Text Analysis in R

Mandy Bowers

April 27, 2018

Text Analysis in R

What is text analysis?

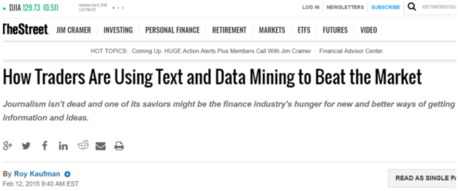
How is text analysis currently being used?

Text Analysis in R

What is text analysis?

- Text analysis allows you to extract key information from unstructured text and organize it into a usable way for analysis.

How is text analysis currently being used?



Text analysis and the Fed

BBVA | RESEARCH

U.S. Economic Watch
27 September 2016

ECONOMIC ANALYSIS

Interpreting the Fedspeak: text analysis on FOMC statements

Kan Chen

- **Sentiments in FOMC statements can be used to analyze monetary policymaking**
- **The peak of the uncertain sentiment precedes the Great Recession, making it a potential indicator for the worst scenario**
- **Sentiments in the statements are significantly correlated with VIX and yield spread**

Sentiment Analysis Of FOMC Statements Reveals A More Hawkish Fed

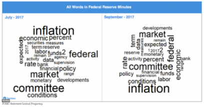
Alap Shah, CONTRIBUTOR
FULL BIO

Federal Reserve Board Chairwoman Janet Yellen speaks during a news [+]

The Federal Reserve System's Federal Open Market Committee (FOMC) meets eight times a year, at 2 p.m. Eastern Time in the basement of a nondescript, Washington, D.C. office building. The terse statements released after those meetings drive the direction of global financial markets and the meeting minutes are carefully scrutinized carefully by the media.

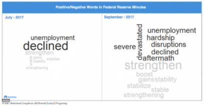
We parsed recent statements and minutes since 2012 using Sentieo's natural language processing and sentiment analysis and found some interesting trends.

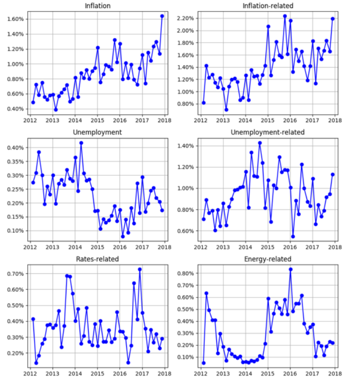
For the most recent statement 9/20, the strongest topic continued to be inflation, as highlighted in the unfiltered word cloud shown here.



FOMC Statement Lexical Frequency

The intensity was roughly equivalent to the prior statement, as the Fed continues to be vexed by an inflation shortfall versus expectations. Based on the statements alone, this analysis would suggest that Fed intentions have barely changed. However, when we apply sentiment analysis to the words in the documents using the Loughran-McDonald context-specific lexicon, which assigns a simple positive or negative value to words based on the financial services industry context, the 9/20 statement occurs as much more hawkish.





Introduction to the Dataset

- Today we'll be working with FOMC meeting minutes.
  - Source: https://www.federalreserve.gov/monetarypolicy/fomc_historical_year.htm
- What is the FOMC? Why is it important?
- First we'll work with a very limited sample to get a feel for text analysis before we take on a lot of data:
  - Feb 1-2, 2005
  - Jan 27-28, 2009

Looking at our data: The Corpus

Using R's ‘tm’(text mining) package to open our ‘corpus’.

```
docs <- tm::VCorpus(DirSource("Sources/FOMC_ex1"))
class(docs)
## [1] "VCorpus" "Corpus"
length(docs)
## [1] 2
docs[1]
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 1
docs[1][[1]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 328776
length(docs[1][[1]]$content)
## [1] 708
docs[1][[1]]$meta
##   author       : character(0)
##   datetimestamp: 2018-03-23 19:54:27
##   description  : character(0)
##   heading      : character(0)
##   id           : FOMC20050202meeting.txt
##   language     : en
##   origin       : character(0)
```

Looking at our data: The Text

What does it look like inside our corpus? Let's find out.

```
## docs[1][[1]]$content
docs[1][[1]]$content[1]
```

[1] "Meeting of the Federal Open Market Committee on"

```
docs[1][[1]]$content[2]
```

