correlaid_text_analysis

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

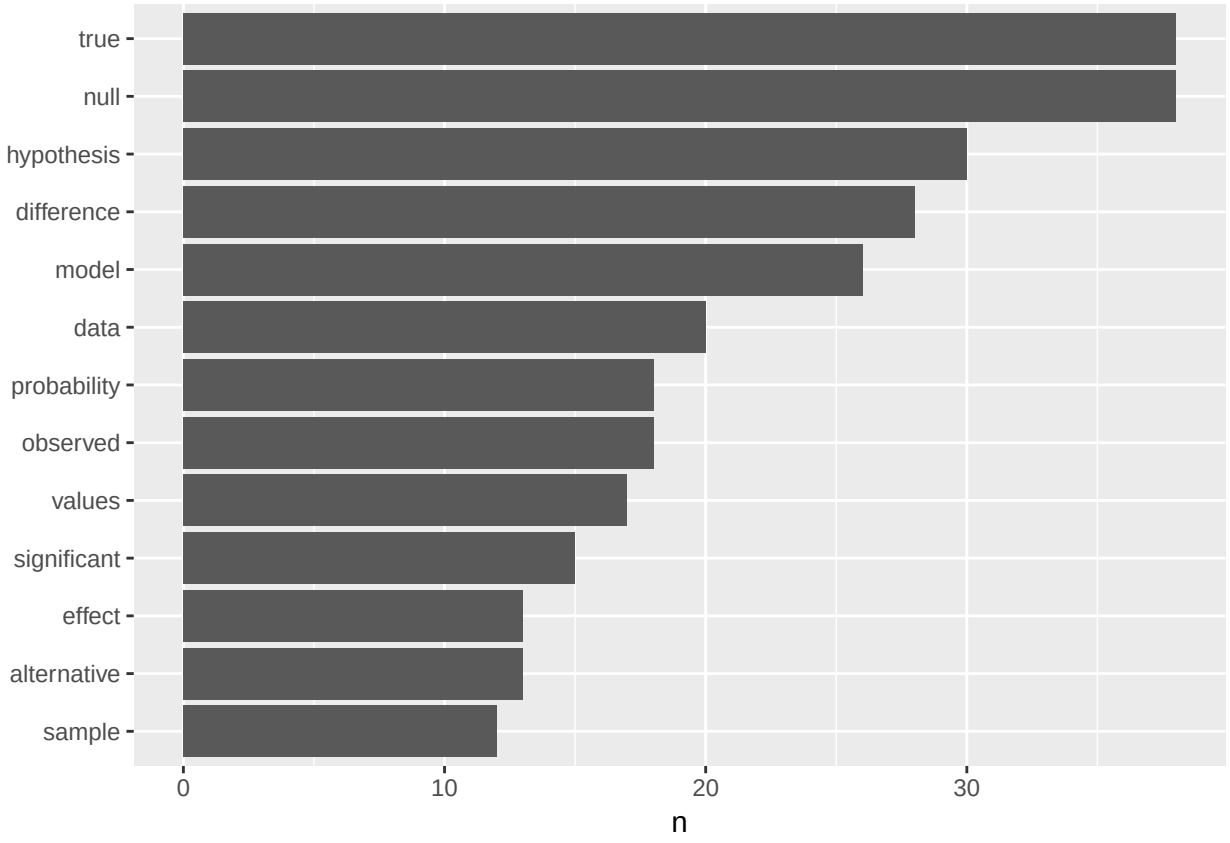
November 3, 2018

Scraping webpages

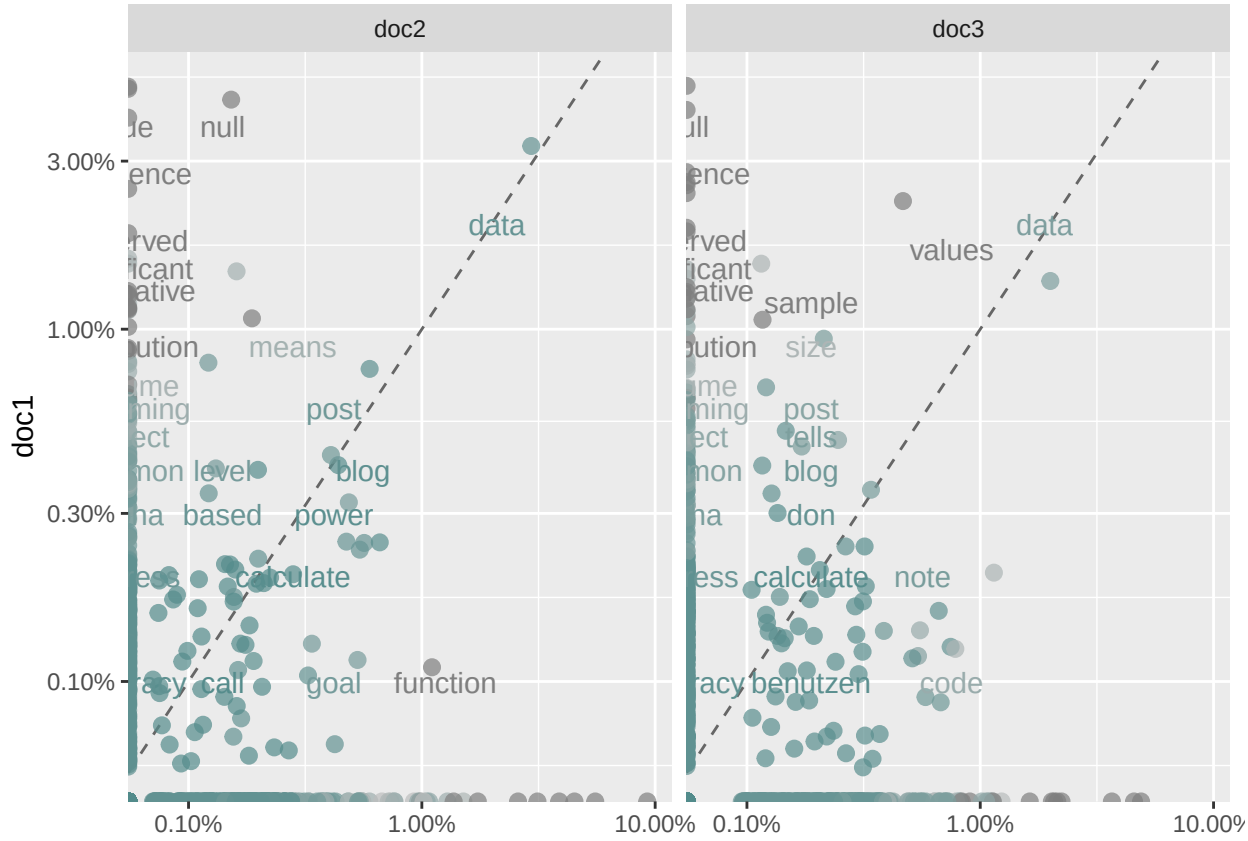
tidy text

look at single word frequency and visualize

first at doc1



plotting and comparing the three articles



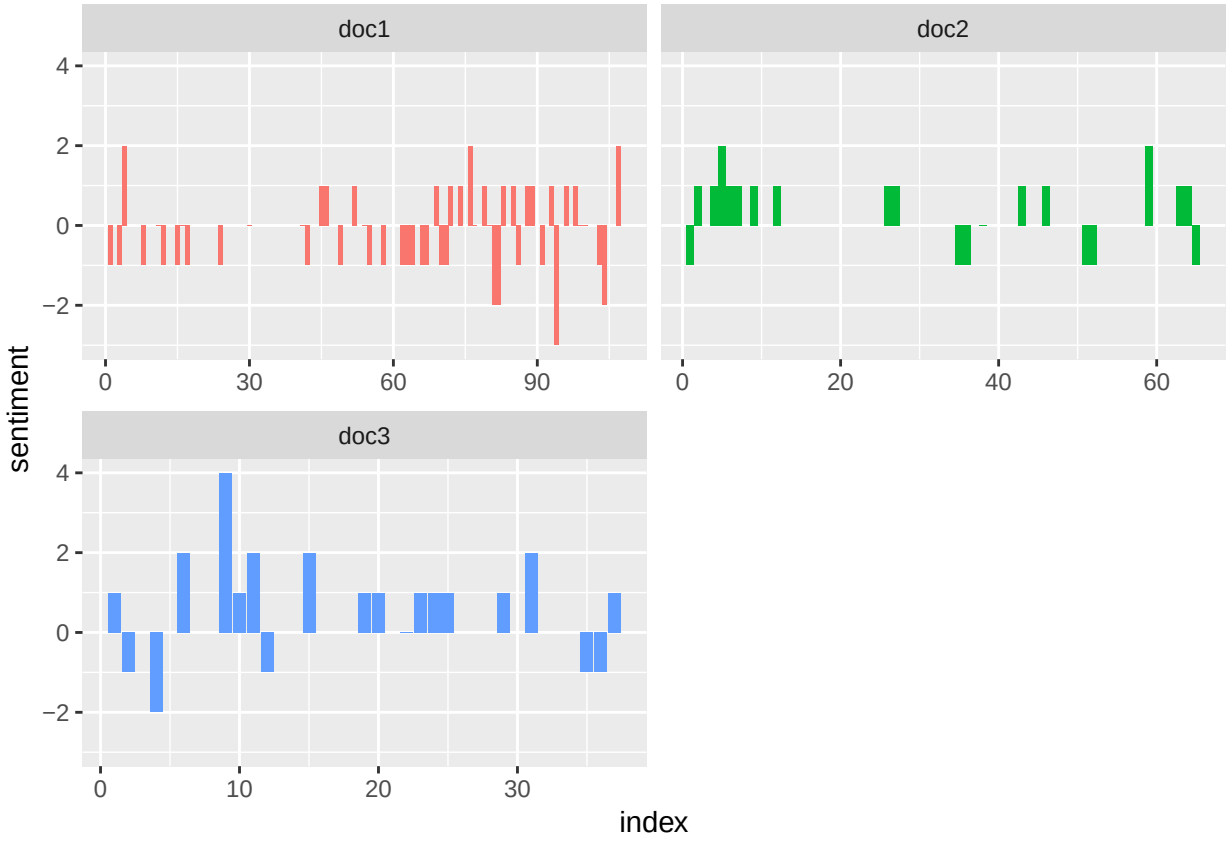

```
##
##  Pearson's product-moment correlation
##
## data:  proportion and doc1
## t = -0.90635, df = 800, p-value = 0.365
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.10103135  0.03728257
## sample estimates:
##         cor
## -0.03202772
```

```
##
##  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
```

