Minutes Analysis

Connor Krenzer

1/4/2021

Introduction

The Federal Reserve is the central bank of the United States. Central banks conduct a country's monetary policy (policy pertaining to money–interest rates, regulation of the banking sector, etc.) in the aims of achieving it's 'dual mandate' of full employment (loosely defined as "anyone who wants a job can get a job") and price stability (predictable price changes throughout the United States as a whole). You can read more about what the Fed is and what they do on their website. For all of you Fed haters out there, thinking that I'm misleading people by linking to the Fed's website, do not disparage! Perhaps one of these articles is more your style.

Every six weeks, policy makers on the Federal Reserve's Federal Open Market Committee (FOMC) meet to discuss economic indicators to determine and vote on policy in response to these indicators, among other things. The FOMC Statement released following this meeting can send the entire economy into turmoil–so much so that the minutes are withheld from the public for three weeks before becoming available.

The data in this analysis includes all FOMC Minutes between January 28th, 2009 to November 5th, 2020 (inclusive). As of the time of writing, the Minutes for the December 15th-16th meeting have not been released.

I will explore these texts using various text mining approaches including sentiment analysis, tf-idf, and topic modeling to identify trends within the FOMC's Meetings.

Packages

I recommend pulling the information using the HTML version of these meetings. The PDFs are a nightmare to clean (speaking from experience)! If you do decide to read the pdf text into R, I suggest using the extract_text() function from the tabulizer package because it is smart enough to recognize columns of selectable PDF text. Other popular packages for reading in PDFs, such as readtext or pdftools, will only read across the page left to right, then down–despite the pages containing two separate columns of text.

```
# The names of all your installed packages
package_names <- rownames(installed.packages())

# Installing packages used in this project
if(!"dplyr" %in% package_names) install.packages("dplyr")
if(!"stringr" %in% package_names) install.packages("stringr")
if(!"tidytext" %in% package_names) install.packages("tidytext")
if(!"rvest" %in% package_names) install.packages("rvest")
if(!"tidyr" %in% package_names) install.packages("tidyr")
if(!"ggplot2" %in% package_names) install.packages("ggplot2")
```

```r
if(!"patchwork" %in% package_names) install.packages("patchwork")
if(!"lubridate" %in% package_names) install.packages("lubridate")
if(!"knitr" %in% package_names) install.packages("knitr")
if(!"igraph" %in% package_names) install.packages("igraph")
if(!"ggraph" %in% package_names) install.packages("ggraph")
if(!"grid" %in% package_names) install.packages("grid")
if(!"widyr" %in% package_names) install.packages("widyr")
if(!"topicmodels" %in% package_names) install.packages("topicmodels")

# Loading in dplyr because it is used too often
# to :: it the entire time
library(dplyr)

# Packages which are difficult to use without loading:
library(rvest)
library(ggplot2)
library(patchwork)


rm(package_names)
```

Data Import

Since I started this process using the pdf files, I will use their file names to determine the corresponding url. If I had started mining this data from HTML in the first place, a more elegant solution might have been used, but downloading all the PDF files and using their names to put together urls works fine enough...

There are two formats for the HTML file–one for the Fed Minutes before 2012 and those released on or after that year. With this in mind, two functions will be used to read in the different formats. The documents are parsed in chronological order but are also arranged after being added to the tibble, in the event that something were to happen to the order of the files in the "Fed Minutes Releases" folder from which these names were pulled.

```r
# The code in this section "makes" the dataset
old_format <- function(id){

  # Note: using str_c() to prevent the code from going out of
  # bounds in the output (so I don't have to get into dark LaTex magic!)
  name <- stringr::str_c("https://www.federalreserve.gov/monetarypolicy/",
                         id,
                         ".htm")

  tibble(doc_id = id, text = read_html(name) %>%
           html_nodes("#leftText") %>%
           html_text() %>%
           stringr::str_split("\n", simplify = T) %>%
           as.vector() %>%
           stringr::str_to_lower() %>%
           stringr::str_replace_all("'s|\"|,|:|;|-{2,}", " ") %>%
           magrittr::extract(!stringr::str_detect(., "_+")) %>%  # removing the signature
           stringr::str_c(., collapse = " ") %>%
           stringr::str_replace_all("mr\\.{1}|mrs\\.{1}|ms\\.{1}", " ") %>%
           stringr::str_replace_all("messrs[\\.]?|mmes[\\.]?", " ") %>%
```

```
          stringr::str_replace_all("mss[\\.]?|mses[\\.]?", " ") %>%
          stringr::str_replace_all("a\\.m\\.|p\\.m\\.|d\\.c\\.", " ") %>%
          stringr::str_replace_all("\\d", " ") %>%
          stringr::str_squish() %>%
          stringr::str_remove_all("return to text[\\d]*|return to top") %>%
          stringr::str_trim()
  ) %>%
    return()

}#end of old_format()


# For releases 2012 and onward:
# We can always assume the first 37 entries are bogus
# Keep everything following the 37th entry
new_format <- function(id){
  name <- stringr::str_c("https://www.federalreserve.gov/monetarypolicy/",
                         id,
                         ".htm")

  tibble(doc_id = id, text = read_html(name) %>%
          html_nodes("p") %>%
          html_text() %>%
          stringr::str_trim() %>%
          stringr::str_squish() %>%
          magrittr::extract(38:length(.)) %>%
          magrittr::extract(!stringr::str_detect(., "_+")) %>%  # removing the signature
          stringr::str_to_lower() %>%
          stringr::str_replace_all("'s|\"|,|:|;|-{2,}", " ") %>%
          stringr::str_c(., collapse = " ") %>%
          stringr::str_replace_all("mr\\.{1}|mrs\\.{1}|ms\\.{1}", " ") %>%
          stringr::str_replace_all("messrs[\\.]?|mmes[\\.]?", " ") %>%
          stringr::str_replace_all("mss[\\.]?|mses[\\.]?", " ") %>%
          stringr::str_replace_all("a\\.m\\.|p\\.m\\.|d\\.c\\.", " ") %>%
          stringr::str_replace_all("\\d", " ") %>%
          stringr::str_squish() %>%
          stringr::str_replace_all("return to text[\\d]*|return to top", " ") %>%
          stringr::str_trim()
  ) %>%
    return()

}#end of new_format()




# The paths to the PDF files
dirs <- c("Fed Minutes Releases/Obama Era/Pre 2012/",
          "Fed Minutes Releases/Obama Era/Post 2012/",
          "Fed Minutes Releases/Trump Era/")
```

```r
# An empty tibble--document names and their corresponding
# text will be added to it.
documents <- tibble::tibble(doc_id = NULL, text = NULL)


# I'm not crazy about for loops, but I think this code is
# straightforward enough to follow
# Reading in the old formats (pre-2012)
for(i in stringr::str_remove_all(list.files(dirs[1]), "\\.pdf")){
  documents <- bind_rows(documents, old_format(i))

  # simple yet effective way of showing the operation's progress
  cat(".")
}

# Reading in the new formats (2012-present)
for(i in stringr::str_remove_all(list.files(dirs[2:3]), "\\.pdf")){
  documents <- bind_rows(documents, new_format(i))

  # simple yet effective way of showing the operation's progress
  cat(".")
}

# Arranging the documents in chronological order
documents <- arrange(documents, doc_id)
```

The Dataset

The data is almost ready for analysis. We now have two columns: one for the document name, and another for the contents of that document. If you are interested in performing a similar analysis, the data frame can now be transformed in a number of ways. You can use the unnest_tokens() function from the tidytext package to extract tokens (words, groups of words, paragraphs, etc.) and perform sentiment analysis, or you could make a term-document matrix and perform topic modeling.

We will do both, but let's begin by taking a look at the dataset:

> s of dollars equivalent)bank of canada bank of mexico any changes in the terms of existing swap arrangements and the proposed terms of any new arrangements that may be authorized shall be referred for review and approval to the committee. . all transactions in foreign currencies undertaken under paragraph .a. abo