[1] "February 1-2, 2005"

```
docs[1][[1]]$content[4]
```

[1] "A meeting of the Federal Open Market Committee was held in the offices of the Board of Governors of the Federal Reserve System in Washington, D.C., at 1:30 p.m. on Tuesday, February 1, 2005, and continued at 9:00 a.m. on Wednesday, February 2, 2005. Those present were the following:"

```
docs[1][[1]]$content[705]
```

[1] "CHAIRMAN GREENSPAN. I wish to announce that in record time the Federal Reserve Board acted expeditiously to change the discount rate, and you all know the direction and the amount. As a consequence, I now adjourn this meeting and suggest that we go to lunch."

```
docs[1][[1]]$content[706]
```

Preparing the Text: Question

- We need to make our text more machine-friendly. What do you think we need to do to this text to make it machine readable?

Preparing the Text: Answer

- We need to make our text more machine-friendly. What do you think we need to do to this text to make it machine readable?
  - Remove punctuation
  - Remove case
  - Remove numbers
  - Remove white space
  - Remove ‘useless’ words == **stopwords**
  - Remove useless endings on words (“ing” “s” etc.) = **stemming words**

Preparing the Text: the TM package

Fortunately, we have one package prepared to help us take care of all of these things

Text cleaning task	TM package solution
Remove punctuation	removePunctuation
Make lowercase	content_transformer(tolower)
Remove numbers	removeNumbers
Remove stopwords	removeWords, stopwords(“english”)
Stem words	stemDocument
Remove white space	stripWhitespace

These are used as arguments in tm_map(*corpus to edit*, *option*)

Preparing the Text: using the TM package to clean text (Example)

```
docs[1][[1]]$content[4]
```

[1] "A meeting of the Federal Open Market Committee was held in the offices of the Board of Governors of the Federal Reserve System in Washington, D.C., at 1:30 p.m. on Tuesday, February 1, 2005, and continued at 9:00 a.m. on Wednesday, February 2, 2005. Those present were the following:"

```
docs1 <- tm_map(docs, removePunctuation)
docs1[1][[1]]$content[4]
```

[1] "A meeting of the Federal Open Market Committee was held in the offices of the Board of Governors of the Federal Reserve System in Washington DC at 130 pm on Tuesday February 1 2005 and continued at 900 am on Wednesday February 2 2005 Those present were the following"

Preparing the Text: Cleaning Text (Exercise)

```
## Transform the capitalization to lowercase
docs2 <- tm_map(docs1,                        )
#View the content to make sure it worked!
docs2[1][[1]]$content

## Remove the numbers from the documents
docs3 <- tm_map(       ,                )
docs3[1][[1]]$

## Remove stopwords from the documents
docs_cleaned <- tm_map(       ,         ,            )
docs_cleaned[ ][[ ]]$
```

Preparing the Text: Cleaning Text (Answer)

```
## Transform the capitalization to
## lowercase
docs2 <- tm_map(docs1, content_transformer(tolower))
# View the content to make sure it
# worked!
docs2[1][[1]]$content[4]
```

[1] "a meeting of the federal open market committee was held in the offices of the board of governors of the federal reserve system in washington dc at 130 pm on tuesday february 1 2005 and continued at 900 am on wednesday february 2 2005 those present were the following"

```
## Remove the numbers from the documents
docs3 <- tm_map(docs2, removeNumbers)
docs3[1][[1]]$content[4]
```

[1] "a meeting of the federal open market committee was held in the offices of the board of governors of the federal reserve system in washington dc at pm on tuesday february and continued at am on wednesday february those present were the following"

```
## Remove stopwords from the documents
docs_cleaned <- tm_map(docs3, removeWords,
    stopwords("english"))
docs_cleaned[1][[1]]$content[4]
```