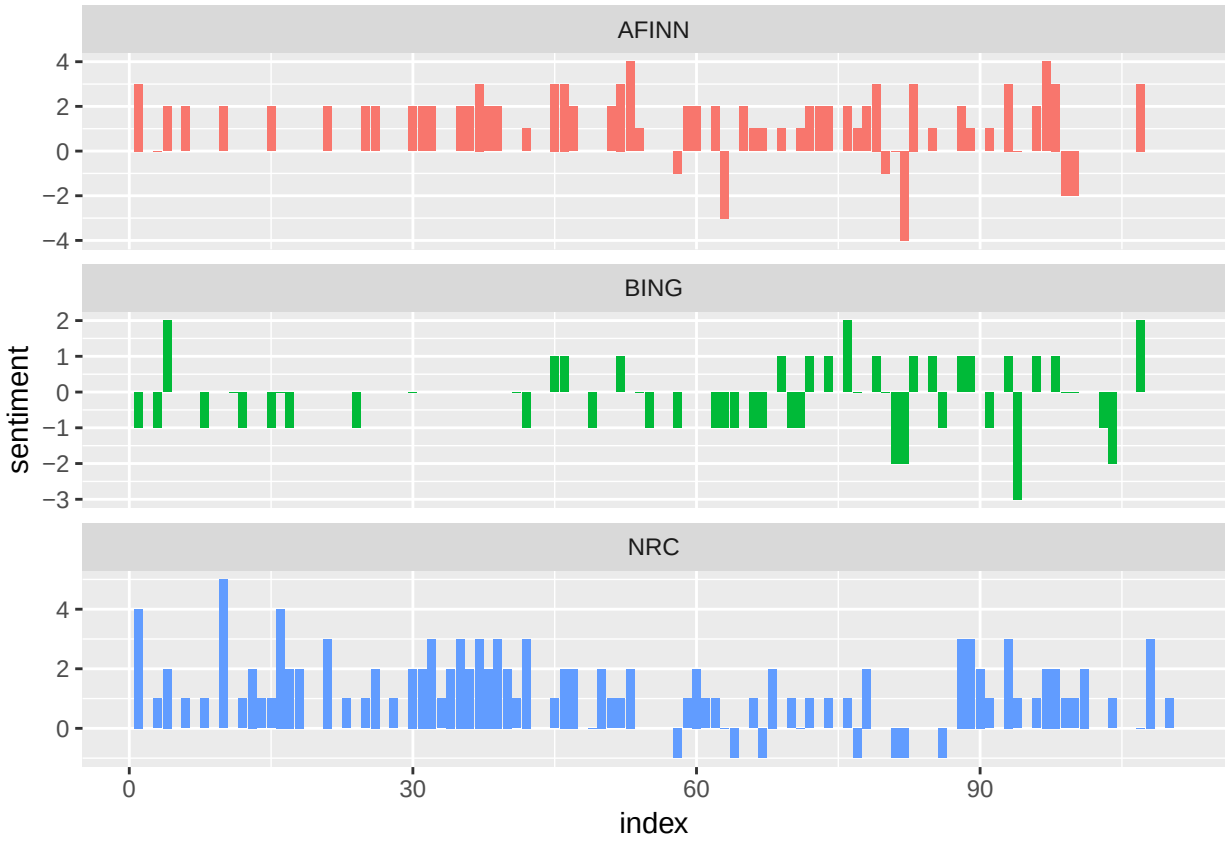
As we saw in the plots, the word frequencies have little frequencies in three articles.

