Assignment 2 - Group 4

Team members - Jingwen Li, Elie Kostenbaum, Moushumi Pardesi

Team members' initials have been used throughout the HW to denote who worked on the specific part.

1. Measuring Hawkish/Dovish Tone of FOMC Statements (100 points)

1. Scrape the text of the FOMC statements from January 2000 to present. You will need to use https://www.federalreserve.gov/monetarypolicy/fomccalendars.htm/for 2018-2023 and/https://www.federalreserve.gov/monetarypolicy/fomc_historical_ year.htm for 2000-2017.

(a) How many statements do you obtain? (b) Provide summary statistics (mean, standard deviation, minimum, first quartile, median, third quartile, maximum) for the number of words in each statement.

In [ ]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
import datetime
import nltk
import re
from nltk.tokenize import sent_tokenize
from nltk.corpus import stopwords
import requests
import bs4
import re
import time
import pandas as pd
from datetime import datetime
from nltk.tokenize import sent_tokenize
import matplotlib.pyplot as plt
import string
from torch.nn.functional import softmax
import torch
from transformers import GPT2Tokenizer, GPT2ForSequenceClassification, pipeline
```

```
from statsmodels.tsa.ar_model import AutoReg
import statsmodels.api as sm
from scipy import stats

import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
def gen_soup_bs4(url):
    '''takes a url and returns a soup object'''

    headers = {'User-Agent': 'Mozilla/5.0 (Linux; Android 5.1.1; SM-G928X Build/LMY47X) ' + \
                          'AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.83 Mobile Safari/537
              "Accept-Encoding":"gzip, deflate",
              "Accept":"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
              "DNT":"1","Connection":"close",
              "Upgrade-Insecure-Requests":"1"}

    response = requests.get(url = url, headers = headers)
    assert response.status_code == 200 #need to be 200 (ideally)
    soup = bs4.BeautifulSoup(response.text, "html.parser")
    time.sleep(1)

    return soup

def generate_absolute_url(relative_url):
    '''get absolute URL from relative URL'''

    domain = 'https://www.federalreserve.gov'
    return domain + relative_url
```

In [ ]:
```
seed_url = 'https://www.federalreserve.gov/monetarypolicy/fomchistorical{}.htm'
pre2016 = [seed_url.format(n) for n in range(2000, 2017)]

pre2016_statements_url = []

for url in pre2016:
    soup = gen_soup_bs4(url)
    for i in soup.find_all('a', {'href': re.compile(r'\/\w+\/press\w*\/')}):
        if i.text.lower() == 'statement':
            statement_url = generate_absolute_url(i['href'])
            pre2016_statements_url.append(statement_url)
```

```
seed_url_2017 = 'https://www.federalreserve.gov/monetarypolicy/fomccalendars.htm'
soup_post2017 = gen_soup_bs4(seed_url_2017)

post2017_statements_url = []
for i in soup_post2017.find_all('a', {'href': re.compile(r'\/newsevents\/pressreleases\/')}):
    if i.text.lower() == 'html': #just get the statements
        statement_url = generate_absolute_url(i['href'])
        post2017_statements_url.append(statement_url)

all_urls = pre2016_statements_url + post2017_statements_url
```

In [ ]:

```
#scrape the text of statements
data = {
    'date': [],
    'full_text': []
}

for url in all_urls:

    statement_soup = gen_soup_bs4(url)
    date = re.findall(r'\d{5,}', url)[0] #get date of the statement
    year = int(date[:4]) #year of statement

    #federal reserve changed their website after 2005. Need to cater differently
    if year < 2006:
        try:
            txt = statement_soup.find_all('tr')[1].text
        except:
            txt = statement_soup.find_all('tr')[0].text
    else:
        txt = statement_soup.find('div', attrs={'class': 'col-xs-12 col-sm-8 col-md-8'}).text

    #remove unnecessary text
    for pattern in ['(\\n|\\t|\\r)', 'Voting .* policy .* were.*',
                    '(For immediate release|For release at .* E*T)',
                    'Frequently Asked Questions.*',
                    '[\w\s]+other central banks is available at the following websites:.*',
                    'Release Date:.*?(?=T)']:
        txt = re.sub(pattern, '', txt)

    data['date'].append(date)
```

```
    data['full_text'].append(txt)

statements_df = pd.DataFrame(data)
statements_df['date'] = pd.to_datetime(statements_df['date'])
statements_df.sort_values('date', inplace=True)
```

In [ ]:
```
statements_df['count'] = statements_df['full_text'].apply(lambda x: len(x.split()))
print('(a) JL & EK')
len(statements_df)
```

```
(a) JL & EK
```

Out[ ]:
```
194
```

In [ ]:
```
print('(b) JL & EK')
statements_df['count'].describe()
```

```
(b) JL & EK
```

Out[ ]:
```
count    194.000000
mean     323.742268
std      169.926853
min       77.000000
25%      175.000000
50%      286.000000
75%      430.500000
max      798.000000
Name: count, dtype: float64
```

2. Use the methodology described in section 3.1 (pages 4-8) of Tadle (2022) to measure the tone of each speech 2^2

Download the Fed Funds Effective Rate from https://fred.stlouisfed.org/series/DFF. Plot both both the statement tone and the Fed Funds Effective Rate over time.

In [ ]:
```
statements_df['date'] = pd.to_datetime(statements_df['date'], format='%Y%m%d')
statements_df = statements_df.sort_values(by='date')
statements_df = statements_df.set_index('date')
df_keywords = pd.read_excel('Tadle2022_keywords.xlsx')
df_funds = pd.read_csv('DFF.csv')


def tokenize_and_clean(statements_df):
```

```
    sentences = sent_tokenize(statements_df)
    sentences = [sentence.translate(str.maketrans('', '', string.punctuation)).lower() for sentence in sen
    return sentences

def contains_keywords(sentence, keywords):
    sentence = str(sentence)
    keywords = [str(keyword) for keyword in keywords]
    return any(keyword in sentence for keyword in keywords)


def score_sentence(sentence, df_keywords):
    pos_keys = set(df_keywords['positive'])
    neg_keys = set(df_keywords['negative'])
    negations = set(df_keywords['negation'])
    hawkish = set(str(keyword) for keyword in df_keywords['hawkish'])  # Convert to string
    dovish = set(str(keyword) for keyword in df_keywords['dovish'])    # Convert to string

    tokens = nltk.word_tokenize(sentence)

    p = 0
    n = 0

    for idx, token in enumerate(tokens):
        if token in pos_keys:
            if any(tok in tokens[max(idx-3,0):idx] for tok in negations):
                n += 1
            else:
                p += 1
        elif token in neg_keys:
            if any(tok in tokens[max(idx-3,0):idx] for tok in negations):
                p += 1
            else:
                n += 1

    if (p > n and any(word in sentence for word in hawkish)) or (p < n and not any(word in sentence for wo
        return 1
    elif (p > n and any(word in sentence for word in dovish)) or (p < n and any(word in sentence for word
        return -1
    else:
        return 0

statements_df['sentences'] = statements_df['full_text'].apply(tokenize_and_clean)
```

```
filtered_sentences = statements_df['sentences'].apply(lambda sentences: [s for s in sentences if contains_l
statements_df['scores'] = filtered_sentences.apply(lambda sentences: [score_sentence(s, df_keywords) for s
statements_df['index'] = statements_df['scores'].apply(lambda scores: 100 * np.mean(scores) if len(scores)

df_funds.set_index('DATE', inplace=True)
df_funds.index = pd.to_datetime(df_funds.index)

for date in statements_df.index:
    statements_df.loc[date, 'fed_rate'] = df_funds.loc[date, 'DFF']
```

In [ ]:

```
statements_df
```

Out[ ]:

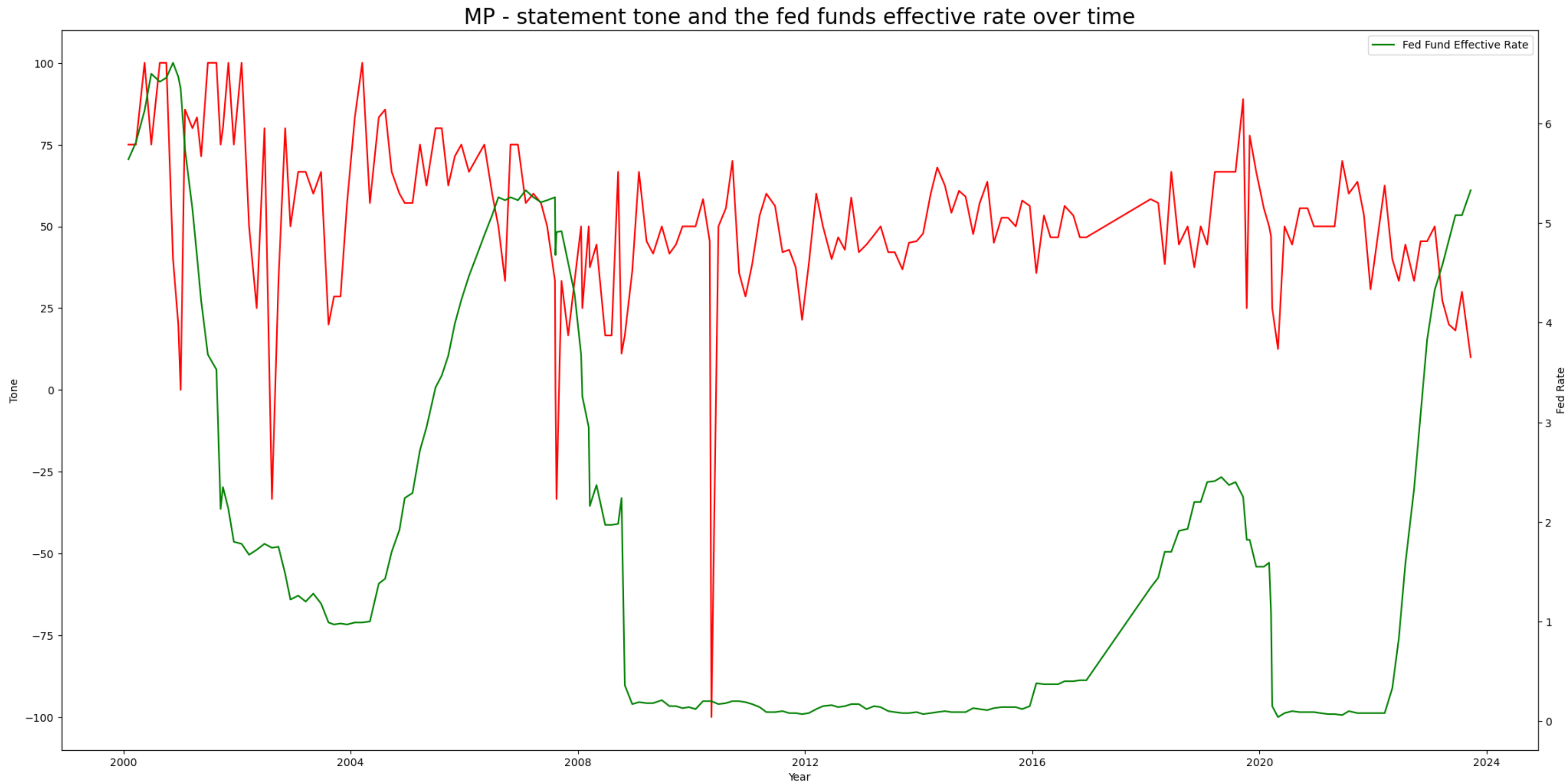
date	full_text	count	sentences	scores	index	fed_rate
2000-02-02	The Federal Open Market Committee voted today ...	187	[the federal open market committee voted today...	[1, 1, 0, 1]	75.000000	5.64
2000-03-21	The Federal Open Market Committee voted today ...	194	[the federal open market committee voted today...	[1, 0, 1, 1]	75.000000	5.81
2000-05-16	The Federal Open Market Committee voted today ...	184	[the federal open market committee voted today...	[1, 1, 1]	100.000000	6.13
2000-06-28	The Federal Open Market Committee at its meeti...	161	[the federal open market committee at its meet...	[0, 1, 1, 1]	75.000000	6.50
2000-08-22	The Federal Open Market Committee at its meeti...	162	[the federal open market committee at its meet...	[1, 1]	100.000000	6.42
...	...	...	...	...	...	...
2023-03-22	Recent indicators point to modest growth in sp...	320	[recent indicators point to modest growth in s...	[0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0]	27.272727	4.58
2023-05-03	Economic activity expanded at a modest pace in...	286	[economic activity expanded at a modest pace i...	[0, 1, 0, 0, 0, 1, 0, 0, 0, 0]	20.000000	4.83
2023-06-14	Recent indicators suggest that economic activi...	296	[recent indicators suggest that economic activ...	[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]	18.181818	5.08
2023-07-26	Recent indicators suggest that economic activi...	288	[recent indicators suggest that economic activ...	[1, 1, 0, 0, 0, 1, 0, 0, 0, 0]	30.000000	5.08
2023-09-20	Recent indicators suggest that economic activi...	290	[recent indicators suggest that economic activ...	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]	10.000000	5.33

194 rows × 6 columns

In [ ]:

```
fig, ax1 = plt.subplots(figsize=(20, 10))
fig.tight_layout()
ax2 = ax1.twinx()
ax2.plot(statements_df.index, statements_df['fed_rate'], 'g-', label='Fed Fund Effective Rate')
ax1.plot(statements_df.index, statements_df['index'], 'r-', label='Statement Tone')
ax1.set_xlabel('Year')
ax1.set_ylabel('Tone')
ax2.set_ylabel('Fed Rate')
plt.legend()
```

```
plt.title("MP - statement tone and the fed funds effective rate over time", fontsize=20)
plt.show()
```



3. Comment on the Tadle (2022) methodology. What do you like about it? What are its shortcomings?

JL, EK, MP

Like - The methodology is able to assess the statement tone in a time series format which can be very helpful for say an automated trading strategy where a quick assessment is necessary and there isn't much time after the statement being released for an analyst to read and then judge the market direction or effect. Also, The methodology processes a large volume of FOMC documents, which can be a valuable resource for economic analysis, converting unstructured text data into something suitable for quantitative analysis.

I think the scope of Tadle 2022 can also be expanded to check the effect of the fed funds statements on treasuries, tbills, etc. to incorporate into a medium or high frequency trading strategy. Thus, the scalability of this method is really a strong point.

Dislike - It is pretty new, so the reliability that comes over time with multiple independent citations and tests is lacking. Also, it is very dependent on the accurate outline of hawkish/dovish vocabulary fed into the system. If the vocabulary is flawed, then the output would be flawed. Also, the use of a dictionary of positive and negative terms for sentiment analysis can be limited in capturing nuanced sentiments. It may not account for context-specific meanings of word or the evolving language used in FOMC documents

4. Describe and implement a different way to measure hawkish/dovish tone of FOMC statements

How does your alternative measure address some of the shortcomings in the Tadle (2022) method? What is the correlation between the Tadle (2022) measure and your measure?

MP - Alternative 1 - TextBlob library

One alternative way could be to TextBlob identify the sentiment of text whether hawkish/ dovish. Both methods are based on lexicon. But TextBlob uses a sentiment dictionary that contains so called polarity scores of many words. The polarity score means whether it is positive or negative. To calculate the sentiment of a text, TextBlob simply calculates the average polarity score of all the words in the text. This is like having a tadle dictionary.

Address shortcomings

1. Tadle 2022 is based on a fixed set of hawkish and dovish words and phrases so the method is unable to accurately capture the hawkish/dovish tone of texts that use new language or vague sentences. textblob measure addresses this shortcoming by using a machine learning model to identify the hawkish/dovish tone of texts. textblob is trained on a large dataset.

2. Tadle 2022 does not consider sequence of words in a text. This means that the method may not be able to accurately distinguish between hawkish and dovish texts that have similar word counts but different word orders. textblob is able to consider the order of words in a text.

Correlation

Expected to be high but textblob might be more accurate.

Implementation -

In [ ]:
```
from textblob import TextBlob