Preparing the Text: Finishing up our text cleaning

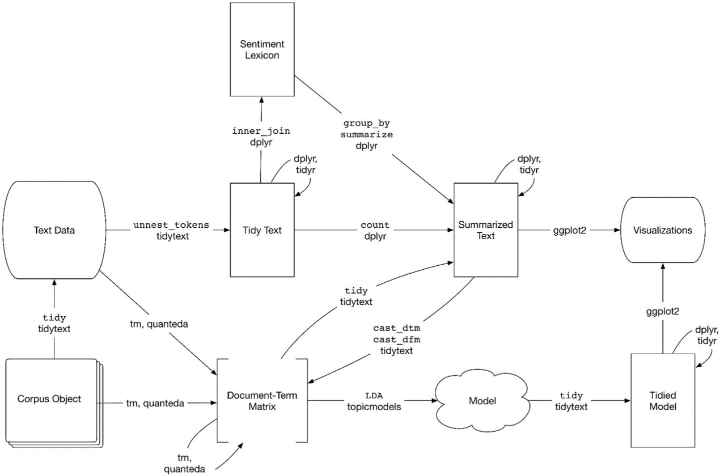
- What does stemming do? Why is this useful?

```
docs_stemmed <- tm_map(docs_cleaned, stemDocument)
docs_stemmed[1][[1]]$content[4]
```

[1] "meet feder open market committe held offic board governor feder reserv system washington dc pm tuesday februari continu wednesday februari present follow"

Preparing the Text: Tidy Text and the Tidyverse

- We can use the tidyverse to manipulate text more easily
- What names of packages do you recognize here?



Putting our Corpus into the Tidyverse

- How can we ‘tidy’ our corpus? With ‘tidy’!

```
tidy_docs <- tidy(docs_stemmed)

# What class is tidy_docs?
class(tidy_docs)

tidy_docs
tidy_docs$text
```

Unnest_tokens

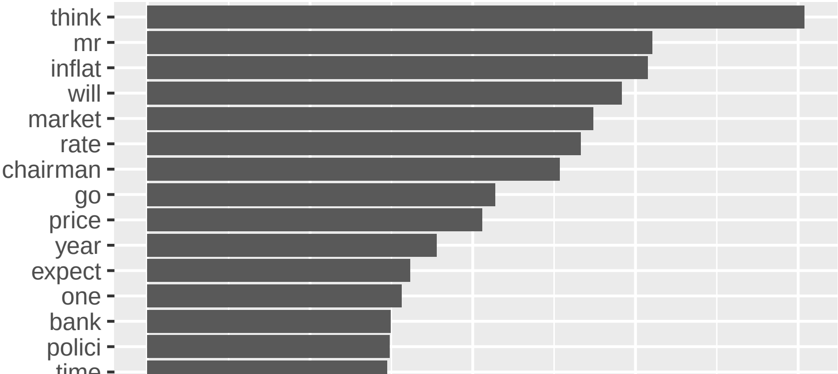
- The power of the tidyverse is harnessed by splitting each word into one datapoint
- We can split a document using `unnest_tokens`

```
tidy_words <- tidy_docs %>% unnest_tokens(word,
    text)

# let's view our words (scroll across to
# the word column)
head(tidy_words, 10)
tail(tidy_words, 10)
```

Graphing with Tidytext

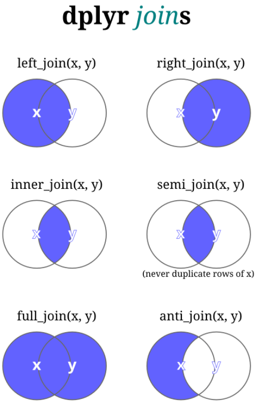
- Let's use the tidyverse (our old friend ggplot) to analyze our newly tidy text

```
word_freq <- tidy_words %>% dplyr::count(word,
    sort = TRUE) %>% filter(n > 50) %>% mutate(word = reorder(word,
    n)) %>% head(15) %>% ggplot(aes(word,
    n)) + geom_col() + xlab(NULL) + coord_flip()
word_freq
```



Removing Unwanted Words: Revisiting Joins

- Let's get rid of some of these less interesting words!
- I have a list of unwanted words. Which join could I use to take them out of my `tidy_words`?



