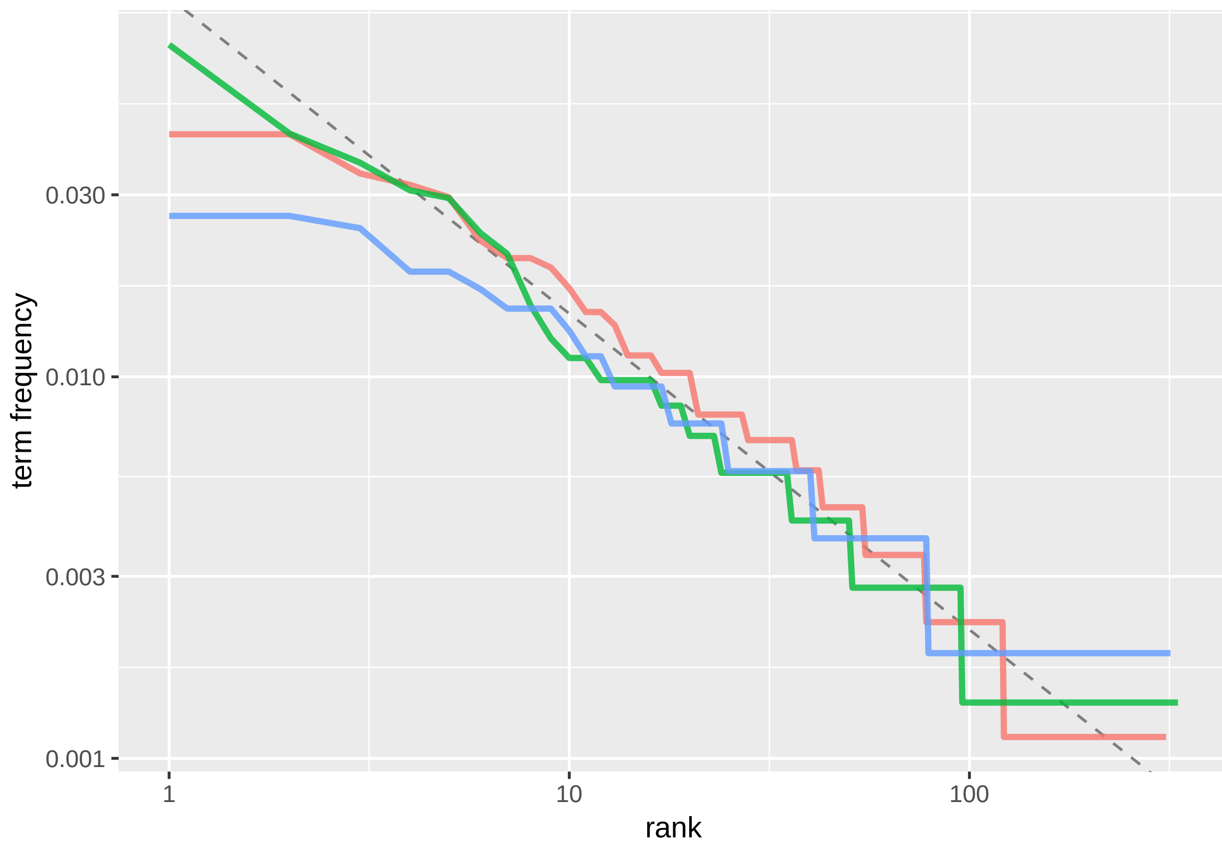



negative
misconception
lose
illiterate
object
odd
evil
loose
critical
fallen
misunderstand
confusion
confusing
impossible
deviation
plot
arbitrary
errors
hard
complaining
disrupt
die
false
error
fall
falls
sneaky
broken
reject
misconceptions
difficulty
wrong
hype
significant
crazy
unfamiliar
supported
master
prefer
correct
positive
intriguing
grateful
capable
efficient
handy
intuitive
happy
easy
correctly
easier
trust
versatile
intelligence
enjoy
fair
excited
recommend
imaginative
sufficient
leading
clearer
ready
brilliant
meaningful
accurate
insightful
complement
dedicated
modern
congratulate
conveniently
successfully
incredibly
nicely
genius
secure
promise
pretty
realistic
positive