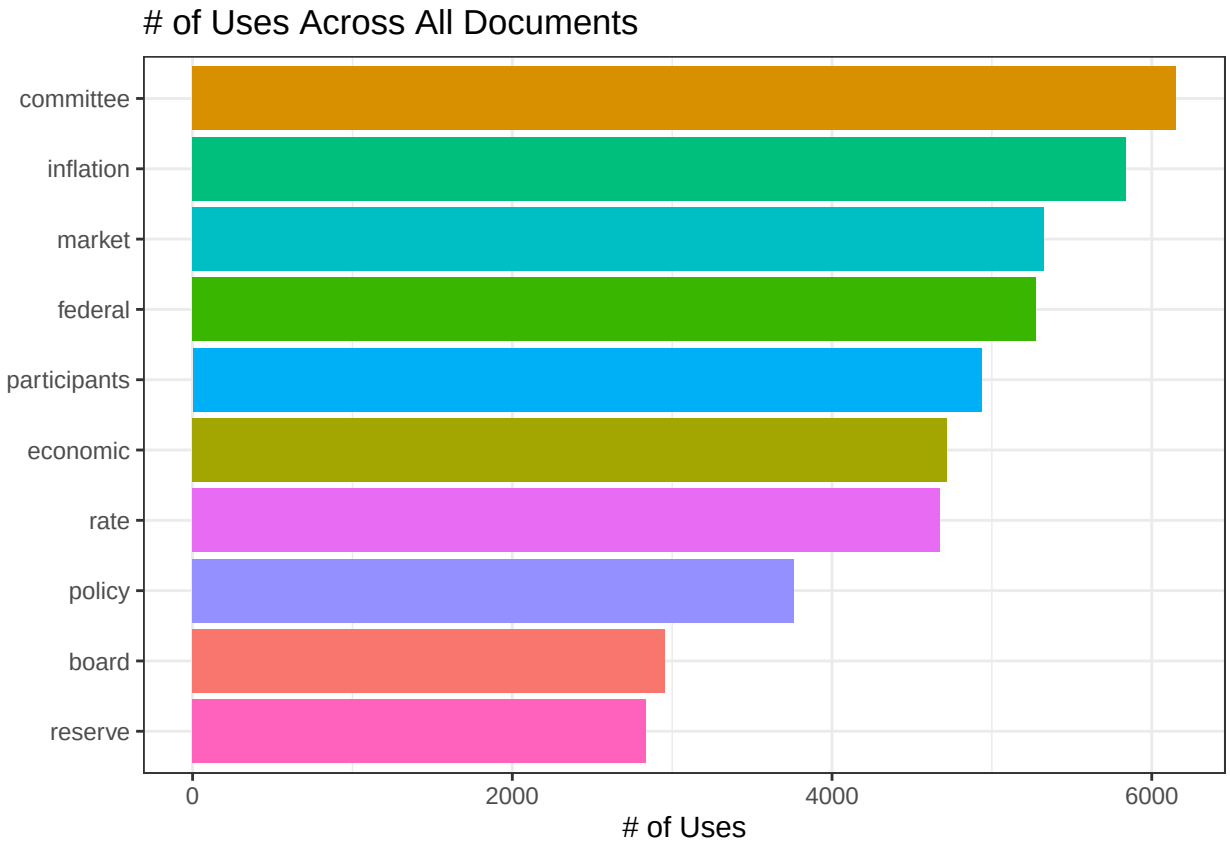
The quote above is an excerpt from the Minutes release for the meeting dated January 28th, 2009. You might notice that some words have parsing irregularities, such as the lack of space between characters on the parentheses surrounding "millions of dollars equivalent." This section of text was chosen to represent the 'worst-case-scenario' for parsing these texts. No algorithm is perfect, but one can see these errors are infrequent and therefore unlikely to heavily skew the results of the analysis. The text is from a table detailing relations with the Bank of Canada and the Bank of Mexico–tables typically have issues parsing.

A great place to start any textual analysis is with word counts.

Common Words

Brian F. Madigan is the Secretary who writes all these meetings–he has been the secretary for at least the last 11 years! In a way, we are exploring one man's work over the course of a decade...

After removing 'stop words'–those words that are extremely common and provide very little meaning (Ex. "the," "of," "a," "to," etc.)–which words does Brian use most frequently?



These are words one expects to find in official banking documents; the terms "committee," "inflation," and "market" dominate across all 95 meetings. These words may be insightful to some, but most people would agree that most of these terms are uninteresting finds. Much of what Brian wrote is generic banking lingo from which little meaning can be derived. We can still use these terms to draw interesting conclusions with these terms, however; we will visualize their usage over time, for example, but we first want to know which terms are unique to different documents.

To extract words that have most significance in each document, we can add a weighting factor to terms with the tf-idf statistic.

TF-IDF

The tf-idf statistic is designed to reduce the importance of words that occur often across documents. The math behind it is based on the natural logarithm. It is the product of the 'term frequency' and 'inverse document frequency:

Term Frequency = (# of Occurrences in Document) / (# of Unique Words in Document)

Inverse Document Frequency = ln((Total # of Docs in Corpus) / (# of Docs where Term Appears))

Therefore... **tf-df = (tf * idf)**

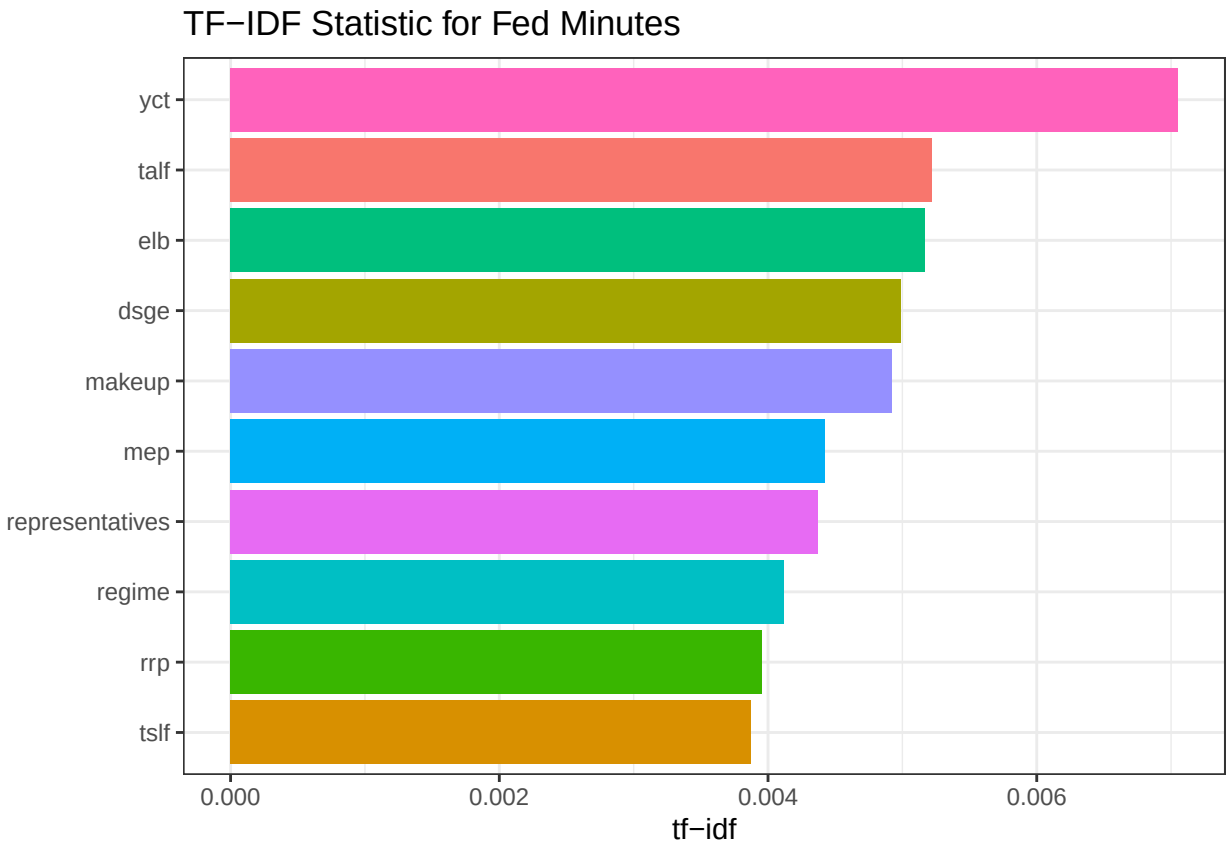
You can think of tf-idf as a formula that finds terms that occur frequently–but not too frequently. Let's see which words are most important among all Fed Minutes releases, as measured by the tf-idf:

Table 1: Important Words as Measured by tf-idf

doc_id	word	n	tf	idf	tf_idf
fomcminutes20201105	pandemic	35	0.0037977	2.762117	0.0104898
fomcminutes20200315	coronavirus	34	0.0034729	2.607967	0.0090573
fomcminutes20200916	pandemic	32	0.0032673	2.762117	0.0090247
fomcminutes20200729	pandemic	25	0.0028620	2.762117	0.0079053
fomcminutes20200315	outbreak	29	0.0029622	2.607967	0.0077253
fomcminutes20171101	hurricanes	22	0.0029283	2.607967	0.0076368
fomcminutes20200429	outbreak	24	0.0027574	2.607967	0.0071911
fomcminutes20200610	yct	15	0.0015247	4.553877	0.0069433
fomcminutes20200429	coronavirus	23	0.0026425	2.607967	0.0068915
fomcminutes20170920	hurricanes	18	0.0020829	2.607967	0.0054320

I'm tired of seeing those words everywhere. The correct interpretation of this table, however, is that the words "pandemic," "coronavirus," and "outbreak" are the most important terms, as determined by the tf-idf. Let's filter out the words "pandemic," "coronavirus," "virus," and "outbreak" to see what else tf-idf deems important. Since we all know when hurricane season rolls around, let's remove the term "hurricane" as well. Finally, after some inspection, the names of the months, "shall," and "our" should also be filtered out.