Removing Unwanted Words: Creating our ‘Y’

- `tidy_words` will be our X, and we want to remove a list of words Y
- Let’s first create our Y:

```
remove_words <- c("mr", "go", "one", "us",
    "like", "will", "can", "just", "also",
    "now", "chairman", "thank", "vice")
rm_words_table <- data.frame(remove_words)
colnames(rm_words_table) <- c("word")
```

Removing Unwanted Words: Using anti_join

- `tidy_words` will be our X, and we want to remove a list of words Y
- Let's first create our Y:

```
interesting_tidy_words <- tidy_words %>%
    anti_join(rm_words_table)
## Joining, by = "word"
## Warning: Column `word` joining character vector and factor, coercing into
## character vector
```

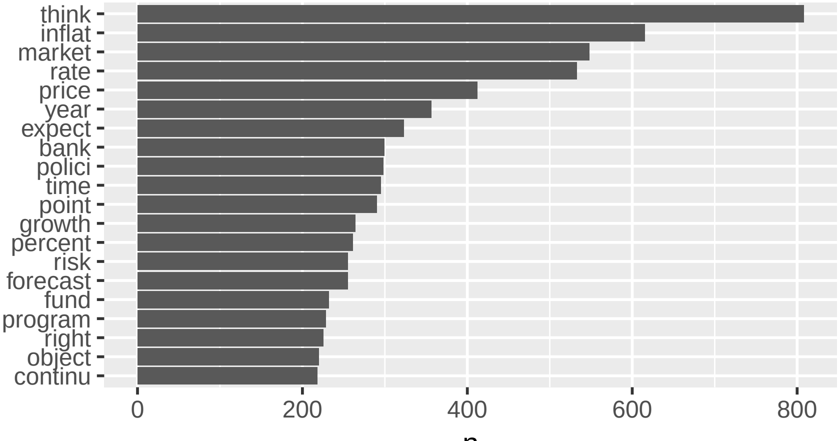
Graphing with Tidytext: Exercise

- Fill in the graph code below based on our previous word frequency graph code

```
interesting_word_freq <-                          %>%
  count(     , sort = TRUE) %>%
  filter(   > 50) %>%
         (word = reorder(word, n)) %>%
  head(   ) %>%
  ggplot(aes(word,   )) +
  geom_col() +
  xlab(NULL) + coord_flip()
interesting_word_freq
```

Graphing with Tidytext: Exercise (Answer)

```
interesting_word_freq <- interesting_tidy_words %>%
    count(word, sort = TRUE) %>% filter(n >
    50) %>% mutate(word = reorder(word, n)) %>%
    head(20) %>% ggplot(aes(word, n)) + geom_col() +
    xlab(NULL) + coord_flip()
interesting_word_freq
```



Comparing Articles' Word Frequency

- So far we've been looking at our two different articles together. How can we compare how words changed between 2005 and 2009?
- Let's remind ourselves what interesting_tidy_wods has in it.

```
colnames(interesting_tidy_words)
```

[1] "author" "datetimestamp" "description" "heading"
[5] "id" "language" "origin" "word"

- Which one of these will allow us to distinguish between article dates?

Manipulating the ID: Case When

- The current IDs we have are long and complicated – let's switch the names using `case_when`

```
words_0509 <- interesting_tidy_words %>%
    select(word, id) %>% mutate(meeting = case_when(id ==
    "FOMC20050202meeting.txt" ~ "fomc2005",
    id == "FOMC20090128meeting.txt" ~ "fomc2009")) %>%
    select(-id)
```