Sentiment analysis

bing sentiment analysis

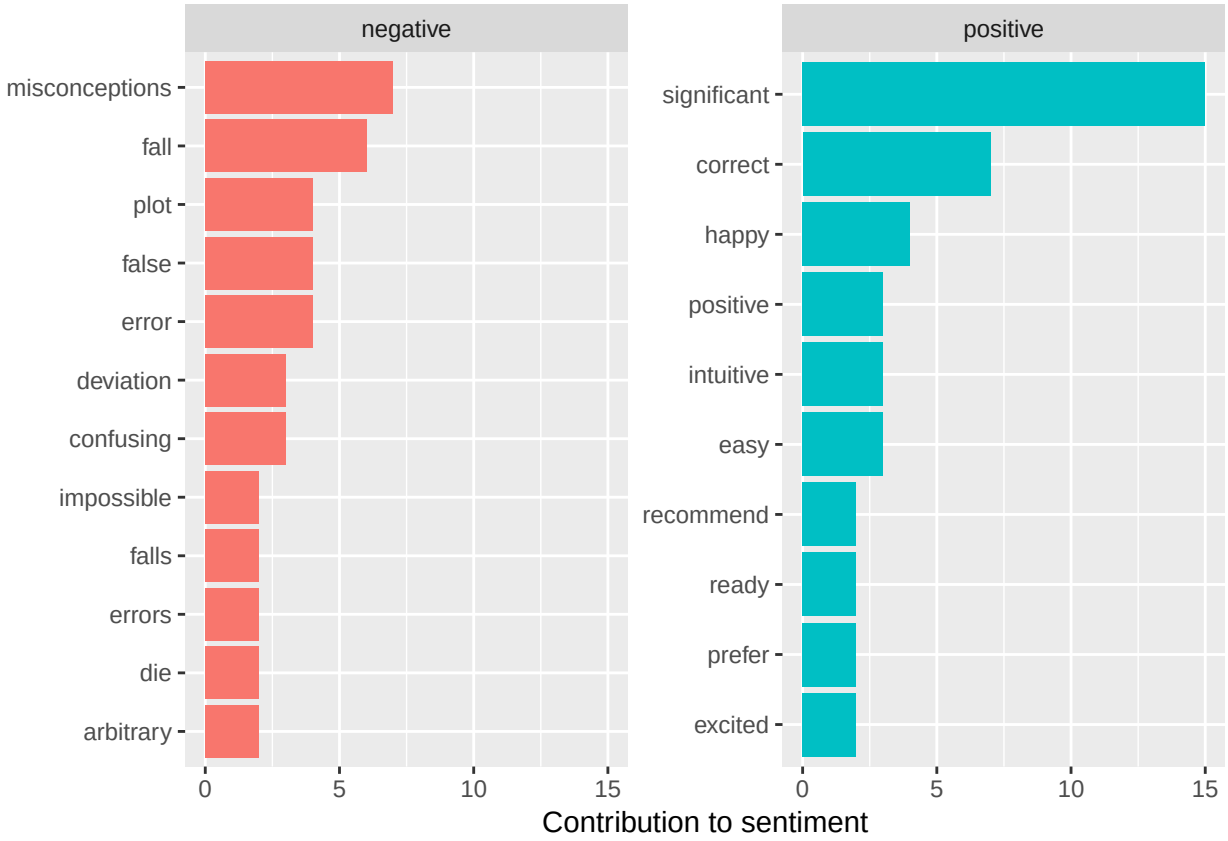


Comparing the three sentiment dictionaries

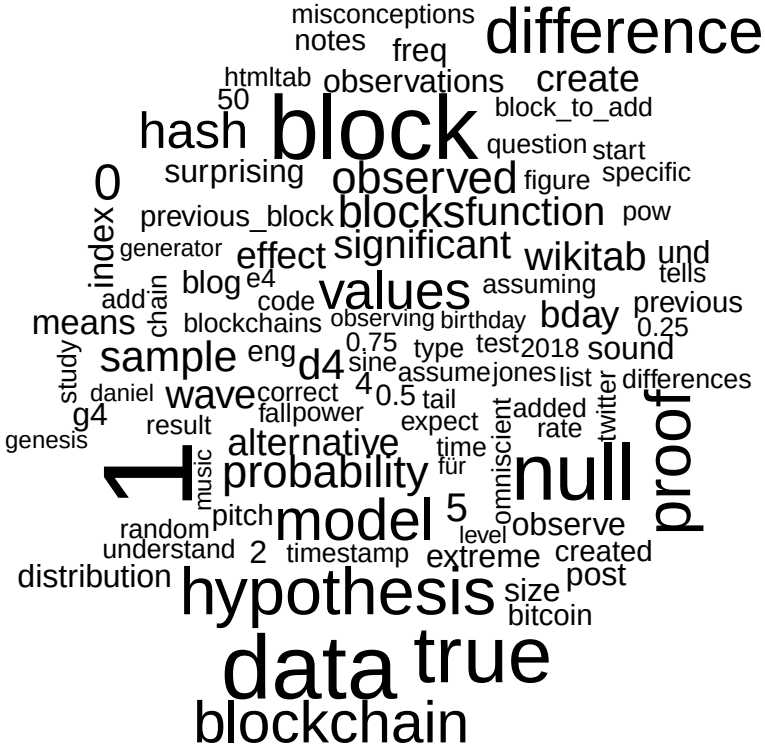


Most common sentiment words

```
## # 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

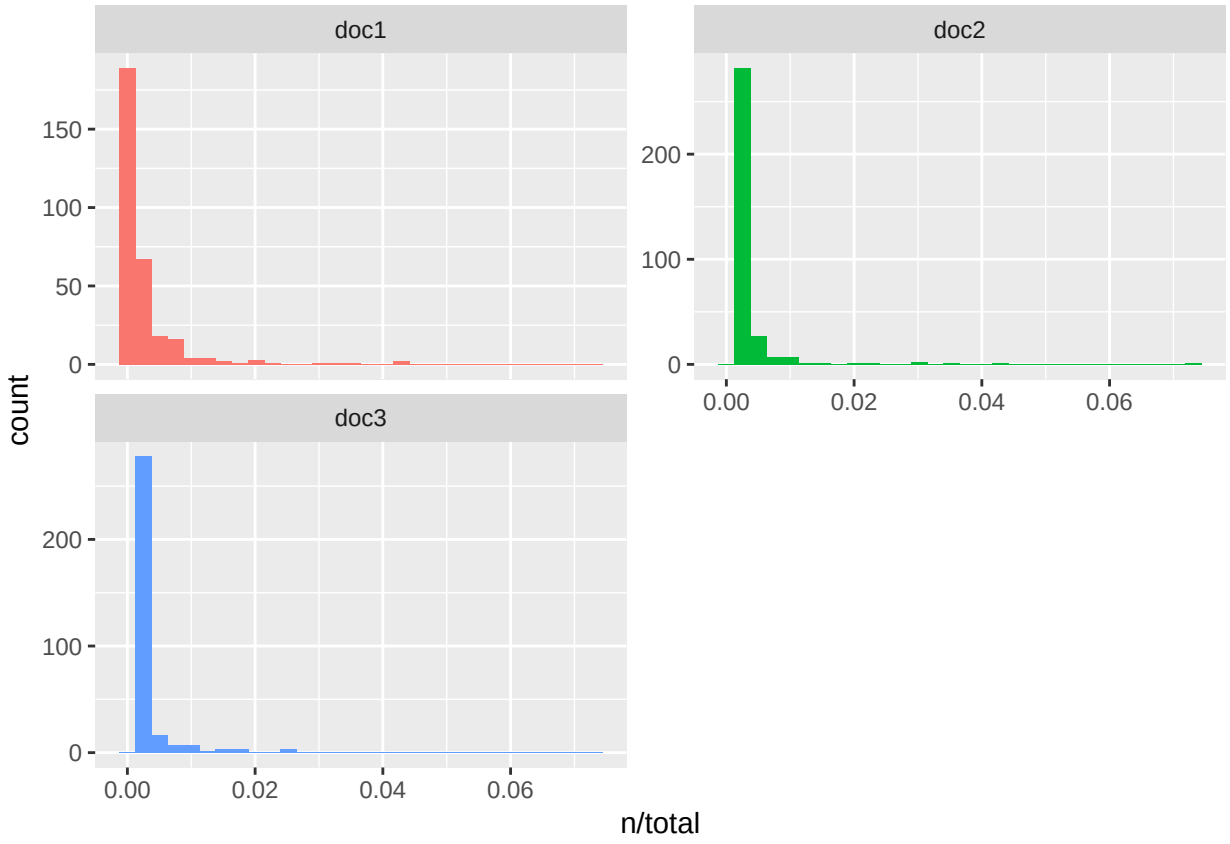