Let's see a plot of these terms after removing these terms:



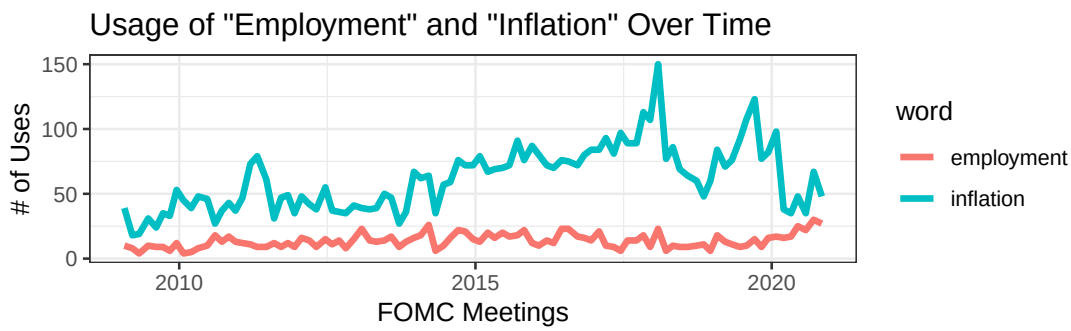
That's better. The Fed uses many different acronyms to describe its programs. To name a few, "yct" is the "yield control target," talf is short for "term asset-backed securities loan facilities," and "elb" stands for "effective lower bound."

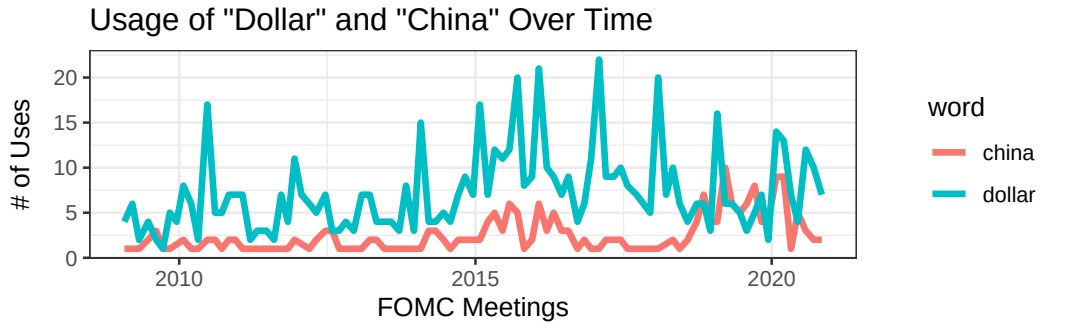
You may be wondering how these terms relate to the current events at the time. Yield curve controls (yield curve targets, YCT), for instance, were a hot topic in June 2020, when the Fed was evaluating the experiences central banks across the developed world had with controlling government bond yields in response to the recession. As a matter of fact, "yct" only appeared in *one* Fed Minutes release and appeared fifteen times! The second phrase, term asset-backed securities loan facilities (talf), was a major policy the Fed pursued in the aftermath of the Great Recession.

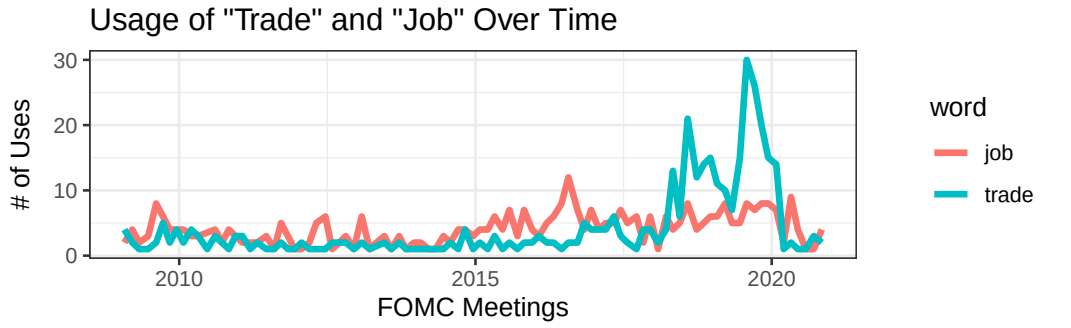
Clearly, then, we can see the value brought from this statistic. It allows us to identify important aspects of documents to hone in on topics of interest.

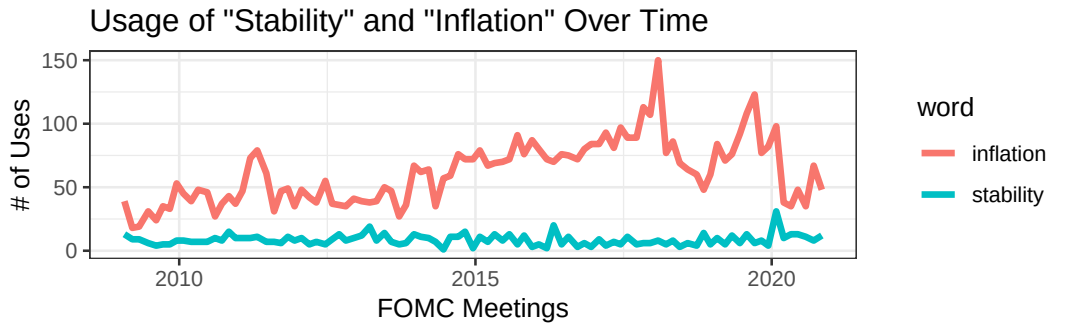
Change Over Time

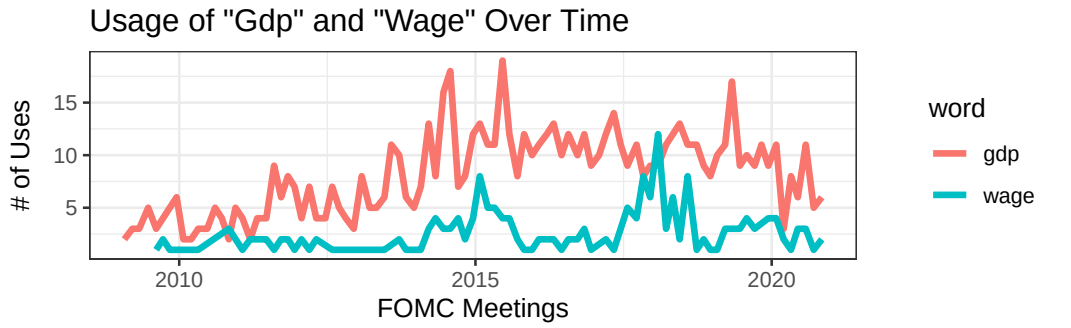
In addition to finding which terms are important to specific documents (FOMC Meetings in this instance), we can also calculate the usage of words over time. We can find some cool trends, such as the following:


Usage of "Employment" and "Inflation" Over Time
# of Uses
150
100
50
0
2010
2015
2020
FOMC Meetings
word
employment
inflation


Usage of "Dollar" and "China" Over Time
# of Uses
20
15
10
5
0
2010
2015
2020
FOMC Meetings
word
china
dollar

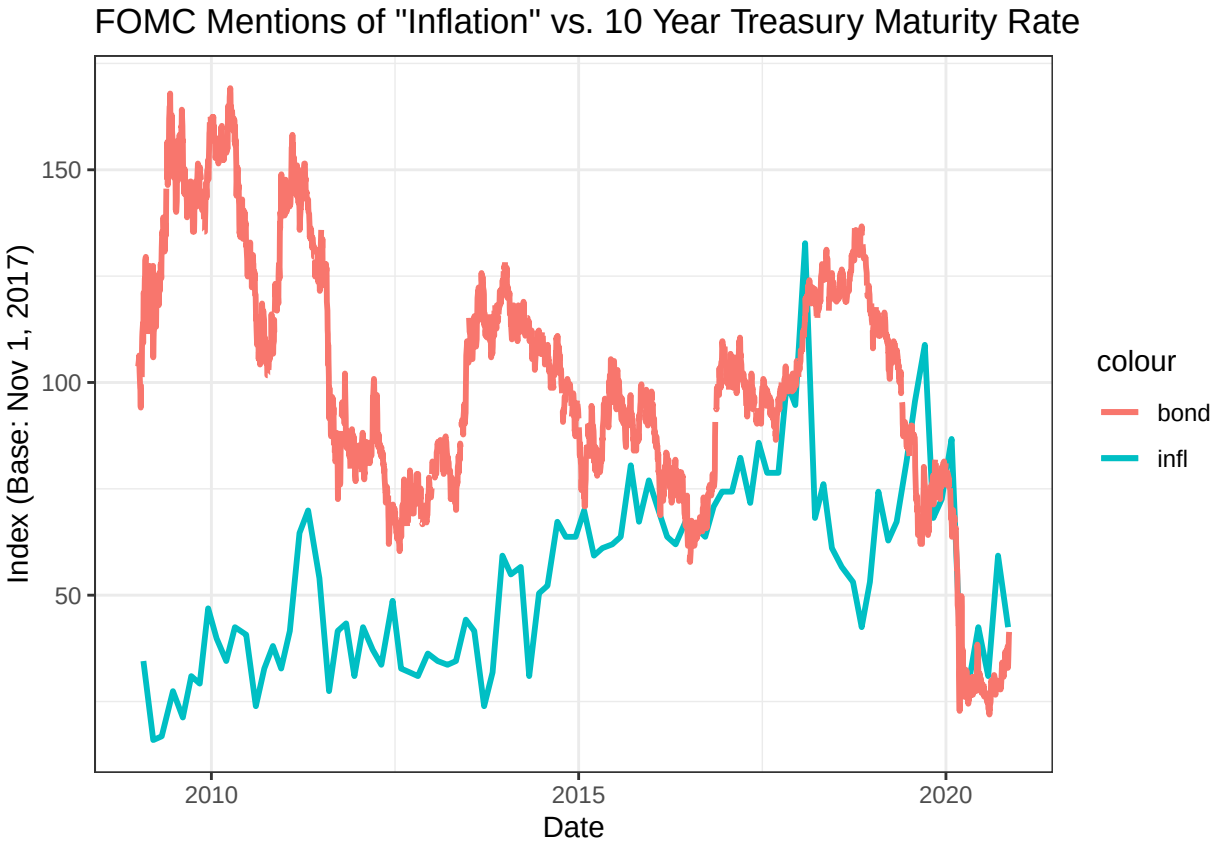

Usage of "Trade" and "Job" Over Time
# of Uses
30
20
10
0
2010
2015
2020
FOMC Meetings
word
job
trade