Graphing Articles' Word Frequency: Preparing the Data

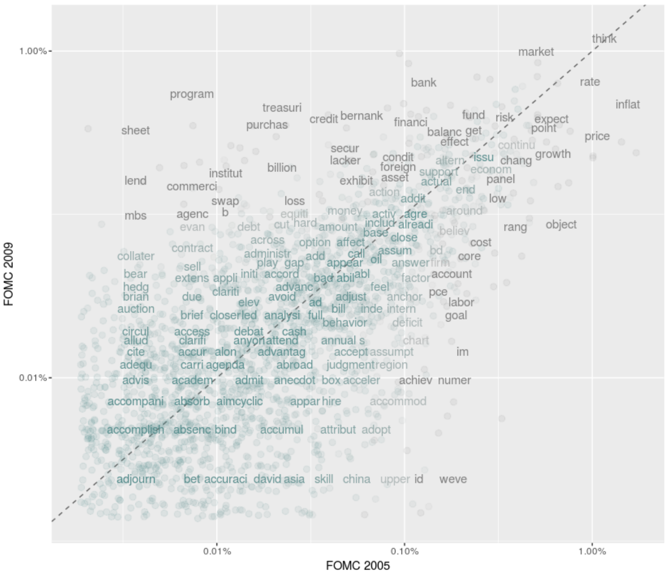
```
frequency <- words_0509 %>%
  count(meeting, word) %>% #create col 'n' - times per meeting a word appears
  group_by(meeting) %>% #tells mutate we will sum on 'meeting'
  mutate(proportion = n / sum(n)) %>% #calculate a column w times
  # a word appears in the minutes divided by the total words in given meeting
  select(-n) %>% #remove extraneous column
  spread(meeting, proportion) #reorganizes data around 'word'

head(frequency,5)
## # A tibble: 5 x 3
##      word     fomc2005     fomc2009
##     <chr>        <dbl>        <dbl>
## 1     aaa           NA 1.649666e-04
## 2  aaarat           NA 1.374722e-04
## 3      ab           NA 3.024388e-04
## 4 abandon 3.702881e-05           NA
## 5    abat           NA 2.749443e-05
```

Graphing Articles' Word Frequency: Creating the Graph

- What will this graph look like?

```
ggplot(frequency, aes(x = fomc2005, y = fomc2009,
    color = abs(fomc2005 - fomc2009))) +
    geom_abline(color = "gray40", lty = 2) +
    geom_jitter(alpha = 0.1, size = 2.5,
        width = 0.3, height = 0.3) + geom_text(aes(label = word),
    check_overlap = TRUE, vjust = 1.5) +
    scale_x_log10(labels = percent_format()) +
    scale_y_log10(labels = percent_format()) +
    scale_color_gradient(limits = c(0, 0.001),
        low = "darkslategray4", high = "gray75") +
    theme(legend.position = "none") + labs(y = "FOMC 2009",
    x = "FOMC 2005")
```

Graphing Articles' Word Frequency: Creating the Graph



Word Clouds: Introduction

- That graph was interesting, but it was a little hard to read. Besides, we're most interested in what words are appearing when. . . introducting wordclouds!
- Let's create a word cloud for 2005 together, and then you can make one for the 2009 words

```
word_cloud_05 <- words_0509 %>% filter(meeting ==
    "fomc2005") %>% count(word, sort = TRUE) %>%
    with(wordcloud(word, n, max.words = 80))
```

Word Clouds: Output

Word Clouds: Exercise

- Now use the template below to create a wordcloud for 2009! (it's ok if R gets mad at you because things can't fit on the screen)

```
word_cloud_09 <-              %>%
  filter(                  ) %>%
  count(word,       = TRUE) %>%
  with(wordcloud(         ,    , max.words =     ))
```

Comparing Word Clouds

- Let's compare the most common words in the early 2005/2009 FOMC meetings

Sentiment Analysis: Introduction

- Another popular method of text analysis is opinion mining/sentiment analysis. When we read text we understand whether it is positive/negative (plus additional emotions). How can we make a computer understand that?
- A common approach to sentiment analysis that makes use of the tidyverse is to measure the 'sentiment' of each individual word and combine all the words of the texts to create a score for the overall text or portions of the text.
- To this end, the tidytext package contains several sentiment lexicons in the `sentiments` database. Note that these datasets were generally constructed via crowdsourcing or present-day individuals.

Sentiment Analysis: Bing

- Today, we'll just look at one, `bing`. Explore `bing` with

```
get_sentiments("bing")
```

```
## # A tibble: 6,788 x 2
##           word sentiment
##          <chr>     <chr>
##  1     2-faced  negative
##  2     2-faces  negative
##  3          a+  positive
##  4    abnormal  negative
##  5     abolish  negative
##  6  abominable  negative
##  7  abominably  negative
##  8   abominate  negative
##  9 abomination  negative
## 10       abort  negative
## # ... with 6,778 more rows
```

* How does bing categorize words? * Can you foresee any problems with using a dataset like this to analyze text?