tf-idf

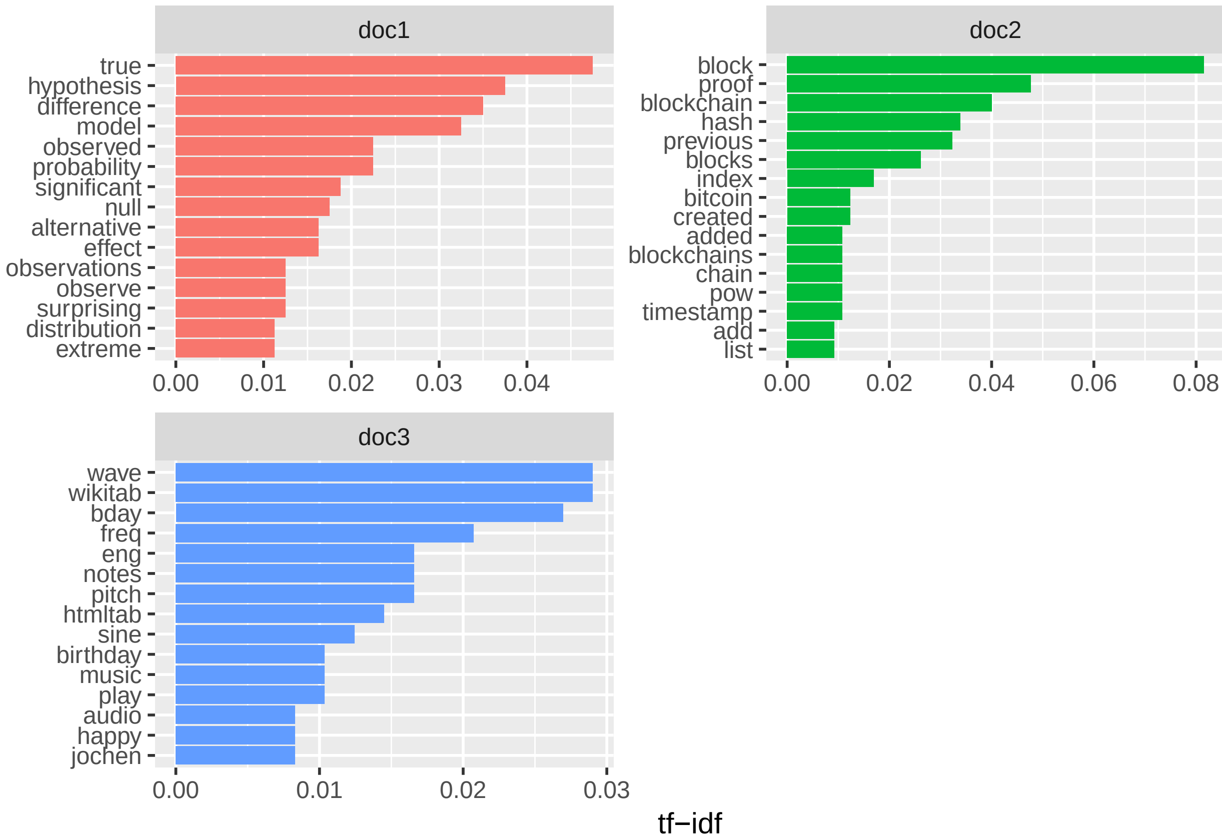



```
## (Intercept)  log10(rank)
##  -1.0057750  -0.8284333
```



tf-idf function

```
## # A tibble: 960 x 6
##    article word           n     tf   idf tf_idf
##    <chr>   <chr>      <int>  <dbl> <dbl>  <dbl>
##  1 doc2    block         53 0.0742  1.10 0.0815
##  2 doc2    proof         31 0.0434  1.10 0.0477
##  3 doc1    true          38 0.0432  1.10 0.0475
##  4 doc2    blockchain    26 0.0364  1.10 0.0400
##  5 doc1    hypothesis    30 0.0341  1.10 0.0375
##  6 doc1    difference    28 0.0319  1.10 0.0350
##  7 doc2    hash          22 0.0308  1.10 0.0339
##  8 doc1    model         26 0.0296  1.10 0.0325
##  9 doc2    previous      21 0.0294  1.10 0.0323
## 10 doc3    wave          14 0.0264  1.10 0.0290
## # ... with 950 more rows
```



n-grams and correlations

We use `unnest_tokens` function to tokenize the articles into consecutive sequences of words, called n-grams. Here we focus on bigrams, aka two consecutive words.

As one might expect, a lot of the most common bigrams are pairs of common (uninteresting) words, such as of the and to be: what we call "stop-words" . This is a useful time to use tidyr's `separate()` and `unite()`, which splits a column into multiple based on a delimiter and reunite them. In this process we can remove cases where either is a stop-word.

Also, we clean the bigrams by `str_extract()` and `filter()` function to remove cases where either is NA, space or non-letter word.