Usage of "Stability" and "Inflation" Over Time
# of Uses
150
100
50
0
2010
2015
2020
FOMC Meetings
word
inflation
stability


Usage of "Gdp" and "Wage" Over Time
# of Uses
15
10
5
2010
2015
2020
FOMC Meetings
word
gdp
wage

We don't have to compare the words used in FOMC meetings to words in FOMC meetings only, of course. Why don't we compare the Fed's use of "inflation" with 10 year bond yields?

Note: Only two Fed Minutes releases started on the first day of the month in the sample period, so the base will be set to 100 on November 1st, 2017. Most economic data only comes out monthly, quarterly, etc., and a common base is needed for comparison.

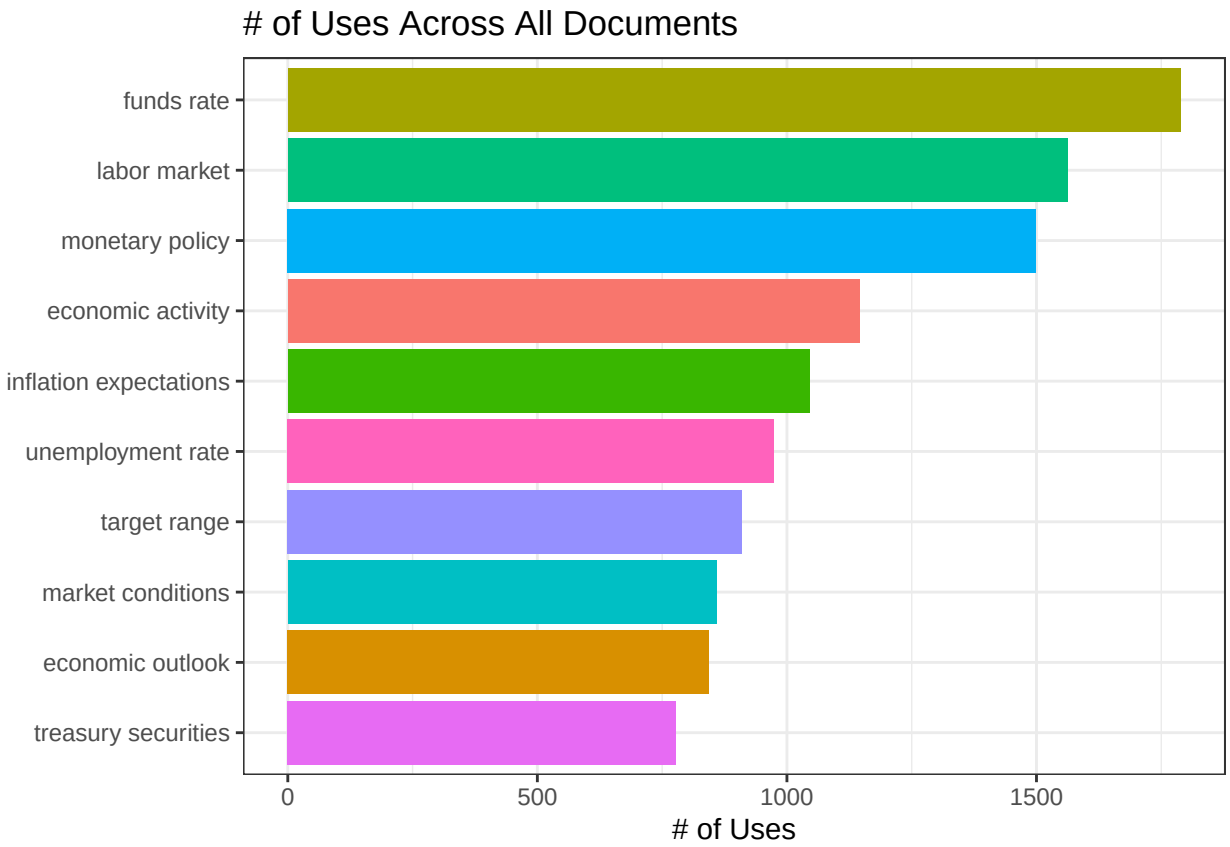


As we can see, the inflation rate and the constant maturity rate seem to be inversely related.

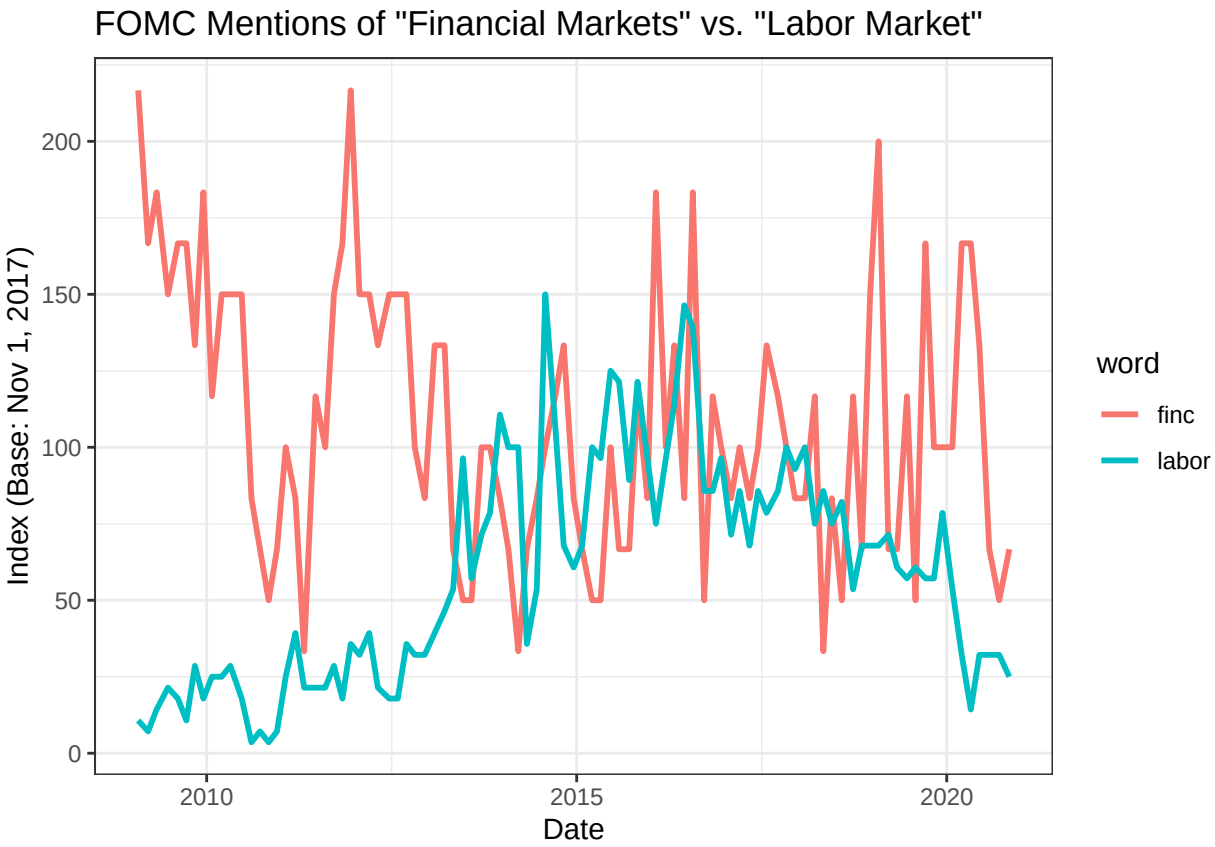
N-GRAMS

We do not need to restrict our analysis to one word. We can extract multiple words, or "n-grams," to add context to the phrases the Fed uses. We can do the same analysis in the previous sections with two or three words instead of one. After some investigation, know that the words "federal," "reserve," "committee," and "intermeeting" are also removed, as I suspect they mostly refer to formalities within the documents and not necessarily to the topics discussed at FOMC meetings.