def analyze_sentiment(statement):
    analysis = TextBlob(statement)
    sentiment_score = analysis.sentiment.polarity
    return sentiment_score

statements_df['Sentiment_Score'] = statements_df['full_text'].apply(analyze_sentiment)
```

In [ ]:
```
statements_df.reset_index()
```

Out[ ]:

	date	full_text	count	sentences	scores	index	fed_rate	Sentiment_Score
0	2000-02-02	The Federal Open Market Committee voted today ...	187	[the federal open market committee voted today...	[1, 1, 0, 1]	75.000000	5.64	0.118457
1	2000-03-21	The Federal Open Market Committee voted today ...	194	[the federal open market committee voted today...	[1, 0, 1, 1]	75.000000	5.81	0.107359
2	2000-05-16	The Federal Open Market Committee voted today ...	184	[the federal open market committee voted today...	[1, 1, 1]	100.000000	6.13	0.146667
3	2000-06-28	The Federal Open Market Committee at its meeti...	161	[the federal open market committee at its meet...	[0, 1, 1, 1]	75.000000	6.50	0.091313
4	2000-08-22	The Federal Open Market Committee at its meeti...	162	[the federal open market committee at its meet...	[1, 1]	100.000000	6.42	0.139259
...	...	...	...	...	...	...	...	...
189	2023-03-22	Recent indicators point to modest growth in sp...	320	[recent indicators point to modest growth in s...	[0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0]	27.272727	4.58	0.153030
190	2023-05-03	Economic activity expanded at a modest pace in...	286	[economic activity expanded at a modest pace i...	[0, 1, 0, 0, 0, 1, 0, 0, 0, 0]	20.000000	4.83	0.181746
191	2023-06-14	Recent indicators suggest that economic activi...	296	[recent indicators suggest that economic activ...	[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]	18.181818	5.08	0.169697
192	2023-07-26	Recent indicators suggest that economic activi...	288	[recent indicators suggest that economic activ...	[1, 1, 0, 0, 0, 1, 0, 0, 0, 0]	30.000000	5.08	0.165079
193	2023-09-20	Recent indicators suggest that economic activi...	290	[recent indicators suggest that economic activ...	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]	10.000000	5.33	0.177273

194 rows × 8 columns

In [ ]:

```
keywords = pd.read_excel('Tadle2022_keywords.xlsx')
dff = pd.read_csv('DFF.csv')
dff['DATE']=pd.to_datetime(dff['DATE'])
df = dff.merge(statements_df, left_on='DATE', right_on='date', how='inner')
df = df[['DATE','DFF','index','Sentiment_Score']]
df
```

Out[ ]:

	DATE	DFF	index	Sentiment_Score
0	2000-02-02	5.64	75.000000	0.118457
1	2000-03-21	5.81	75.000000	0.107359
2	2000-05-16	6.13	100.000000	0.146667
3	2000-06-28	6.50	75.000000	0.091313
4	2000-08-22	6.42	100.000000	0.139259
...	...	...	...	...
189	2023-03-22	4.58	27.272727	0.153030
190	2023-05-03	4.83	20.000000	0.181746
191	2023-06-14	5.08	18.181818	0.169697
192	2023-07-26	5.08	30.000000	0.165079
193	2023-09-20	5.33	10.000000	0.177273

194 rows × 4 columns

In [ ]:
```
df.corr()
```

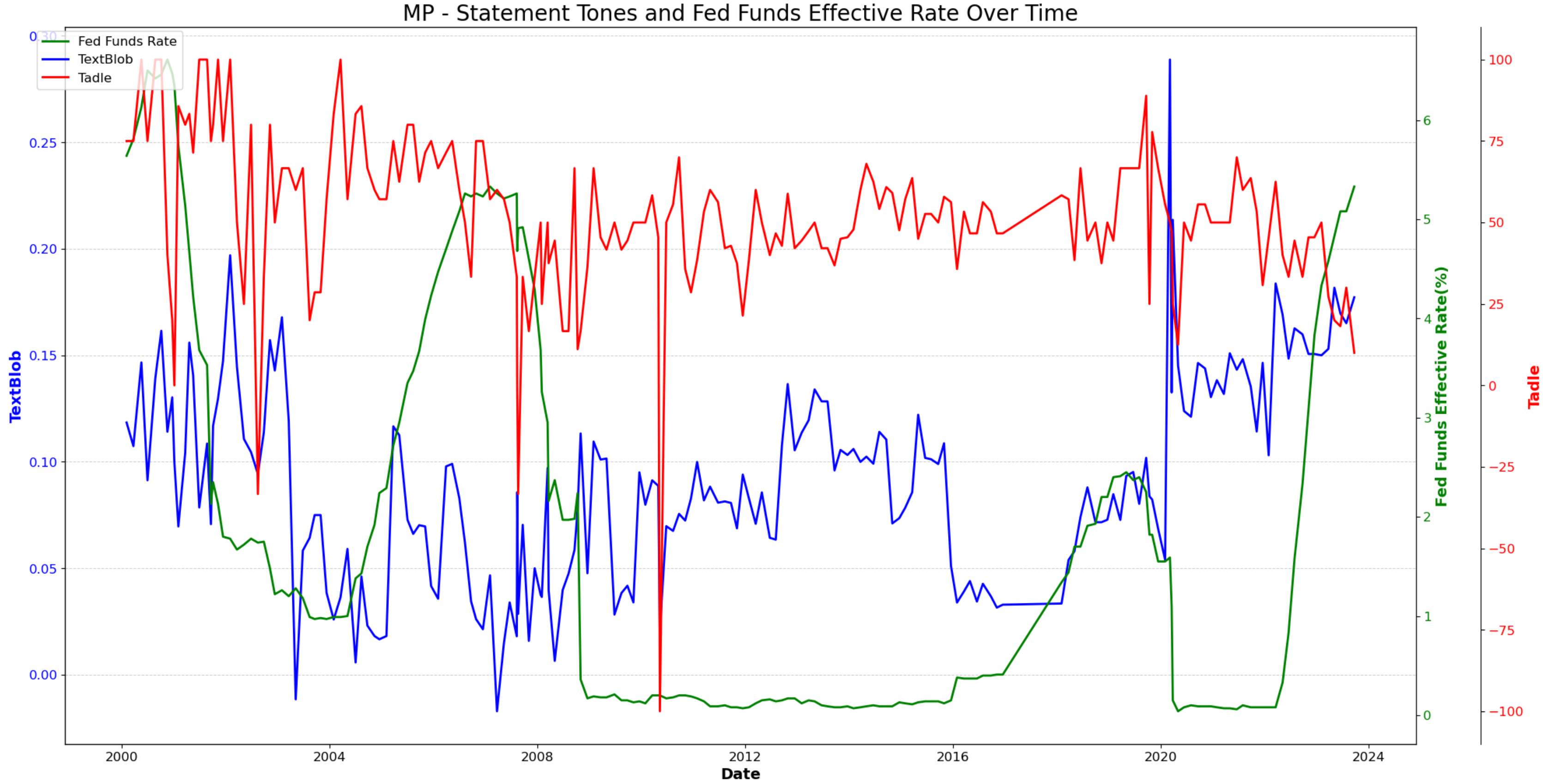
Out[ ]:

	DFF	index	Sentiment_Score
DFF	1.000000	0.116716	-0.037973
index	0.116716	1.000000	0.053878
Sentiment_Score	-0.037973	0.053878	1.000000

5. Redo the plot from problem 2, adding your tone measure. Your plot should include the Tadle (2022) measure, your measure and the Fed Funds Effective Rate.

In [ ]:
```
fig, ax1 = plt.subplots(figsize=(20, 10))

color = 'blue'
ax1.set_xlabel('Date', fontsize=14, fontweight='bold')
ax1.set_ylabel('TextBlob', color=color, fontsize=14, fontweight='bold')
```