Wordclouds


misconceptions
notes
freq
difference
htmltab
observations
create
50
block
block_to_add
hash
question
start
0
surprising
observed
figure
specific
index
previous_block
blocksfunction
pow
generator
effect
significant
wikitab
und
tells
blog
e4
values
assuming
add
chain
code
means
blockchains
observing
birthday
bday
previous
0.25
study
sample
eng
d4
0.75
sine
type
test
2018
sound
daniel
wave
correct
4
0.5
assume
jones
list
differences
tail
g4
result
fallpower
expect
added
twitter
rate
genesis
alternative
time
omniscient
proof
für
1
music
probability
null
pitch
model
5
random
level
observe
understand
2
timestamp
extreme
created
distribution
hypothesis
size
post
bitcoin
data
true
blockchain

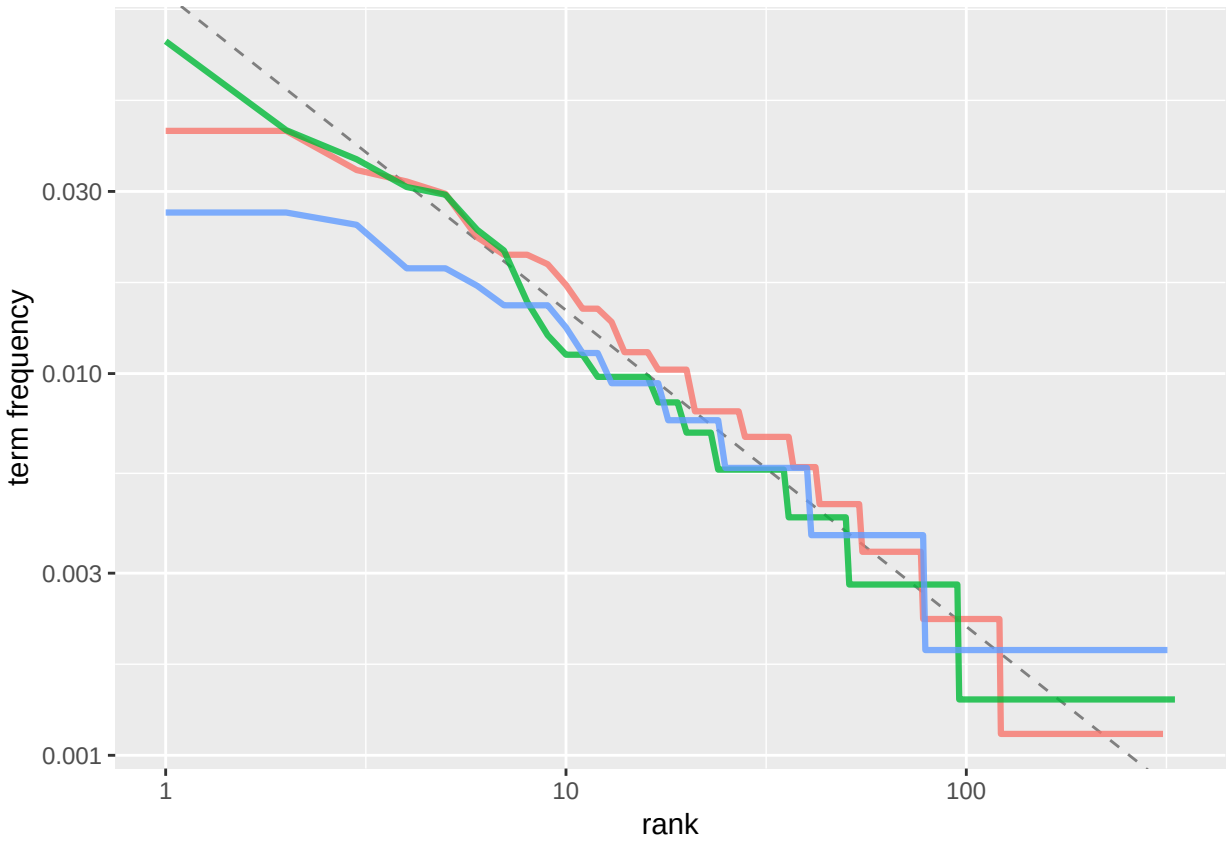
negative

positive

Chapter 3 tf-idf

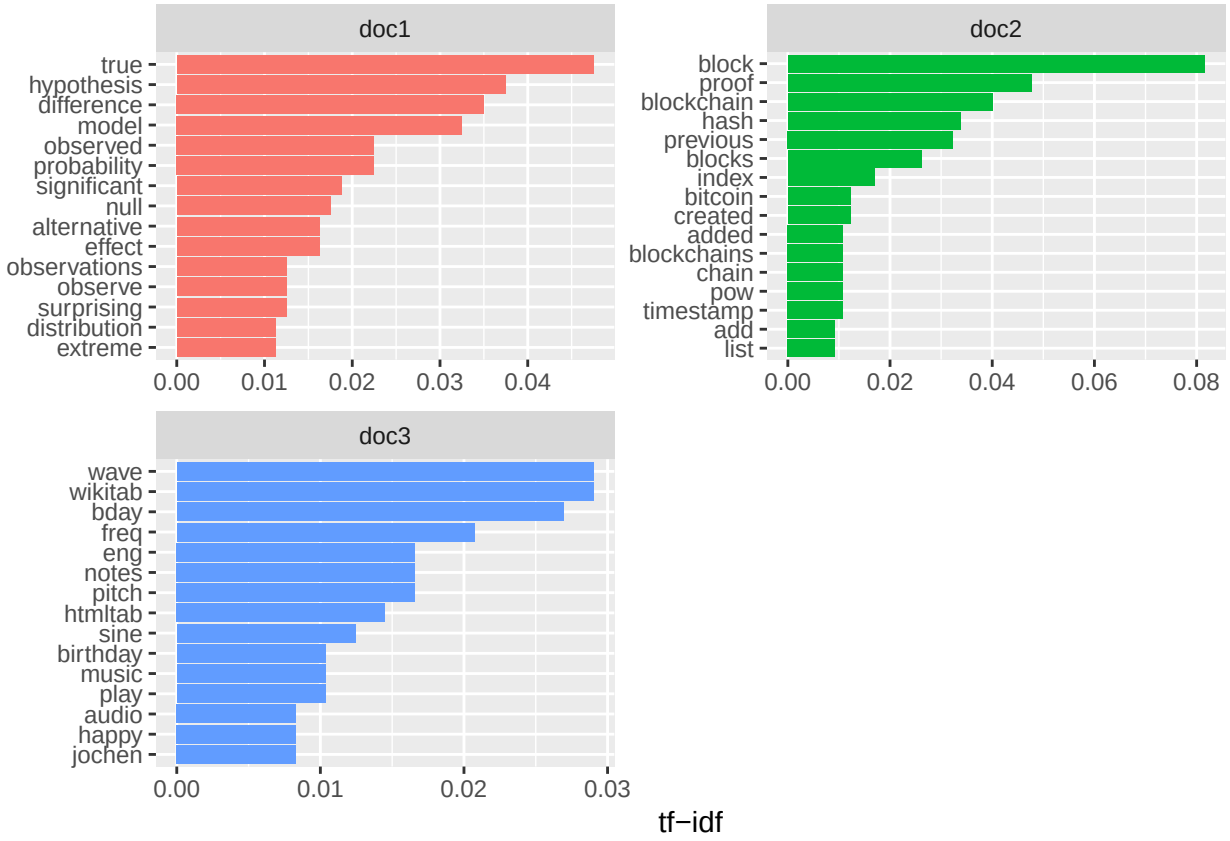