Then we look at tf_idf of bigrams and visualize them.

```
## # A tibble: 10 x 6
##    article bigram                   n     tf   idf tf_idf
##    <chr>   <chr>                <int>  <dbl> <dbl>  <dbl>
##  1 doc1    null hypothesis         21 0.0843  1.10 0.0927
##  2 doc1    null model              15 0.0602  1.10 0.0662
##  3 doc2    previous block          13 0.0478  1.10 0.0525
##  4 doc1    alternative model        7 0.0281  1.10 0.0309
##  5 doc2    previous hash            7 0.0257  1.10 0.0283
##  6 doc1    omniscient jones         6 0.0241  1.10 0.0265
##  7 doc1    sample size              6 0.0241  1.10 0.0265
##  8 doc1    significant result       6 0.0241  1.10 0.0265
```

```
##   9 doc3    wikitab eng                  6 0.0231  1.10 0.0254
## 10 doc1    alternative hypothesis       5 0.0201  1.10 0.0221
```

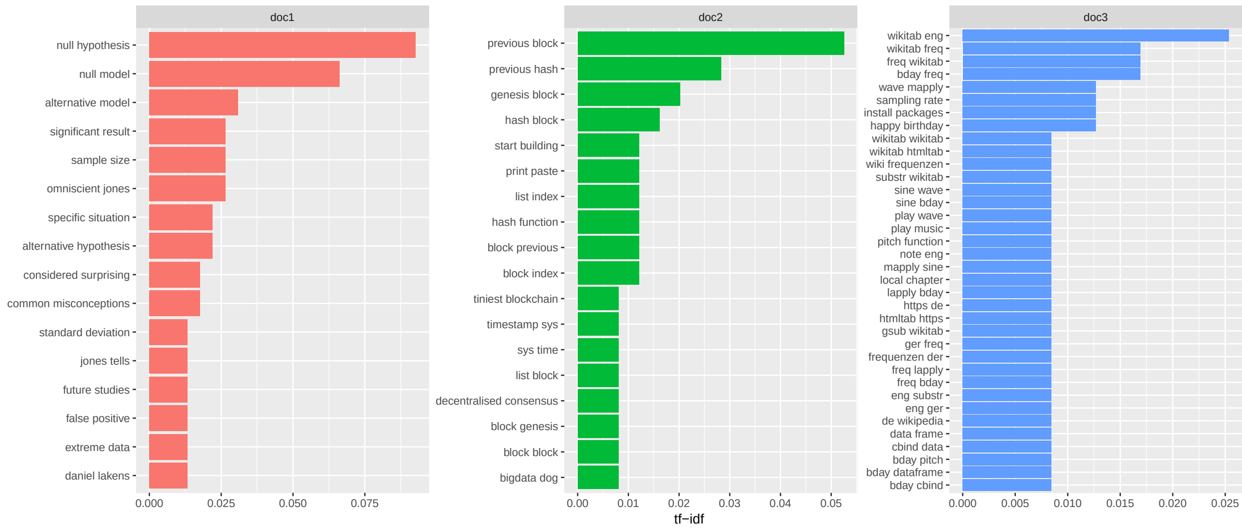


Figure: The 12 bigrams with the highest tf-idf

There are advantages and disadvantages to examining the tf-idf of bigrams rather than individual words. Pairs of consecutive words might capture structure that isn't present when one is just counting single words, and may provide context that makes tokens more understandable . However, the per-bigram counts are also sparser: a typical two-word pair is rarer than either of its component words. Thus, bigrams can be more useful when we have a larger text dataset.

- Using bigrams to provide context in sentiment analysis

```
## # A tibble: 0 x 4
## # ... with 4 variables: word1 <chr>, word2 <chr>, score <int>, nn <int>
```

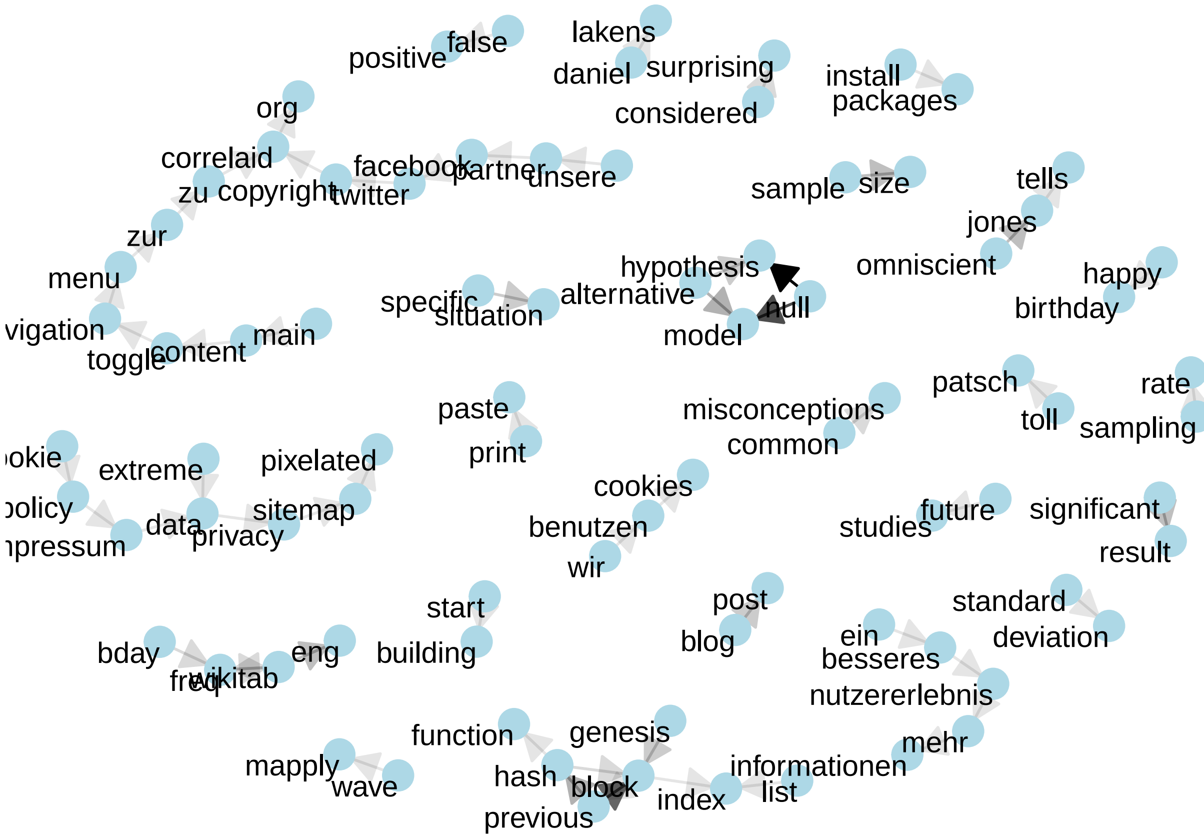
For these are three academic articles and there are not many bigrams with negative terms. So we can skip this part.