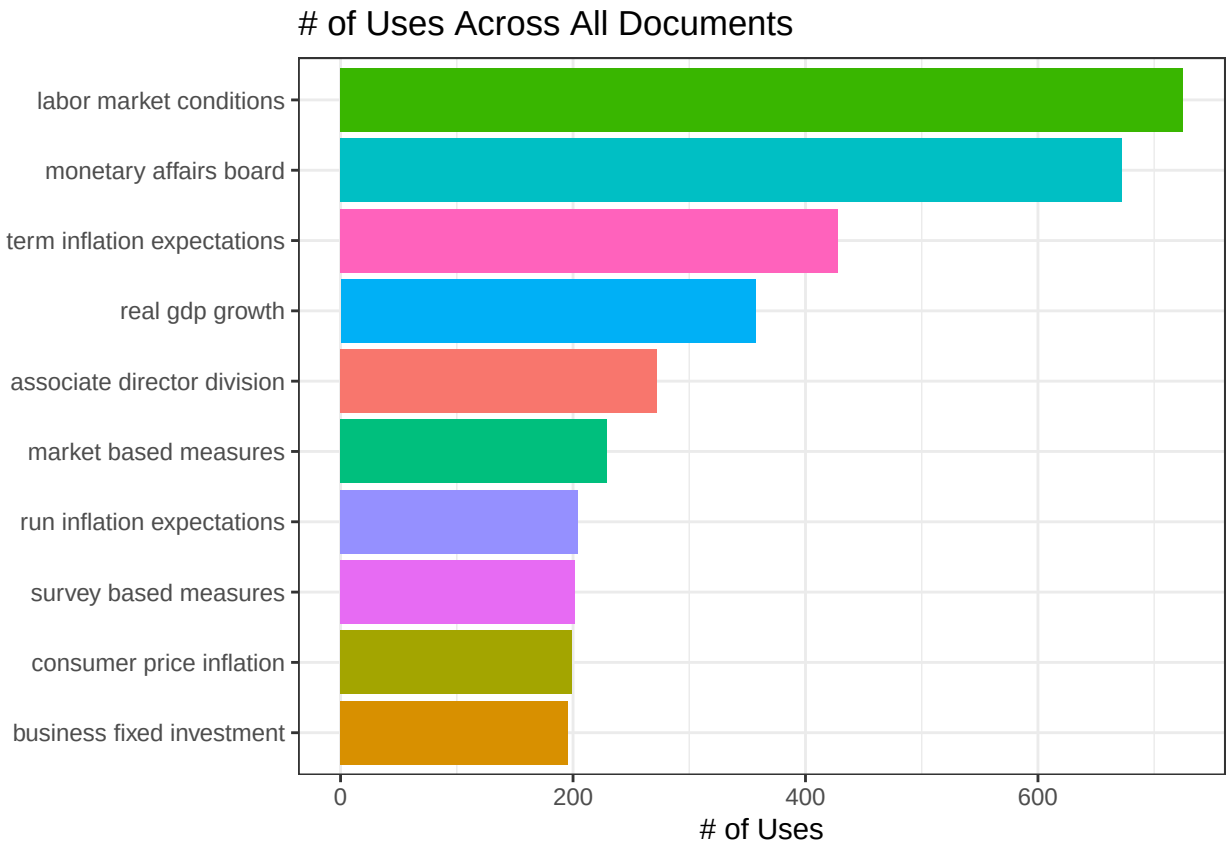
First, the most common pairs of words:



Using two words for the analysis allows us to ask more specific questions about the data. It allows us to ask things like, how has the Fed's use of "financial markets" compared with its use of "labor market" over time?



We can extend this line of thinking further with pairings of three words:



These terms might benefit from exploring combinations of four words to find words that precede these phrases, such as "*strong* labor market conditions" or "*long* term inflation expectations." This is not an optimal path to pursue, however, because the more combinations of terms to find, the smaller the data set becomes. Going from two to three terms made the "# of Uses" drop by about one-third.

Sentiment Analysis

Now that we have terms stored as bigrams (and trigrams!), we can perform sentiment analysis on the Fed Minutes releases. Various lexicons have been built up over the years to determine of a document. In this case, the Loughran and McDonald dictionary of financial sentiment words (Loughran and McDonald 2011). Other popular lexicons, such as the "AFINN," "NRC," or "BING" are not suited for documents coming from a bank–these three lexicons assign values to terms such as "share" or "risk" despite the fact those words may not carry any connotation in a financial setting.

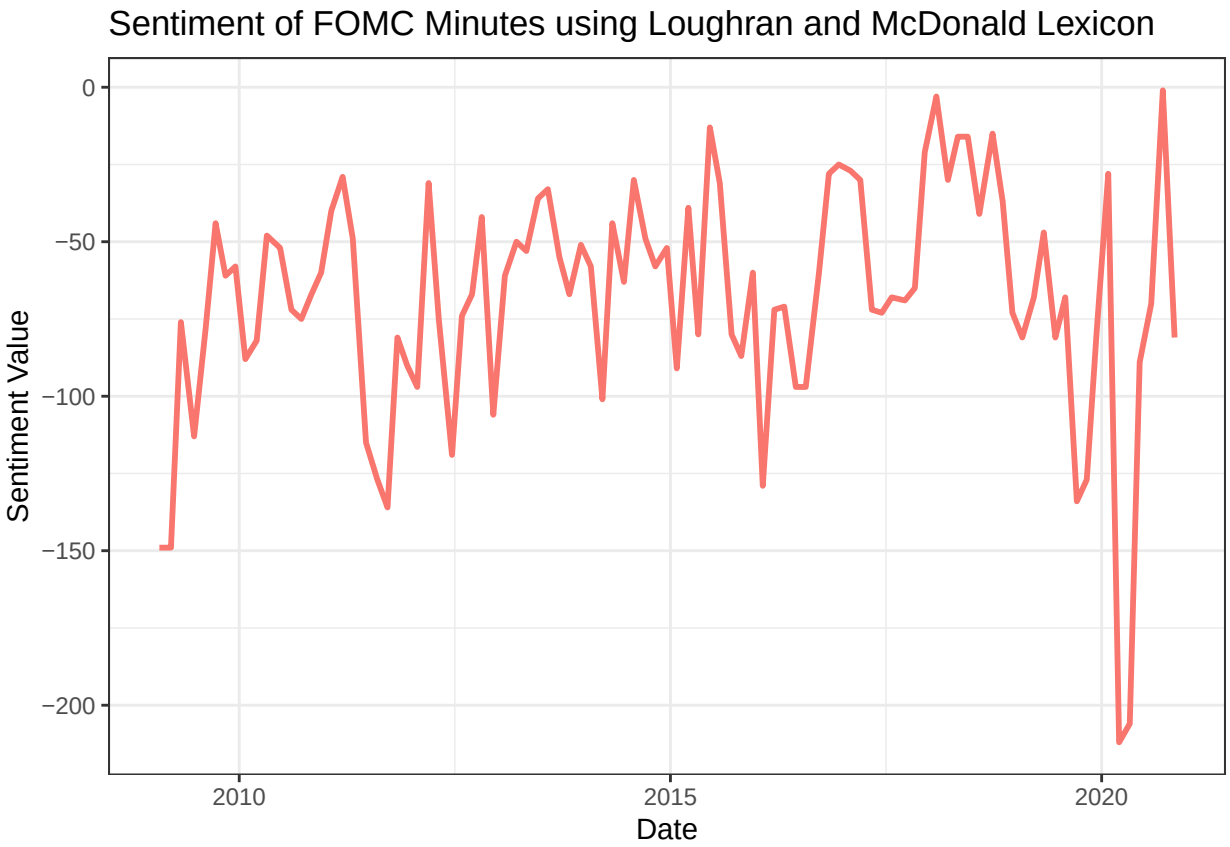
Sentiment analysis could be used for the terms without sorting them into pairs, but this adds error when negation terms are used. *Weak* growth does not mean the same thing as *strong* growth, but this distinction would never be found by using only one term.

Whenever negation words preface a word in the loughran lexicon, we can flip its sign or set its sentiment value to zero, ensuring negation words do not skew the true sentiment values.

Before continuing to the sentiment analysis, it is important to note that most sentiment lexicons have more positive words than negative words, so the results should be considered in relation to one another. For comparisons to other variables, an index is likely more appropriate.

```
negation_words <- c("not", "no", "weak", "low", "never", "without", "slow")
```