```
## (Intercept) log10(rank)
##  -1.0057750  -0.8284333
```



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
```



Chapter 4 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
```

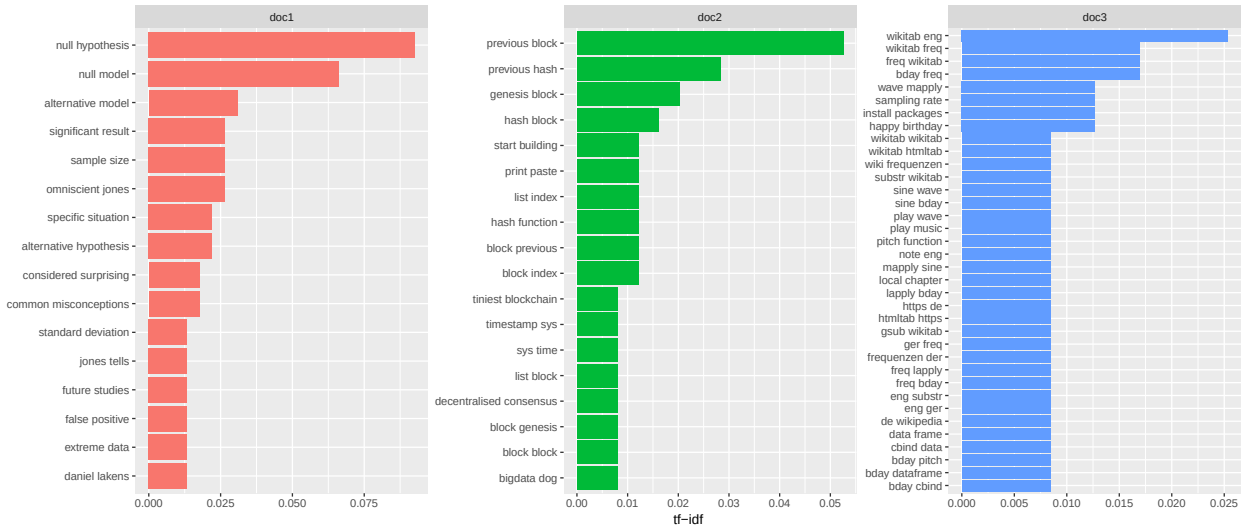


Figure: The 12 bigrams with the highest tf-idf

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 9e0b0fb DN-- 82 62 --
## + attr: name (v/c), n (e/n)
## + edges from 9e0b0fb (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
```