- Visualizing a network of bigrams with ggraph

```
## # A tibble: 6 x 3
##   word1       word2          n
##   <chr>       <chr>      <int>
## 1 null        hypothesis    21
## 2 null        model         15
## 3 previous    block         13
## 4 alternative model          7
## 5 previous    hash           7
## 6 omniscient  jones          6
```

```
## IGRAPH ffeb51b DN-- 82 62 --
## + attr: name (v/c), n (e/n)
## + edges from ffeb51b (vertex names):
##  [1] null       ->hypothesis     null       ->model
##  [3] previous   ->block          alternative->model
##  [5] previous   ->hash           omniscient ->jones
##  [7] sample     ->size           significant->result
##  [9] wikitab    ->eng            alternative->hypothesis
## [11] blog       ->post           genesis    ->block
```

```
## [13] specific   ->situation      bday       ->freq
## [15] common     ->misconceptions considered ->surprising
## + ... omitted several edges
```



- Counting and correlating pairs of words

```
##   article       word
## 1    doc1       skip
## 2    doc1       main
## 3    doc1    content
## 4    doc1     toggle
## 5    doc1 navigation
## 6    doc1       menu
```

```
## # A tibble: 294,912 x 3
##    item1      item2      n
##    <chr>      <chr>  <dbl>
##  1 main       skip       3
##  2 content    skip       3
##  3 toggle     skip       3
##  4 navigation skip       3
##  5 menu       skip       3
##  6 zur        skip       3
##  7 zu         skip       3
##  8 correlaid  skip       3
##  9 blog       skip       3
## 10 values     skip       3
```

```
## # ... with 294,902 more rows
```

- Pairwise correlation

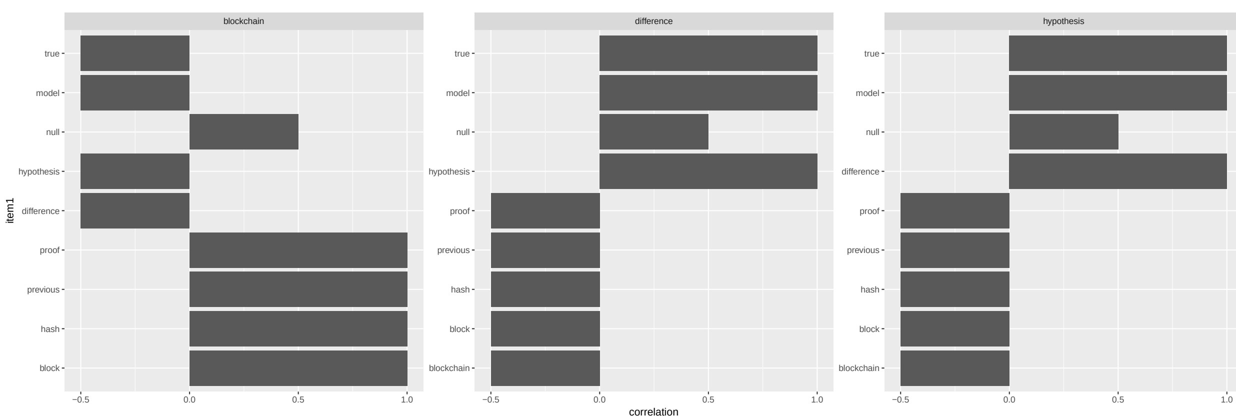
Find the phi coefficient between words based on how often they appear in the same article.

```
## # A tibble: 132 x 3
##    item1      item2      correlation
##    <chr>      <chr>            <dbl>
##  1 true       hypothesis       1.000
##  2 difference hypothesis       1.000
##  3 model      hypothesis       1.000
##  4 hypothesis true             1.000
##  5 difference true             1.000
##  6 model      true             1.000
##  7 hypothesis difference       1.000
##  8 true       difference       1.000
##  9 model      difference       1.000
## 10 hypothesis model            1.000
## # ... with 122 more rows
```