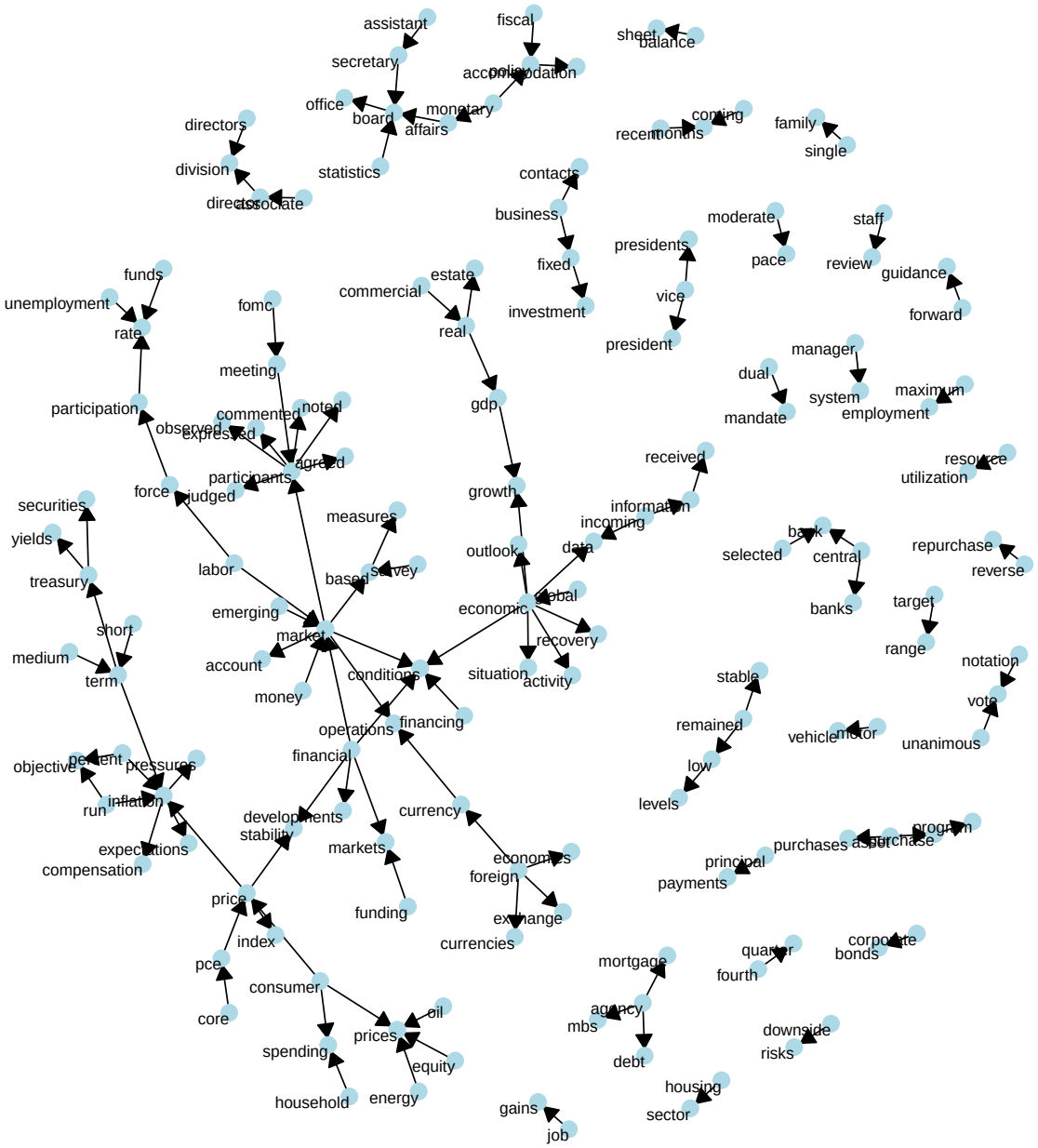
```r
# counts, then puts each word in its own column,
# then formats the date in `doc_id` and searches for negations
# (which will negate the value of `n` if TRUE),
# then joins the sentiment lexicon values,
# then filters for only the positive and negative sentiments,
# then negates values of `n` with a negative sentiment,
# then takes the sum of the values of `n` on each date,
# then plots
bigram_unnested_documents %>%
  count(doc_id, word) %>%
  tidyr::separate(col = word,
                  into = c("word1", "word2"),
                  sep = " ") %>%
  mutate(doc_id = lubridate::as_date(stringr::str_sub(doc_id, start = 12)),
         n = ifelse(word1 %in% negation_words, -n, n)) %>%
  inner_join(tidytext::get_sentiments("loughran"),
             by = c(word2 = "word")) %>%
  filter(sentiment %in% c("positive", "negative")) %>%
  tidyr::pivot_wider(names_from = sentiment,
                     values_from = n,
                     values_fill = 0) %>%
  group_by(doc_id) %>%
  summarize(sentiment = sum(positive) - sum(negative)) %>%
  ggplot(aes(x = doc_id, y = sentiment, color = "red")) +
  geom_line(show.legend = F, size = 1.02) +
  xlab("Date") +
  ylab("Sentiment Value") +
  ggtitle("Sentiment of FOMC Minutes using Loughran and McDonald Lexicon") +
  theme_bw()
```



Look at that dip at the beginning of the 2020 recession! They don't call economics the dismal science for nothing...

Markov Chains

To visualize the relationship between the words further, we can arrange groups of words into a network.



We see how the words relate to one another. Many are interconnected in the massive web going down the left side, but other networks are also formed for separate groups of associated words. While terms such as "participants," "market," "economic," or "inflation" dominate the large network, other networks exist largely independent of these words, such as "job gains" or "forward guidance."

While this visualization of a Markov Chain allows us to visualize which words are typically follow one another, it does not show which words are related within documents. What do we do when we want to see which words appear in the same document (though they might not be adjacent to one another)?

A correlation test addresses this issue.

Correlated Words

We can calculate Pearson correlation coefficients to determine how often particular words appear with one another. Examining correlation coefficients across each of the FOMC Minutes is not ideal since each one consists of thousands of words, many of them being used in each document. The most highly correlated words, broken down by document as opposed to by another tokenization, *that are not perfectly correlated* are still provided below, however. Many of the documents are more than ten pages long and refer to similar topics, so many of the words will have perfect correlation.

Table 2: Phi Coefficients for Whole Documents

item1	item2	correlation
participants	committee	-0.612
inflation	committee	-0.471
policy	participants	0.459
inflation	federal	-0.367
market	committee	0.343
policy	inflation	0.313
policy	committee	-0.281
participants	market	-0.210
participants	inflation	0.157
market	federal	0.140

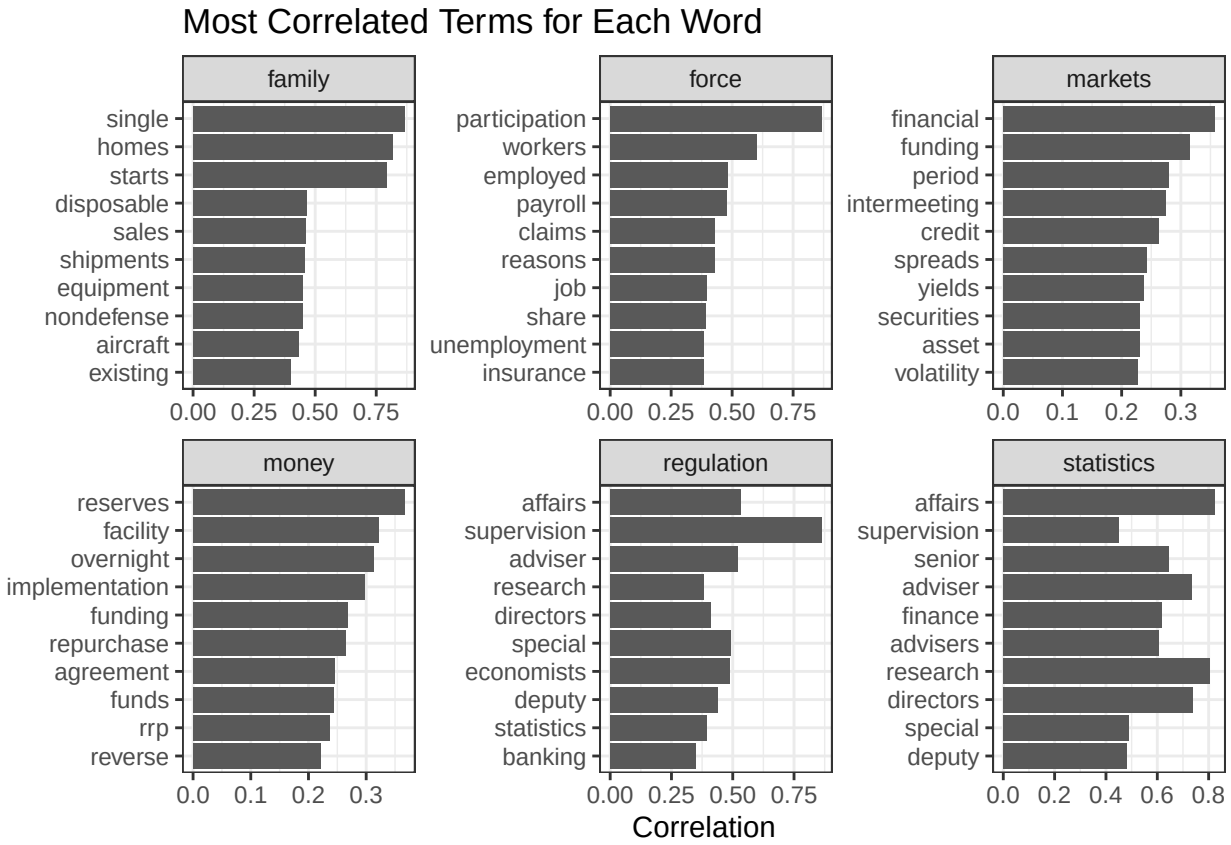
As implied above, what should instead be done is break the documents into smaller pieces and calculate correlation coefficients for each of these subsections. To ensure proper nouns do not dominate the results with perfect correlation (and because R said my computer ran out of memory =/), only those terms which appear more than 100 times will be used to calculate the coefficients. Also note that a significant number of words were removed–words such as "secretary," "jerome," "tuesday," "dallas," "secretary," "governors," and many similar terms–to keep only the information we are interested in.