Sentiment Analysis: Drivers

- Let's look at what words will be driving the sentiment of the 2009 meeting.

```
sentiment_09 <- words_0509 %>% filter(meeting ==
    "fomc2009") %>% inner_join(get_sentiments("bing")) %>%
    count(word, sentiment, sort = TRUE) %>%
    ungroup()
```

Sentiment Analysis: Drivers (Answer)

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

## # A tibble: 288 x 3
##       word sentiment     n
##      <chr>     <chr> <int>
##  1    risk  negative   163
##  2   right  positive   154
##  3    well  positive   110
##  4 concern  negative    80
##  5 support  positive    76
##  6    good  positive    64
##  7    work  positive    62
##  8 problem  negative    61
##  9    loss  negative    50
## 10   sever  negative    46
## # ... with 278 more rows
```

Importing our Second Corpus

- A more interesting question is how does the sentiment of these meetings change over time. For that we'll need more than just two meetings.
- Let's create a larger dataset

```
fomc <- tm::VCorpus(DirSource("Sources/FOMCminutes"))
class(fomc)
```

```
## [1] "VCorpus" "Corpus"
```

```
length(fomc)
```

```
## [1] 40
```

- We have to clean these documents up too! What's a way we can expedite this for the future?

Writing Functions to Clean Text

- What do these functions do?

```
id_tidy <- function(corpus) {
    for (i in 1:nrow(corpus)) {
        temp <- str_extract_all(corpus$id[i],
            "\\(?[0-9]+\\)?")
        temp <- temp[[1]][1]
        temp2a <- substring(temp, 1, 4)
        temp2b <- substring(temp, 5, 6)
        temp3 <- paste(temp2a, temp2b, sep = "-")
        corpus$id[i] <- temp3
    }
    return(corpus)
}

prep_text <- function(corpus) {
    corpus <- corpus %>% tm_map(removeNumbers) %>%
        tm_map(removePunctuation) %>% tm_map(content_transformer(tolower)) %>%
        tm_map(removeWords, stopwords("english")) %>%
        tm_map(removeWords, remove_words) %>%
        tm_map(stemDocument) %>% tidy() %>%
        id_tidy() %>% select(id, text)
}
```

Running our Custom Functions

- Run these functions to get our new data ready to use

```
tidy_fomc <- prep_text(fomc)
tidy_fomc
```

```
## # A tibble: 40 x 2
##          id
##       <chr>
##  1 2005-02
##  2 2005-03
##  3 2005-05
##  4 2005-06
##  5 2005-08
##  6 2005-09
##  7 2005-11
##  8 2005-12
##  9 2006-01
## 10 2006-03
```

Looking at the Change in Sentiment over Time (Exercise)

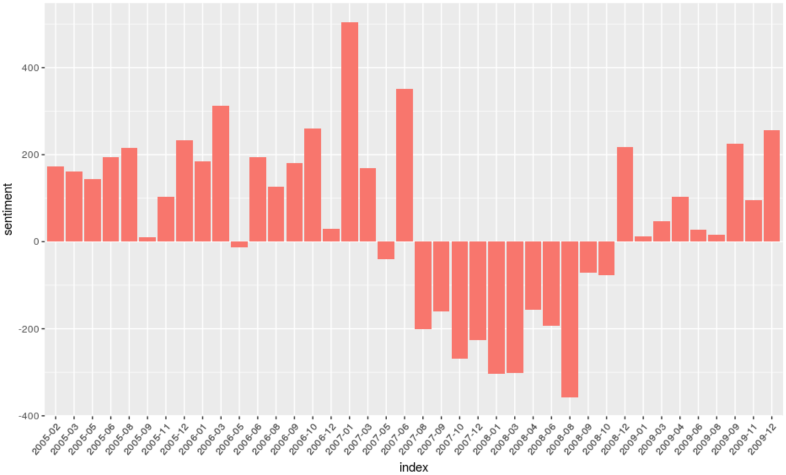
- Fill out the code below to create a chart tracking change over time in sentiment of FOMC meeting minutes.