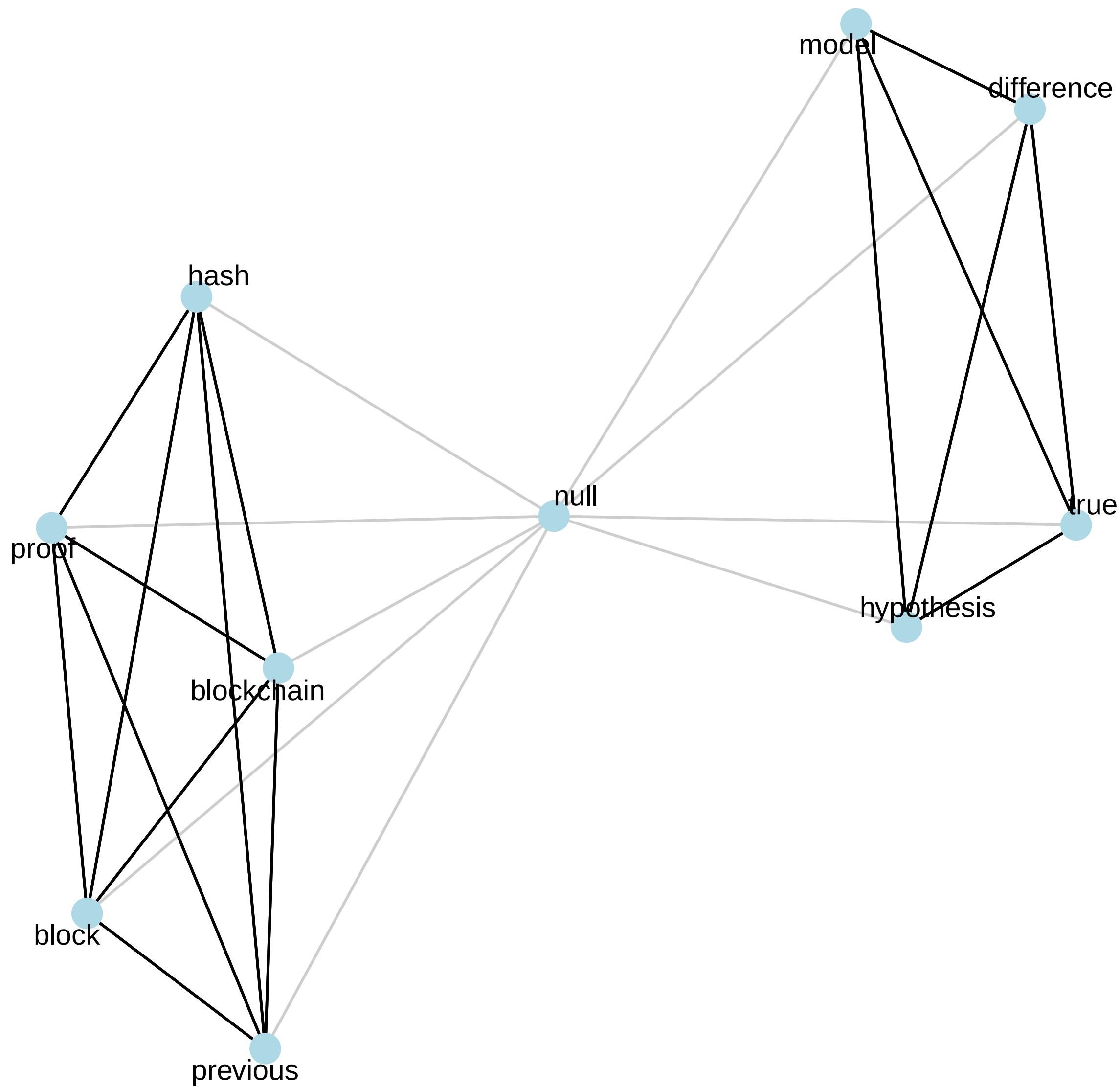
```
##  [1] "true"       "difference" "model"      "hypothesis" "block"
##  [6] "previous"   "hash"       "proof"      "blockchain" "null"
## [11] "data"       "values"
```

```
##  [1] "hypothesis" "true"       "difference" "model"      "blockchain"
##  [6] "block"      "previous"   "hash"       "proof"      "null"
## [11] "values"     "data"
```

Let's pick particular interesting words and find the other words most associated with them.