Table 3: Phi Coefficients for Document Sections

Word 1	Word 2	Correlation
speculative	grade	0.916
notation	completed	0.876
force	participation	0.867
single	family	0.866
supervision	regulation	0.863
accordance	authorize	0.836
insurance	claims	0.830
shipments	nondefense	0.826
affairs	statistics	0.824
homes	family	0.817

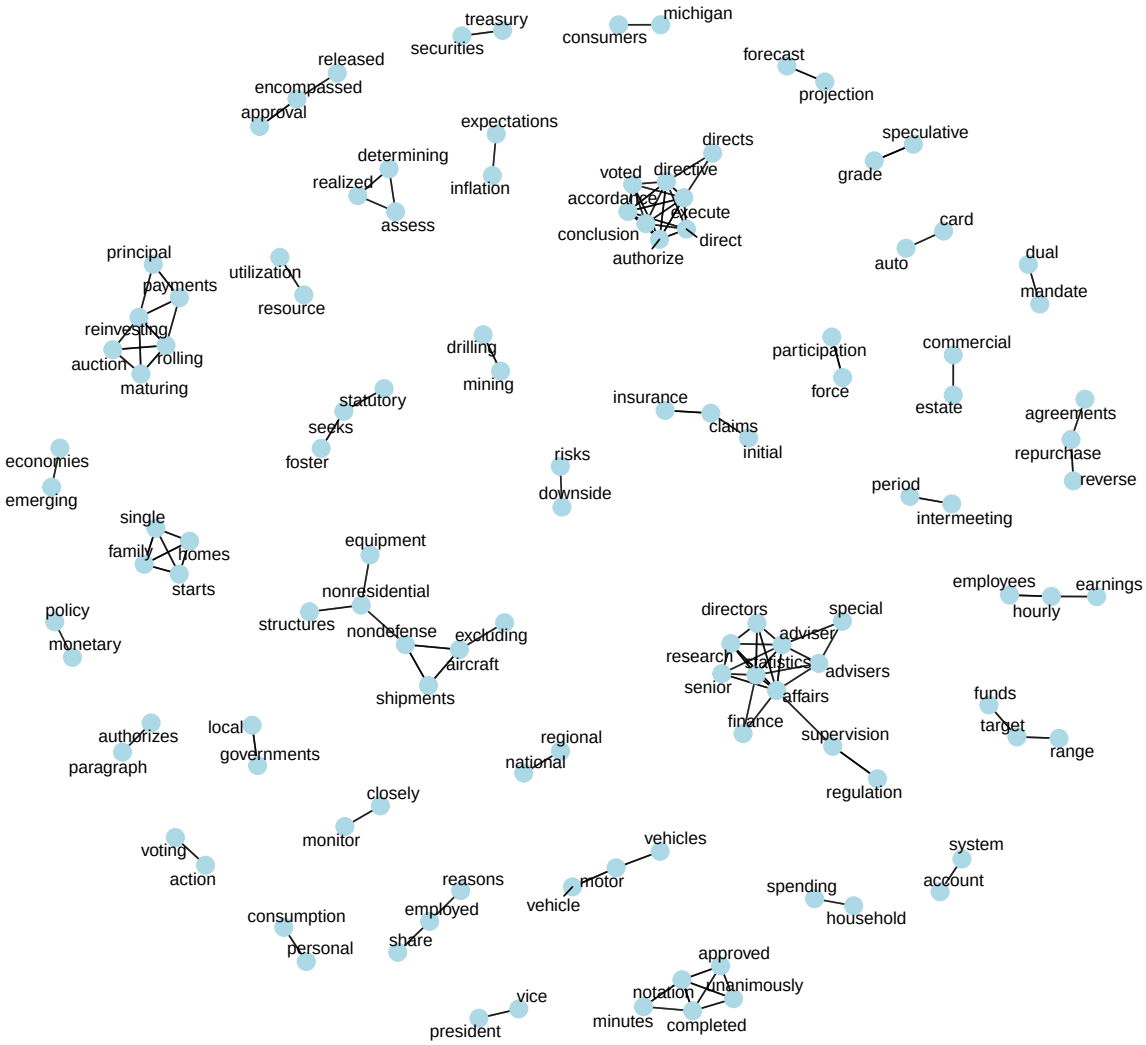
The remainder of this section uses data from the above table.

Let's see a nice graph showing the top 10 correlations of six words:



Pay attention to the scale and note that it is not the same for all words. These graphs paint a similar picture to the graphs from the N-grams section, though this graph is a bit more telling because even though words that frequently appear next to each other are included in these graphs (Ex. "[labor] force participation"), words that often occur in the same sentence or set of sentences appear as well (Ex. "family" and "disposable" are unlikely to be written after one another in a document, the only way for n-grams to find a connection between the two, though correlation coefficients still recognize these co-occurrences).

One of the best uses of these correlations is to pick interesting words and find how other terms relate to them in a network, similar to that from the previous section. Note that since there is no directional relationship, there are no arrows on the graph. Since the number of words is enormous, only those terms with at a correlation of at least 0.6 are included.



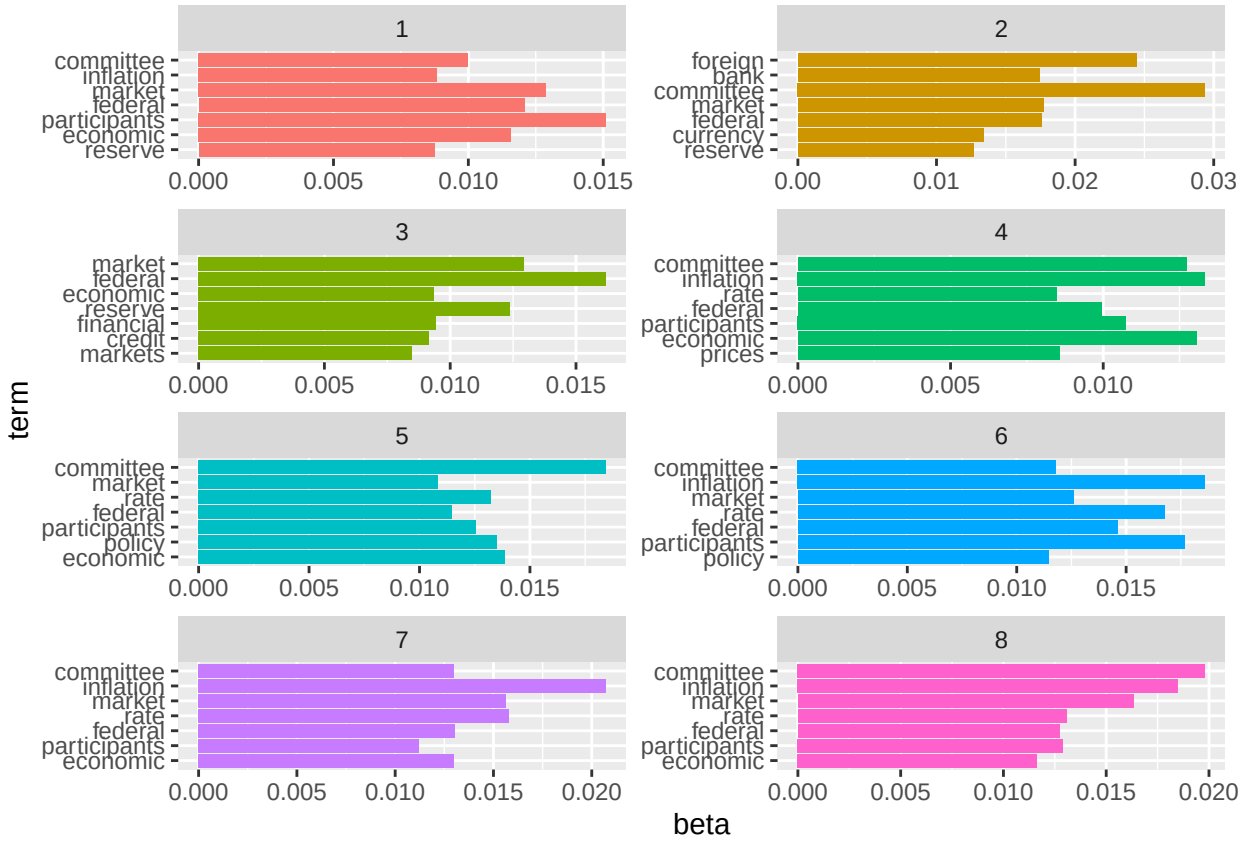
As noted, this graph is very similar to the one from the Markov Chains section. The difference between this graph and its counterpart follows the same explanation provided moments before; n-grams 'pick up' fewer relationships between terms than do correlation coefficients. Correlation coefficients capture relationships between all words, not only those found in succession. One shortcoming of this graph compared to the one from the previous section is that this graph cannot contain directional arrows, so we cannot tell whether these words were used in conjunction with one another, or whether they merely relate to similar topics.

Topic Modeling

Speaking of topics, we can use algorithms to identify topics within the dataset. I will be using the latent Dirichlet allocation algorithm to identify these topics. LDA captures document-topic probabilities (document is defined loosely as "a group of words"). Every document is a mixture of topics, and every topic is a mixture of words. For example, the topics in the FOMC Minutes of January 28th, 2009 could be 40% inflation, 30% financial markets, and 30% labor markets (*every document is a mixture of topics*). The inflation could be made up of words such as "rate," "cpi," etc., while the financial markets topic could be made up of terms

such as "rate," "yield," etc. (*every topic is a mixture of words*).