```
line1, = ax1.plot(df['DATE'], df['Sentiment_Score'], color=color, label='TextBlob', linewidth=2)
ax1.tick_params(axis='y', labelcolor=color)
ax1.tick_params(axis='x', labelsize=12)
ax1.tick_params(axis='y', labelsize=12)
ax1.grid(True, axis='y', linestyle='--', linewidth=0.7, alpha=0.6)

ax2 = ax1.twinx()
color = 'green'
ax2.set_ylabel('Fed Funds Effective Rate(%)', color=color, fontsize=14, fontweight='bold')
line2, = ax2.plot(df['DATE'], df['DFF'], color=color, label='Fed Funds Rate', linewidth=2)
ax2.tick_params(axis='y', labelcolor=color)
ax2.tick_params(axis='y', labelsize=12)

ax3 = ax1.twinx()
ax3.spines['right'].set_position(('outward', 60))
color = 'red'
ax3.set_ylabel('Tadle', color=color, fontsize=14, fontweight='bold')
line3, = ax3.plot(df['DATE'], df['index'], color=color, label='Tadle', linewidth=2)
ax3.tick_params(axis='y', labelcolor=color)
ax3.tick_params(axis='y', labelsize=12)

fig.tight_layout()
fig.legend(handles=[line2, line1, line3], loc='upper left', fontsize=12, borderaxespad=0., bbox_to_anchor=

plt.title("MP - Statement Tones and Fed Funds Effective Rate Over Time", fontsize=20)

plt.show()
```



Correlation is pretty low as 5.4%

JL - Alternative 2

Instead of using the keywords list, I used BERT model to do the sentiment analysis on the FOMC statements, this reduce the dependence on the keyword list and avoid any subjective judegement when doing the sentiment analysis. Another good thing is that it can be easily scaled yet still maintain the similar level of performance.

One alternative way could be to TextBlob identify the sentiment of text whether hawkish/ dovish.

Implementation -

1. Use the FOMC statements 'statements_df' and label as hawkish or dovish. We could do it manually or by using an existing dataset like Tadle (2022).

2. Train an NLP model to identify the sentiment of text. We could use any NLP model like BERT and RoBERTa.

3. Use the model to score the sentiment of new FOMC statements. A higher score can mean a more hawkish tone, and lower score means a more dovish tone.

Address shortcomings in the Tadle (2022) method

1. Tadle uses frequency of words to generate hawkish / dovish sentiment. But any sentence can be either based on context. Eg. "The Committee will continue to monitor inflation and act as appropriate" could be hawkish or dovish depending on context.

2. Tadle does not consider the sequence of words. Eg. the statements "The Committee believes that inflation is likely to remain elevated in the near term" and "The Committee is concerned about the risk of inflation remaining elevated in the near term" mean the same thing, but the second statement is kind of hawkish cos it shows Committee is concerned.

The NLP approach addresses these shortcomings by considering the meaning of the entire statement, instead of merely the frequency words /phrases.

Correlation

The correlation between the Tadle and the NLP method is likely to be high. But we can expect the NLP method to be more accurate, esp for statements that are ambiguous or use new language that is not defined in the vocabulary list.

In [ ]:
```
statements_df2 = pd.read_excel('fomc_statements.xlsx')
keywords = pd.read_excel('Tadle2022_keywords.xlsx')
dff = pd.read_csv('DFF.csv')
dff['DATE']=pd.to_datetime(dff['DATE'])
```

In [ ]:
```
hawkish_keywords = set(keywords['hawkish'].dropna())
dovish_keywords = set(keywords['dovish'].dropna())
positive_keywords = set(keywords['positive'].dropna())
negative_keywords = set(keywords['negative'].dropna())
negation_terms = set(keywords['negation'].dropna())
```

In [ ]:
```
def process_text(text):
    # Tokenize the text into sentences
    sentences = sent_tokenize(text)
```

```
    # Define punctuation to be removed
    punc = set(string.punctuation)

    processed_sentences = []
    for sent in sentences:
        # Remove punctuation and convert to lowercase
        cleaned_text = ''.join(char for char in sent if char not in punc).lower()
        # Remove all sentences that do not contain any keywords (defined as those from the hawkish or dovis
        if any(word in cleaned_text.split() for word in hawkish_keywords or dovish_keywords):
            processed_sentences.append(cleaned_text)
    processed_text = '.'.join(processed_sentences)
    return processed_text
```

In [ ]:

```
# Initialize the sentiment-analysis pipeline
sentiment_pipeline = pipeline("sentiment-analysis")

def sentiment_score(statement):
    try:
        result = sentiment_pipeline(statement)[0]
        label = result["label"]
        score = result["score"]

        if label == "POSITIVE":
            return score
        elif label == "NEGATIVE":
            return -score

    except Exception as e:
        score = 0

def analyze_sentence_2(sentence):
    lines = sentence.split('.')
    sent_score = 0

    for line in lines:
        score = sentiment_score(line)
        sent_score += score

    return sent_score

statements_df2['processed_text'] = statements_df2['full_text'].apply(process_text)
```

```
statements_df2['sent_score_BERT'] = statements_df2['processed_text'].apply(analyze_sentence_2)