```
tidy_fomc_words <- tidy_fomc %>% unnest_tokens(,
    text)
fomc_sentiment <- tidy_fomc_words %>% (get_sentiments()) %>%
    count(index = id, sentiment) %>% spread(sentiment,
    n, fill = 0) %>% mutate(sentiment = positive -
    negative)

ggplot(, aes(index, sentiment, fill = "red")) +
    geom_col(show.legend = FALSE) + theme(axis.text.x = element_text(angle = ,
    hjust = 1, vjust = 1))
```

Looking at the Change in Sentiment over Time (Answer)

```
tidy_fomc_words <- tidy_fomc %>% unnest_tokens(word,
    text)
fomc_sentiment <- tidy_fomc_words %>% inner_join(get_sentiments("bing")) %>%
    count(index = id, sentiment) %>% spread(sentiment,
    n, fill = 0) %>% mutate(sentiment = positive -
    negative)

ggplot(fomc_sentiment, aes(index, sentiment,
    fill = "red")) + geom_col(show.legend = FALSE) +
    theme(axis.text.x = element_text(angle = 50,
        hjust = 1, vjust = 1))
```

Looking at the Change in Sentiment over Time (Image)



Words over Time: Data Prep (Exercise)

- Now we're going to look at how the frequency of individual words changed over the period
- We are going to look at 6 words very important to the Fed and the Great Recession: inflation, unemployment, rate, mortgage, bank, risk
- First we need a summary word counts by FOMC meeting
  - Hint: Look at `tidy_fomc` in the viewer to learn what the column names are

```
fomc_words_count <- tidy_fomc %>% unnest_tokens(,
    ) %>% group_by() %>% count(word)
```

Words over Time: Data Prep (Answer)

```
fomc_words_count <- tidy_fomc %>% unnest_tokens(word,
    text) %>% group_by(id) %>% count(word)
```

Words over Time: Create Yearly Totals

- I don't want to have to look a bajillion meetings. I just want to look at a summary for the year. Let's consolidate our FOMC meetings by year.

```
fomc_year_term_counts <- fomc_words_count %>%
    # extracts just year from the meeting id
tidyr::extract(id, "year", "(\\d+)", convert = TRUE) %>%
    group_by(year, word) %>% # collapses all appearances of
# a given year
summarize(word_total_by_year = sum(n)) %>%
    group_by(year) %>% # provides column of total word coun
# year
mutate(year_total = sum(word_total_by_year))
```

Words over Time: Let's look at our data

```
fomc_year_term_counts %>% filter(word ==
    "rate")
```

```
## # A tibble: 5 x 4
## # Groups:   year [5]
##    year  word word_total_by_year year_total
##   <int> <chr>              <int>      <int>
## 1  2005  rate               1535     163468
## 2  2006  rate               1611     187777
## 3  2007  rate               1686     222251
## 4  2008  rate               2704     234965
## 5  2009  rate               2263     260906
```

- How many times does the word 'bank' appear in 2008?
- How many times does the word 'risk' appear in 2007?
- How many times does the word 'inflation' appear in 2006?

Tracking Individual Words over Time: Exercise

- Now, use this code to create a six facet graph tracking the changes in inflation, umeployment, rate, risk, mortgage, and bank from 2005 to 2009.
- Hints:
  - `%in%` will help as such: 'category' `%in%` c("items you want from category")
  - We want to plot year against a word's appearance rate, where a word's appearance rate is the word's mentions per year divided by the total number of words said in that year

```
fomc_year_term_counts %>%
  filter(      %in% c("inflat",       ,       ,       , "mortgag",      )) %>%
  ggplot(aes(     ,          /        )) +
  geom_point() +
  geom_smooth() +
  facet_wrap(~      , scales = "free_y") +
  scale_y_continuous(labels = scales::percent_format()) +
  ylab("% Frequency of word in FOMC minutes")+
  theme(strip.text.x = element_text(size =   ))
```

Tracking Individual Words over Time: Answer

```
fomc_year_term_counts %>% filter(word %in%
    c("inflat", "unemploy", "rate", "risk",
        "mortgag", "bank")) %>% ggplot(aes(year,
    word_total_by_year/year_total)) + geom_point() +
    geom_smooth() + facet_wrap(~word, scales = "free_y") +
    scale_y_continuous(labels = scales::percent_format()) +
    ylab("% frequency of word in FOMC minutes") +
    theme(strip.text.x = element_text(size = 15))
```

Tracking Individual Words over Time: Results

- Congratulations, you can now quantify some of the evolution in the Fed's monetary policy thinking!