Visualize the correlations and clusters of words.




```
library(topicmodels)

text_dtm <- sentitext %>%
  count(article, word, sort = TRUE)%>%
  ungroup()%>%
  cast_dtm(article, word, n)

text_dtm
```

```
## <<DocumentTermMatrix (documents: 3, terms: 880)>>
## Non-/sparse entries: 1059/1581
## Sparsity           : 60%
## Maximal term length: 64
## Weighting          : term frequency (tf)
```

```
text_lda <- LDA(text_dtm,k =7, control = list(seed = 1234))
text_lda
```

```
## A LDA_VEM topic model with 7 topics.
```

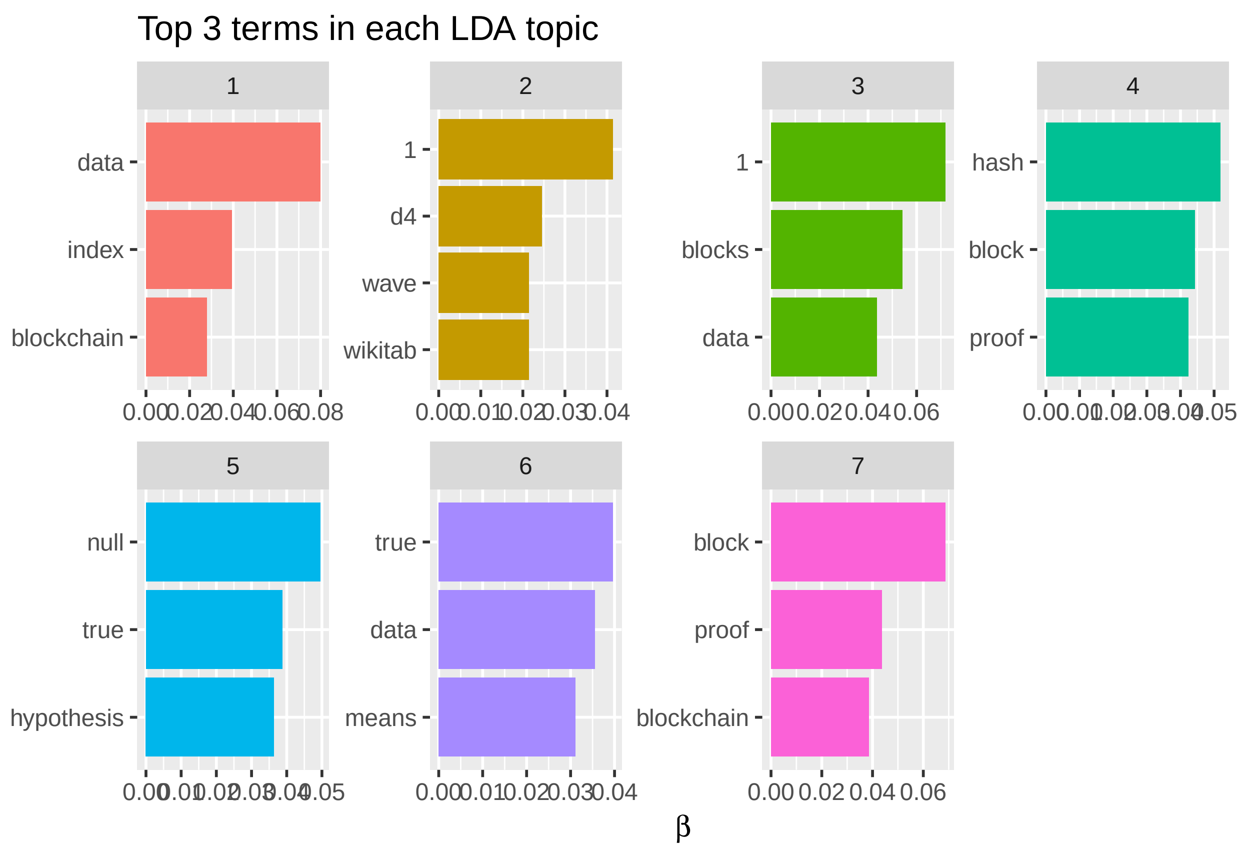
```
tidy_lda <- tidy(text_lda)
tidy_lda
```

```
## # A tibble: 6,160 x 3
##    topic term      beta
##    <int> <chr>    <dbl>
##  1     1 block 2.66e- 2
##  2     2 block 1.73e-38
##  3     3 block 2.37e- 2
##  4     4 block 4.43e- 2
##  5     5 block 1.58e-38
##  6     6 block 3.97e-35
##  7     7 block 6.86e- 2
##  8     1 null  8.41e- 3
##  9     2 null  4.76e-36
## 10     3 null  2.32e- 3
## # ... with 6,150 more rows
```

```
top_terms <- tidy_lda %>%
  group_by(topic) %>%
  top_n(3, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)
top_terms
```

```
## # A tibble: 22 x 3
##    topic term         beta
##    <int> <chr>       <dbl>
##  1     1 data       0.0798
##  2     1 index      0.0393
##  3     1 blockchain 0.0279
##  4     2 1          0.0414
##  5     2 d4         0.0245
##  6     2 wave       0.0215
##  7     2 wikitab    0.0215
##  8     3 1          0.0718
##  9     3 blocks     0.0541
## 10     3 data       0.0436
## # ... with 12 more rows
```

```
top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  group_by(topic, term) %>%
  arrange(desc(beta)) %>%
  ungroup() %>%
  mutate(term = factor(paste(term, topic, sep = "__"),
                       levels = rev(paste(term, topic, sep = "__")))) %>%
  ggplot(aes(term, beta, fill = as.factor(topic))) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  scale_x_discrete(labels = function(x) gsub("__.+$", "", x)) +
  labs(title = "Top 3 terms in each LDA topic",
       x = NULL, y = expression(beta)) +
  facet_wrap(~ topic, ncol = 4, scales = "free")
```