statements_df2['sent_score_BERT'].describe()
```

```
No model was supplied, defaulted to distilbert-base-uncased-finetuned-sst-2-english and revision af0f99b
(https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english).
Using a pipeline without specifying a model name and revision in production is not recommended.
```

Out[ ]:
```
count    194.000000
mean      -1.239476
std        2.332129
min       -7.650847
25%       -2.897442
50%       -1.506407
75%        0.288382
max        6.666866
Name: sent_score_BERT, dtype: float64
```

In [ ]:
```
statements_df2
```

Out[ ]:

	Unnamed: 0	date	full_text	processed_text	sent_score_BERT
0	0	2000-02-02	The Federal Open Market Committee voted today ...	the federal open market committee voted today ...	-3.772988
1	1	2000-03-21	The Federal Open Market Committee voted today ...	the federal open market committee voted today ...	-3.815530
2	2	2000-05-16	The Federal Open Market Committee voted today ...	the federal open market committee voted today ...	-2.826711
3	3	2000-06-28	The Federal Open Market Committee at its meeti...	the federal open market committee at its meeti...	0.422014
4	4	2000-08-22	The Federal Open Market Committee at its meeti...	the federal open market committee at its meeti...	-1.925802
...	...	...	...	...	...
189	146	2023-03-22	Recent indicators point to modest growth in sp...	recent indicators point to modest growth in sp...	3.615172
190	147	2023-05-03	Economic activity expanded at a modest pace in...	economic activity expanded at a modest pace in...	2.620852
191	148	2023-06-14	Recent indicators suggest that economic activi...	recent indicators suggest that economic activi...	3.767323
192	149	2023-07-26	Recent indicators suggest that economic activi...	recent indicators suggest that economic activi...	2.715870
193	150	2023-09-20	Recent indicators suggest that economic activi...	recent indicators suggest that economic activi...	2.807825

194 rows × 5 columns

In [ ]:
```
statements_df.reset_index(inplace=True)
```

In [ ]:
```
statements_df = statements_df.merge(statements_df2[['date','sent_score_BERT']],left_on='date', right_on='da
```

In [ ]:
```
statements_df
```

Out[ ]:

	date	full_text	count	sentences	scores	index	fed_rate	sent_score_BERT
0	2000-02-02	The Federal Open Market Committee voted today ...	187	[the federal open market committee voted today...	[1, 1, 0, 1]	75.000000	5.64	-3.772988
1	2000-03-21	The Federal Open Market Committee voted today ...	194	[the federal open market committee voted today...	[1, 0, 1, 1]	75.000000	5.81	-3.815530
2	2000-05-16	The Federal Open Market Committee voted today ...	184	[the federal open market committee voted today...	[1, 1, 1]	100.000000	6.13	-2.826711
3	2000-06-28	The Federal Open Market Committee at its meeti...	161	[the federal open market committee at its meet...	[0, 1, 1, 1]	75.000000	6.50	0.422014
4	2000-08-22	The Federal Open Market Committee at its meeti...	162	[the federal open market committee at its meet...	[1, 1]	100.000000	6.42	-1.925802
...	...	...	...	...	...	...	...	...
189	2023-03-22	Recent indicators point to modest growth in sp...	320	[recent indicators point to modest growth in s...	[0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0]	27.272727	4.58	3.615172
190	2023-05-03	Economic activity expanded at a modest pace in...	286	[economic activity expanded at a modest pace i...	[0, 1, 0, 0, 0, 1, 0, 0, 0, 0]	20.000000	4.83	2.620852
191	2023-06-14	Recent indicators suggest that economic activi...	296	[recent indicators suggest that economic activ...	[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]	18.181818	5.08	3.767323
192	2023-07-26	Recent indicators suggest that economic activi...	288	[recent indicators suggest that economic activ...	[1, 1, 0, 0, 0, 1, 0, 0, 0, 0]	30.000000	5.08	2.715870
193	2023-09-20	Recent indicators suggest that economic activi...	290	[recent indicators suggest that economic activ...	[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]	10.000000	5.33	2.807825

194 rows × 8 columns

In [ ]:

```
df = dff.merge(statements_df, left_on='DATE', right_on='date', how='inner')
df = df[['date','fed_rate','index','sent_score_BERT']]
df.columns = ['date','DFF','sent_score','sent_score_2']
df
```

Out[ ]:

	date	DFF	sent_score	sent_score_2
0	2000-02-02	5.64	75.000000	-3.772988
1	2000-03-21	5.81	75.000000	-3.815530
2	2000-05-16	6.13	100.000000	-2.826711
3	2000-06-28	6.50	75.000000	0.422014
4	2000-08-22	6.42	100.000000	-1.925802
...	...	...	...	...
189	2023-03-22	4.58	27.272727	3.615172
190	2023-05-03	4.83	20.000000	2.620852
191	2023-06-14	5.08	18.181818	3.767323
192	2023-07-26	5.08	30.000000	2.715870
193	2023-09-20	5.33	10.000000	2.807825

194 rows × 4 columns

In [ ]:

```
df.corr()
```

Out[ ]:

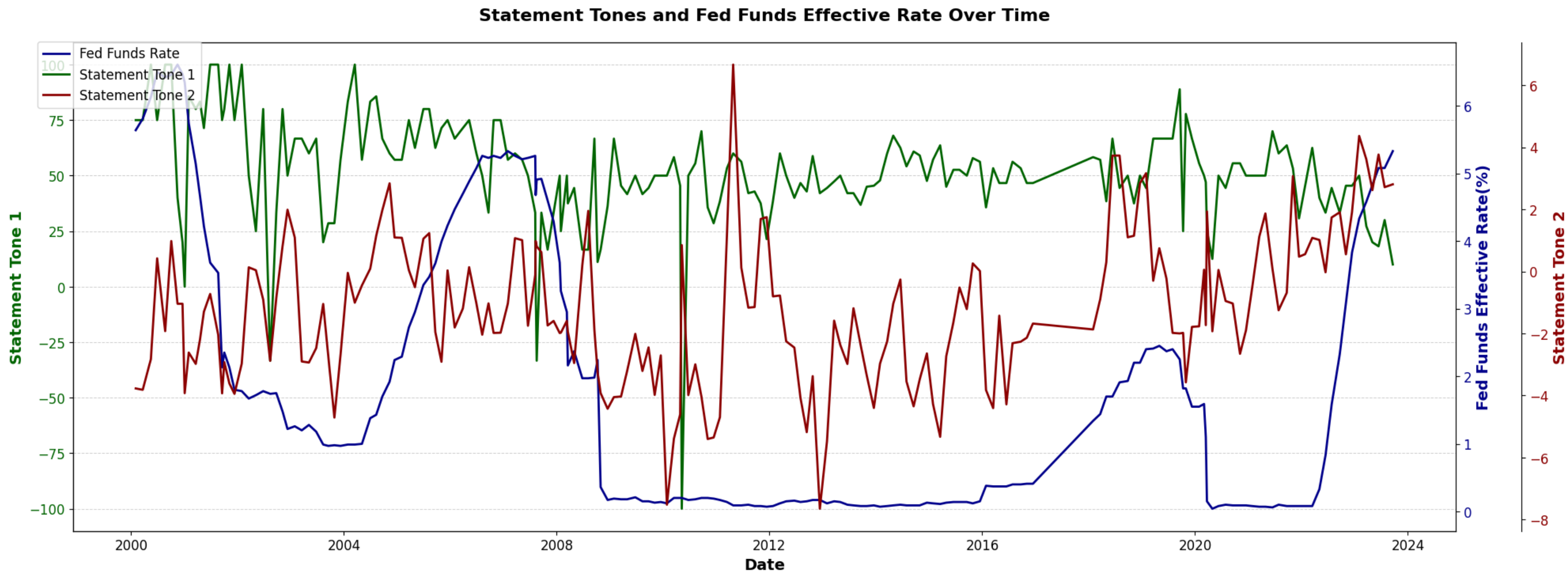
	DFF	sent_score	sent_score_2
DFF	1.000000	0.116716	0.236181
sent_score	0.116716	1.000000	-0.088290
sent_score_2	0.236181	-0.088290	1.000000

This measure is slightly negative correlated with Tadle measure, however, it has a better correlation with the fed rate

In [ ]:

```
fig, ax1 = plt.subplots(figsize=(20, 7))

color = 'darkgreen'
ax1.set_xlabel('Date', fontsize=14, fontweight='bold')
ax1.set_ylabel('Statement Tone 1', color=color, fontsize=14, fontweight='bold')
line1, = ax1.plot(df['date'], df['sent_score'], color=color, label='Statement Tone 1', linewidth=2)
ax1.tick_params(axis='y', labelcolor=color)
```