You might have noticed that the word "rate" appeared in both the inflation and financial markets topics. That is fine–unlike other unsupervised learning/clustering methods, LDA allows words to be reused in different topics. Just as two words have different meanings in human language, two words can appear in different topics with LDA.

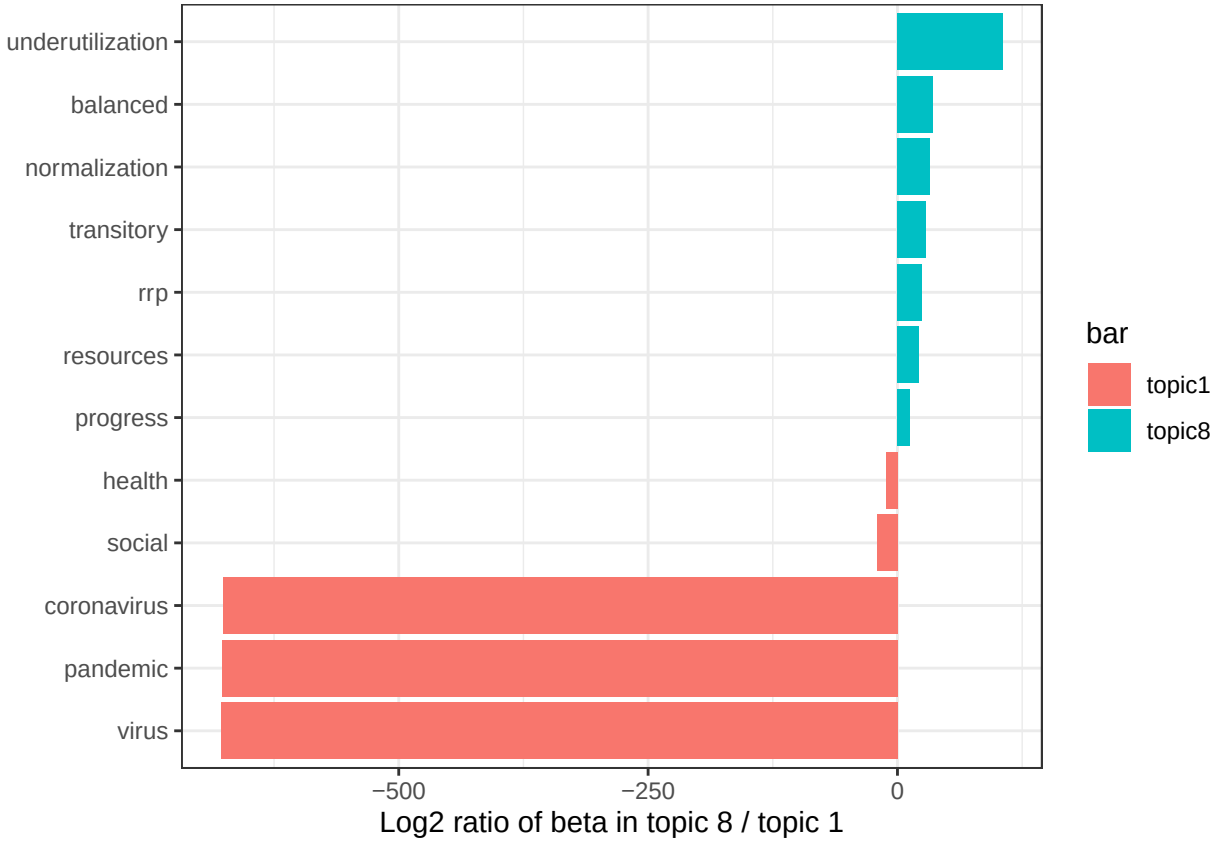


There is much overlap in the topics the model found. This is not surprising–FOMC Minutes usually have a predictable agenda they need to get through.

We may instead be interested in those words having the greatest difference between groups (as opposed to most likely to appear in a particular topic) in each category. We can use a log odds ratio to calculate this difference. For example, which words are most different between topic 8 and topic 1? (For those unfamiliar with the math, just know that the negative values are terms belonging to topic 1 and the positive values belong to topic 8.)

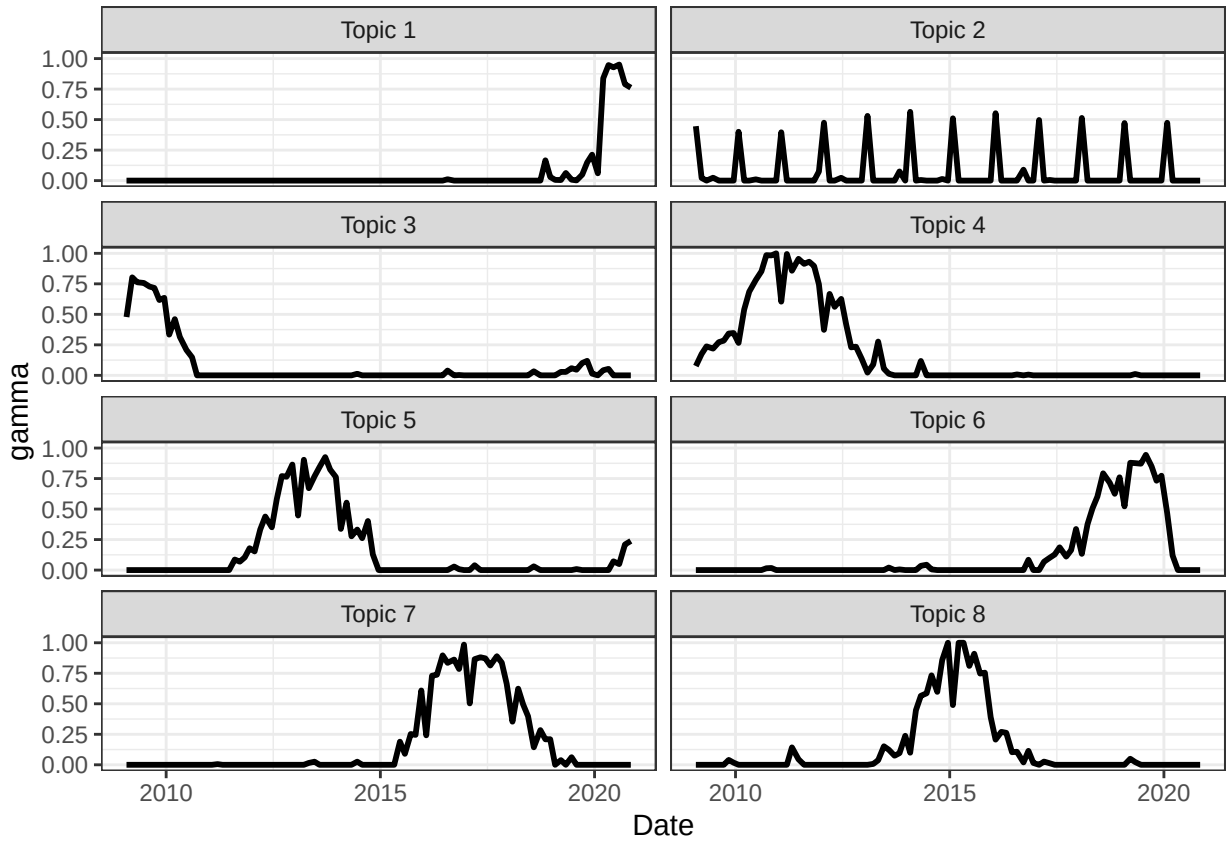
```
fed_topics_beta %>%
  mutate(topic = paste0("topic", topic)) %>%
  tidyr::pivot_wider(names_from = topic, values_from = beta) %>%
  filter(topic8 > 0.001 |topic1 > 0.001) %>%
  mutate(log_ratio = log(topic8 / topic1)) %>%
  filter(abs(log_ratio) > 10) %>%
  mutate(bar = ifelse(log_ratio < 0, "topic1", "topic8")) %>%
  ggplot(aes(x = reorder(x = term, log_ratio), y = log_ratio, fill = bar)) +
  geom_col() +
  ylab("Log2 ratio of beta in topic 8 / topic 1") +
  xlab(NULL) +
  ggtitle(NULL) +
```

```
  theme_bw() +
  coord_flip()
```



If nothing else, we are all familiar with topic 1...

In addition to finding which words belong to which topics, we can also determine which topics belong to which documents. Since we know roughly when topic 1 is identified, we can observe just when these topics were most likely to occur.



In the event you did not believe me, the probability of topic 1 appearing in FOMC Minutes shoots up into the sky during meetings in 2020.

References

Board of Governors of the Federal Reserve System (US), 10-Year Treasury Constant Maturity Rate [DGS10], retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/DGS10, January 7, 2021.

Loughran, T. and McDonald, B. (2016). Textual analysis in accounting and finance: A survey. Journal of Accounting Research 54(4), 1187-1230. doi: 10.2139/ssrn.2504147 https://sraf.nd.edu/textual-analysis/resources/#Master%20Dictionary.