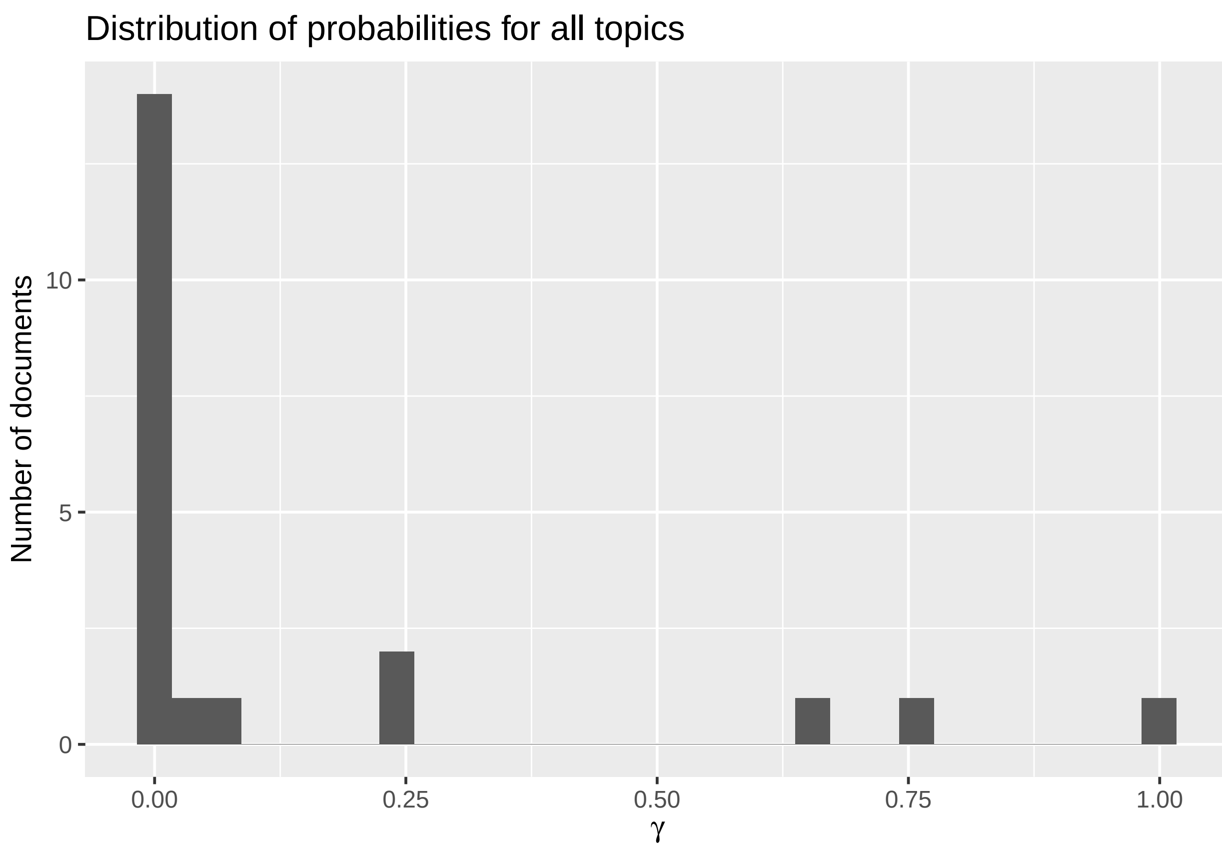



```r
lda_gamma <- tidy(text_lda, matrix = "gamma")
lda_gamma
```

```
## # A tibble: 21 x 3
##    document topic     gamma
##    <chr>    <int>     <dbl>
##  1 doc2         1 0.0446
##  2 doc1         1 0.0000584
##  3 doc3         1 0.0000873
##  4 doc2         2 0.0000705
##  5 doc1         2 0.0000584
##  6 doc3         2 0.999
##  7 doc2         3 0.231
##  8 doc1         3 0.0000584
##  9 doc3         3 0.0000873
## 10 doc2         4 0.0755
## # ... with 11 more rows
```

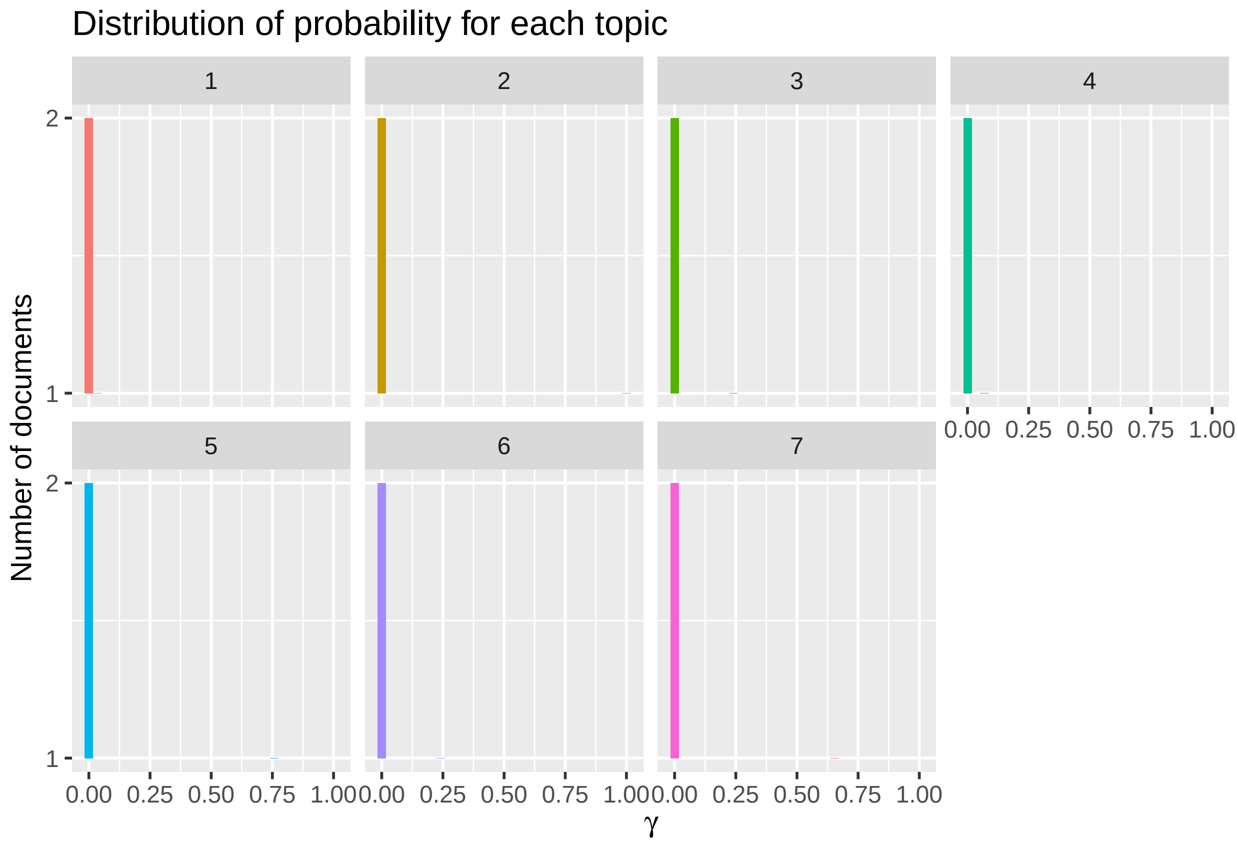
```r
ggplot(lda_gamma, aes(gamma)) +
  geom_histogram() +
  #scale_y_log10() +
  labs(title = "Distribution of probabilities for all topics",
       y = "Number of documents", x = expression(gamma))
```

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




```r
ggplot(lda_gamma, aes(gamma, fill = as.factor(topic))) +
  geom_histogram(show.legend = FALSE) +
  facet_wrap(~ topic, ncol = 4) +
  scale_y_log10() +
  labs(title = "Distribution of probability for each topic",
       y = "Number of documents", x = expression(gamma))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Transformation introduced infinite values in continuous y-axis
## Warning: Removed 196 rows containing missing values (geom_bar).
```



Summary

By using a combination of network analysis, tf-idf, and topic modeling, we have come to a greater understanding of how datasets are related in these three articles from correlaid website. Specifically, we have more information now about how keywords are connected to each other and which datasets are likely to be related. The topic model could be used to suggest keywords based on the words in the description field, or the work on the keywords could suggest the most important combination of keywords for certain areas of study.