```
ax1.tick_params(axis='x', labelsize=12)
ax1.tick_params(axis='y', labelsize=12)
ax1.grid(True, axis='y', linestyle='--', linewidth=0.7, alpha=0.6)

ax2 = ax1.twinx()
color = 'darkblue'
ax2.set_ylabel('Fed Funds Effective Rate(%)', color=color, fontsize=14, fontweight='bold')
line2, = ax2.plot(df['date'], df['DFF'], color=color, label='Fed Funds Rate', linewidth=2)
ax2.tick_params(axis='y', labelcolor=color)
ax2.tick_params(axis='y', labelsize=12)

ax3 = ax1.twinx()
ax3.spines['right'].set_position(('outward', 60))
color = 'darkred'
ax3.set_ylabel('Statement Tone 2', color=color, fontsize=14, fontweight='bold')
line3, = ax3.plot(df['date'], df['sent_score_2'], color=color, label='Statement Tone 2', linewidth=2)
ax3.tick_params(axis='y', labelcolor=color)
ax3.tick_params(axis='y', labelsize=12)

fig.tight_layout()
fig.legend(handles=[line2, line1, line3], loc='upper left', fontsize=12, borderaxespad=0., bbox_to_anchor=

plt.title("Statement Tones and Fed Funds Effective Rate Over Time", fontsize=16, fontweight='bold', pad=20

plt.show()
```



1.6 Complete this problem twice: once with the Tadle (2022) measure of hawkishness and once with your measure. The steps will guide you through using the Fama and MacBeth (1973) procedure to estimate the monetary policy risk premium in industry returns data.4 In particular:

(a) Estimate an AR(1) model from the hawkishness data. Compute the residual. We will call this the “text-based monetary policy shock.”

In [ ]:
```
model = AutoReg(df['sent_score'], lags=1).fit()

print(model.summary())
```

```
                            AutoReg Model Results
==============================================================================
Dep. Variable:             sent_score   No. Observations:                  194
Model:                     AutoReg(1)   Log Likelihood                -873.389
Method:               Conditional MLE   S.D. of innovations             22.340
Date:                Wed, 25 Oct 2023   AIC                           1752.779
Time:                        17:21:25   BIC                           1762.567
Sample:                             1   HQIC                          1756.743
                                  194
=================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------
const            31.3404      3.801      8.246      0.000      23.891      38.790
sent_score.L1     0.3853      0.067      5.766      0.000       0.254       0.516
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            2.5954           +0.0000j            2.5954            0.0000
-----------------------------------------------------------------------------
```

In [ ]:
```
residuals = model.resid
df['rsd_1'] = residuals
df.dropna(inplace=True)
```

(b) Use the value-weighted returns from the daily industry returns file for this problem. For each of the 49 industries, regress returns on the day of the Fed announcement on the text-based monetary policy shock. Create a table with three columns: column 1 has the industry name, column 2 has the OLS regression coefficient and column 3 has the p-value for that coefficient. Sort the table from largest to smallest coefficient. To be concrete, the regression you are running at this stage is: Rit =αi +βiΔHAWKt +εit for each of the 49 industries, indexed by i. Time t here indexes Fed announcement days and ΔHAWKt is the text-based monetary policy shock at time t.

In [ ]:
```
port_df = pd.read_csv('49_Industry_Portfolios_Daily.CSV', skiprows=9)

def process_table(data):
    data = data.rename(columns ={'Unnamed: 0':'date'})
    data['date'] = pd.to_datetime(data['date'].astype('str'), format='%Y%m%d')
    data = data.replace([-99.99, -999], float('nan'))
    return data
```

In [ ]:
```
ret_D = process_table(port_df.iloc[:25543])
```

In [ ]:
```
ret_D
```

Out[ ]:

	date	Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books	Hshld	...	Boxes	Trans	Whlsl	Rtail	Meals	Banks	Insu
0	1926-07-01	0.56	-0.07	NaN	-1.39	0.0	-1.44	0.62	-1.27	-0.9	...	-0.93	0.14	2.77	-0.02	0.27	0.59	0.4
1	1926-07-02	0.29	0.06	NaN	0.78	0.7	1.46	0.03	0.0	-0.34	...	1.07	0.07	0.0	0.01	-0.1	1.04	-0.0
2	1926-07-06	-0.33	0.18	NaN	-1.74	0.5	-0.96	-0.06	4.27	-1.2	...	0.73	-0.2	0.77	-0.22	-0.67	0.45	0.3
3	1926-07-07	3.57	-0.15	NaN	-1.73	-0.12	-0.49	-0.06	-4.1	-0.22	...	2.22	0.18	-3.21	-0.57	-1.08	-1.09	0.2
4	1926-07-08	0.3	1.12	NaN	-0.15	0.3	-0.49	0.24	0.0	-0.01	...	-0.39	0.46	-1.1	-0.38	0.33	0.99	-0.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
25538	2023-07-25	-0.65	0.26	-0.31	0.33	-0.39	-1.06	0.15	0.10	0.60	...	4.06	-0.49	0.36	0.06	-0.19	-1.14	0.0
25539	2023-07-26	-0.62	-0.33	1.04	0.20	0.77	1.70	-0.11	0.51	-0.10	...	-1.53	2.60	-0.49	0.07	-0.06	1.09	0.0
25540	2023-07-27	-0.47	-0.69	-0.97	-1.37	0.26	-1.84	-1.86	-0.30	-1.58	...	-0.22	-1.14	-1.13	-0.28	-0.61	-1.41	-0.4
25541	2023-07-28	1.10	1.17	0.33	1.13	0.76	0.39	1.58	0.19	1.75	...	0.96	1.53	0.31	1.61	0.58	0.60	-0.3
25542	2023-07-31	0.51	-0.58	-0.98	-1.12	-0.16	3.03	1.62	1.53	-0.27	...	-0.28	0.08	0.36	0.59	0.48	0.60	0.3

25543 rows × 50 columns

In [ ]:
```
df
```

Out[ ]:

	date	DFF	sent_score	sent_score_2	rsd_1
1	2000-03-21	5.81	75.000000	-3.815530	14.761809
2	2000-05-16	6.13	100.000000	-2.826711	39.761809
3	2000-06-28	6.50	75.000000	0.422014	5.129217
4	2000-08-22	6.42	100.000000	-1.925802	39.761809
5	2000-10-03	6.46	100.000000	0.978715	30.129217
...	...	...	...	...	...
189	2023-03-22	4.58	27.272727	3.615172	-23.332871
190	2023-05-03	4.83	20.000000	2.620852	-21.848696
191	2023-06-14	5.08	18.181818	3.767323	-20.864669
192	2023-07-26	5.08	30.000000	2.715870	-8.345935
193	2023-09-20	5.33	10.000000	2.807825	-32.899525

193 rows × 5 columns

In [ ]:
```
ind_list = ret_D.columns[1:]
```

In [ ]:
```
ind_list
```

Out[ ]:
```
Index(['Agric', 'Food ', 'Soda ', 'Beer ', 'Smoke', 'Toys ', 'Fun  ', 'Books',
       'Hshld', 'Clths', 'Hlth ', 'MedEq', 'Drugs', 'Chems', 'Rubbr', 'Txtls',
       'BldMt', 'Cnstr', 'Steel', 'FabPr', 'Mach ', 'ElcEq', 'Autos', 'Aero ',
       'Ships', 'Guns ', 'Gold ', 'Mines', 'Coal ', 'Oil  ', 'Util ', 'Telcm',
       'PerSv', 'BusSv', 'Hardw', 'Softw', 'Chips', 'LabEq', 'Paper', 'Boxes',
       'Trans', 'Whlsl', 'Rtail', 'Meals', 'Banks', 'Insur', 'RlEst', 'Fin  ',
       'Other'],
      dtype='object')
```

In [ ]:
```
df_ind = df.merge(ret_D, left_on='date', right_on='date', how='inner')
```

In [ ]:
```
df_ind
```

Out[ ]:

	date	DFF	sent_score	sent_score_2	rsd_1	Agric	Food	Soda	Beer	Smoke	...	Boxes	Trans	Whlsl	Rtail	Me
0	2000-03-21	5.81	75.000000	-3.815530	14.761809	1.05	0.53	0.11	1.97	1.89	...	-0.40	2.28	1.94	2.19	1
1	2000-05-16	6.13	100.000000	-2.826711	39.761809	1.56	0.36	-19.22	-3.39	1.71	...	1.39	0.70	1.51	0.21	2
2	2000-06-28	6.50	75.000000	0.422014	5.129217	1.95	-0.58	-0.86	2.12	-5.40	...	0.66	2.03	0.79	-0.93	-0
3	2000-08-22	6.42	100.000000	-1.925802	39.761809	1.70	-1.15	-0.33	-1.45	-0.64	...	0.08	0.08	0.60	1.42	-0
4	2000-10-03	6.46	100.000000	0.978715	30.129217	1.39	-0.92	-0.16	0.04	1.44	...	0.32	1.33	0.01	0.04	-0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
185	2023-02-01	4.33	50.000000	4.366642	1.145782	0.16	0.38	0.07	0.53	2.59	...	-1.10	2.33	0.80	1.66	0
186	2023-03-22	4.58	27.272727	3.615172	-23.332871	-1.79	-0.96	-0.64	-0.80	-2.19	...	-0.62	-2.12	-1.84	-1.62	-1
187	2023-05-03	4.83	20.000000	2.620852	-21.848696	-0.91	0.14	-0.11	-0.22	-1.19	...	-0.81	0.25	-0.39	-0.43	-2
188	2023-06-14	5.08	18.181818	3.767323	-20.864669	-1.00	0.16	0.46	0.60	-0.86	...	-0.72	1.13	-0.66	-0.03	0
189	2023-07-26	5.08	30.000000	2.715870	-8.345935	-0.62	-0.33	1.04	0.20	0.77	...	-1.53	2.60	-0.49	0.07	-0

190 rows × 54 columns

In [ ]:

```
def industry_regression(df, rsd):
    results = []

    for industry in ind_list:
        X = df[rsd]
        X = sm.add_constant(X)
        X = X.apply(pd.to_numeric, errors='coerce')
```

```
        y = df[industry]
        y = y.apply(pd.to_numeric, errors='coerce')

        model = sm.OLS(y, X).fit()

        coef = model.params[rsd]
        p_value = model.pvalues[rsd]

        results.append([industry, coef, p_value])

    results_df = pd.DataFrame(results, columns=["Industry", "Coefficient", "P-value"])

    return results_df
```

(c) Comment on the ordering of the industries. Is it in line with what you would have expected?

```
For regression between Tald measure text_based monetary policy shock and industry returns
```

In [ ]:
```
res_1 = industry_regression(df_ind, 'rsd_1')
```

In [ ]:
```
res_1.sort_values('Coefficient')
```

Out[ ]:

	Industry	Coefficient	P-value
34	Hardw	-0.017881	0.050851
19	FabPr	-0.015653	0.080323
26	Gold	-0.015607	0.154194
28	Coal	-0.012979	0.263405
2	Soda	-0.012145	0.098837
47	Fin	-0.011352	0.249021
35	Softw	-0.010544	0.183258
39	Boxes	-0.010358	0.111064
27	Mines	-0.008300	0.333916
31	Telcm	-0.007271	0.200296
15	Txtls	-0.007231	0.372639
36	Chips	-0.007228	0.421202
20	Mach	-0.006710	0.341054
32	PerSv	-0.006148	0.330867
6	Fun	-0.005796	0.574573
22	Autos	-0.005579	0.499449
29	Oil	-0.004993	0.472504
18	Steel	-0.004839	0.591283
14	Rubbr	-0.003943	0.531703
42	Rtail	-0.003878	0.527089
7	Books	-0.003828	0.517324
16	BldMt	-0.003032	0.647005
33	BusSv	-0.002670	0.630576
9	Clths	-0.002370	0.737587
46	RlEst	-0.001917	0.816057

	Industry	Coefficient	P-value
41	Whlsl	-0.001756	0.733326
44	Banks	-0.001695	0.850451
40	Trans	-0.001645	0.813148
10	Hlth	-0.001518	0.756859
5	Toys	-0.001381	0.849200
3	Beer	-0.001196	0.767852
4	Smoke	-0.001155	0.845659
38	Paper	-0.000702	0.899136
13	Chems	-0.000640	0.923776
48	Other	-0.000314	0.961833
11	MedEq	-0.000231	0.963614
21	ElcEq	-0.000202	0.978687
0	Agric	-0.000113	0.987358
37	LabEq	0.001225	0.858827
43	Meals	0.001785	0.757739
8	Hshld	0.001837	0.674592
1	Food	0.002152	0.587144
30	Util	0.002356	0.631210
24	Ships	0.002966	0.665414
17	Cnstr	0.003014	0.722965
45	Insur	0.005169	0.449420
25	Guns	0.006429	0.316453
12	Drugs	0.008491	0.054600
23	Aero	0.013167	0.112856

The ordering of industries, primarily leaning towards negative coefficients, aligns with the expectation that hawkish monetary policies usually dampen investment returns. However, the general lack of statistical significance (high p-values) across these results suggests caution in drawing firm conclusions, highlighting the need for further analysis, potentially with a larger dataset or additional control variables to parse out these relationships more clearly.

(d) Now turn to the monthly returns data.6 Again use the value-weighted returns. Separately for each month, regress returns of each industry on its "beta" from step (b). To be concrete, for each month, indexed by T, you are running the following regression:

In [ ]:
```
port_df2 = pd.read_csv('49_Industry_Portfolios.CSV', skiprows=11)

def process_table(data):
    data = data.rename(columns ={'Unnamed: 0':'date'})
    data['date'] = pd.to_datetime(data['date'].astype('str'), format='%Y%m')
    data = data.replace([-99.99, -999], float('nan'))
    return data

ret_M = process_table(port_df2.iloc[:1154])
```

In [ ]:
```
ret_M
```

Out[ ]:

	date	Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books	Hshld	...	Boxes	Trans	Whlsl	Rtail	Meals	Banks
0	1926-07-01	2.37	0.12	-99.99	-5.19	1.29	8.65	2.50	50.21	-0.48	...	7.70	1.92	-23.79	0.07	1.87	4.61
1	1926-08-01	2.23	2.68	-99.99	27.03	6.50	16.81	-0.76	42.98	-3.58	...	-2.38	4.85	5.39	-0.75	-0.13	11.83
2	1926-09-01	-0.57	1.58	-99.99	4.02	1.26	8.33	6.42	-4.91	0.73	...	-5.54	0.08	-7.87	0.25	-0.56	-1.75
3	1926-10-01	-0.46	-3.68	-99.99	-3.31	1.06	-1.40	-5.09	5.37	-4.68	...	-5.08	-2.62	-15.38	-2.20	-4.11	-11.82
4	1926-11-01	6.75	6.26	-99.99	7.29	4.55	0.00	1.82	-6.40	-0.54	...	3.84	1.61	4.67	6.52	4.33	-2.97
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1149	2022-04-01	-0.14	2.60	4.22	3.03	6.37	-13.74	-27.84	-10.86	2.04	...	-3.93	-10.93	-2.14	-11.41	-5.47	-8.53
1150	2022-05-01	7.29	-3.26	-0.22	-1.60	2.67	-0.85	-3.50	-6.95	-5.12	...	-4.57	-4.59	1.03	-5.64	-3.29	3.41
1151	2022-06-01	-12.45	-1.91	0.46	-0.02	-11.63	-12.96	-10.87	-12.37	-2.56	...	-8.44	-7.14	-6.43	-8.50	-9.02	-12.39
1152	2022-07-01	6.38	3.68	3.28	5.49	0.56	5.63	17.04	12.08	0.76	...	7.09	9.33	9.08	16.33	11.89	8.54
1153	2022-08-01	5.23	-0.46	-4.40	-1.87	-0.12	-5.77	-2.26	-5.00	-2.16	...	-9.11	-1.46	-1.60	-3.46	-1.47	-3.41

1154 rows × 50 columns

In [ ]:

```
df
```

Out[ ]:

	date	DFF	sent_score	sent_score_2	rsd_1
1	2000-03-21	5.81	75.000000	-3.815530	14.761809
2	2000-05-16	6.13	100.000000	-2.826711	39.761809
3	2000-06-28	6.50	75.000000	0.422014	5.129217
4	2000-08-22	6.42	100.000000	-1.925802	39.761809
5	2000-10-03	6.46	100.000000	0.978715	30.129217
...	...	...	...	...	...
189	2023-03-22	4.58	27.272727	3.615172	-23.332871
190	2023-05-03	4.83	20.000000	2.620852	-21.848696
191	2023-06-14	5.08	18.181818	3.767323	-20.864669
192	2023-07-26	5.08	30.000000	2.715870	-8.345935
193	2023-09-20	5.33	10.000000	2.807825	-32.899525

193 rows × 5 columns

In [ ]:

```
ret_M['year'] =ret_M['date'].dt.year
ret_M['month'] =ret_M['date'].dt.month

df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
df_ind2 = pd.merge(ret_M, df,  how='inner', left_on=['year','month'], right_on = ['year','month'])
df_ind_M = df_ind2.set_index('date_x')
```

(f) What is the standard deviation of λ? Use this to compute the t-statistic. Is the risk premium significantly different from zero at the 10%, 5% or 1% level?

In [ ]:

```
df_ind2
```

Out[ ]:

	date_x	Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books	Hshld	...	RlEst	Fin	Other	year	month	date_y
0	2000-03-01	4.26	10.76	-0.29	0.13	5.11	7.69	10.70	13.12	-14.26	...	3.28	14.71	-2.00	2000	3	2000-03-21
1	2000-05-01	-2.47	18.06	-7.24	11.87	19.51	1.15	1.26	-6.34	2.79	...	-2.71	-6.81	10.45	2000	5	2000-05-16
2	2000-06-01	-0.94	2.49	1.06	6.05	2.96	-1.68	0.29	-0.03	-2.53	...	-1.06	14.51	2.30	2000	6	2000-06-28
3	2000-08-01	-1.94	-2.71	-2.22	-10.84	19.12	2.11	5.93	2.52	3.50	...	0.07	19.35	13.65	2000	8	2000-08-22
4	2000-10-01	-10.98	6.19	13.14	8.99	22.87	6.62	-5.79	-1.30	9.98	...	0.18	-5.77	-5.31	2000	10	2000-10-03
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
179	2022-01-01	1.36	1.28	0.12	-1.95	7.88	-16.47	-19.08	-5.62	-6.53	...	-7.93	-3.66	1.44	2022	1	2022-01-26
180	2022-03-01	10.78	1.27	-0.67	3.34	-2.25	-7.51	-4.14	0.63	-3.25	...	-1.81	-0.11	8.55	2022	3	2022-03-16
181	2022-05-01	7.29	-3.26	-0.22	-1.60	2.67	-0.85	-3.50	-6.95	-5.12	...	-2.34	3.95	-1.44	2022	5	2022-05-04
182	2022-06-01	-12.45	-1.91	0.46	-0.02	-11.63	-12.96	-10.87	-12.37	-2.56	...	-13.60	-9.79	-12.67	2022	6	2022-06-15
183	2022-07-01	6.38	3.68	3.28	5.49	0.56	5.63	17.04	12.08	0.76	...	14.10	10.44	10.01	2022	7	2022-07-27

184 rows × 57 columns

In [ ]:

```
df_ind_M = df_ind_M[ind_list]
```

In [ ]:

```
res_1.set_index('Industry', inplace=True)
```

In [ ]:

```
def industry_regression_beta(df, betas, col):
    results = []

    for i, row in df[ind_list].iterrows():
        X = betas
        X = sm.add_constant(X)
```

```
        X = X.apply(pd.to_numeric, errors='coerce')

        y = row
        y = y.apply(pd.to_numeric, errors='coerce')

        model = sm.OLS(y, X).fit()

        coef = model.params[col]
        p_value = model.pvalues[col]

        results.append([i, coef, p_value])

    results_df = pd.DataFrame(results, columns=["year-month", "Coefficient", "P-value"])

    return results_df
```

In [ ]:
```
res_M1 = industry_regression_beta(df_ind_M, res_1['Coefficient'], 'Coefficient')
```

In [ ]:
```
res_M1
```

Out[ ]:

	year-month	Coefficient	P-value
0	2000-03-01	230.395991	0.201539
1	2000-05-01	568.882275	0.000717
2	2000-06-01	-374.574747	0.027269
3	2000-08-01	-220.221743	0.213193
4	2000-10-01	557.810757	0.002378
...	...	...	...
179	2022-01-01	140.398095	0.397691
180	2022-03-01	-345.727020	0.031338
181	2022-05-01	70.714659	0.519712
182	2022-06-01	218.992569	0.048122
183	2022-07-01	8.042951	0.963885

184 rows × 3 columns

In [ ]:

```
res_M1['Coefficient'].describe()
```

Out[ ]:

```
count    184.000000
mean      -0.158804
std      203.164786
min     -719.117557
25%     -107.805437
50%        1.789165
75%      119.062618
max      568.882275
Name: Coefficient, dtype: float64
```

(e) What is the average λ across all months? This is the risk premium associated with holding assets exposed to monetary policy risk. Comment on how its sign can be interpreted.

In [ ]:

```
lambda_mean = res_M1['Coefficient'].describe()[1]
print(f"avg of λ: {lambda_mean.round(3)}")
```

```
avg of λ: -0.159
```

A λ of -0.159 suggests slight a inverse relationship between the industry returns and hawkish monetary policy shocks on average. This means that for a unit increase in the hawkishness of monetary policy, the industry's returns decrease by approximately the 15% of amount.

(f) What is the standard deviation of λ? Use this to compute the t-statistic. Is the risk premium significantly different from zero at the 10%, 5% or 1% level?

In [ ]:
```
lambda_std = res_M1['Coefficient'].describe()[2]
print(f"std of λ: {lambda_std.round(3)}")
```

```
std of λ: 203.165
```

In [ ]:
```
standard_error = lambda_std / 184**0.5

t_statistic = lambda_mean / standard_error

print("t-statistic:", t_statistic.round(3))
```

```
t-statistic: -0.011
```

In [ ]:
```
print(f'10% level critical t-value:{stats.t.ppf(1 - (0.1/2), 183).round(3)}')
print(f'5% level critical t-value:{stats.t.ppf(1 - (0.05/2), 183).round(3)}')
print(f'1% level critical t-value:{stats.t.ppf(1 - (0.01/2), 183).round(3)}')
```

```
10% level critical t-value:1.653
5% level critical t-value:1.973
1% level critical t-value:2.603
```

Thus the risk premium is not significantly different from zero at all levels.

1.6 Redo a to f with my measure

In [ ]:
```
model2 = AutoReg(df['sent_score_2'], lags=1).fit()

print(model2.summary())
```

```
                            AutoReg Model Results
==============================================================================
Dep. Variable:            sent_score_2   No. Observations:                  193
Model:                      AutoReg(1)   Log Likelihood                -377.612
Method:                Conditional MLE   S.D. of innovations              1.729
Date:                 Wed, 25 Oct 2023   AIC                            761.225
Time:                         17:29:07   BIC                            770.997
Sample:                              1   HQIC                           765.182
                                   193
===================================================================================
                      coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
const              -0.3760      0.142     -2.651      0.008      -0.654      -0.098
sent_score_2.L1     0.6709      0.054     12.432      0.000       0.565       0.777
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.4905           +0.0000j            1.4905            0.0000
-----------------------------------------------------------------------------
```

In [ ]:

```
residuals2 = model2.resid
df['rsd_2'] = residuals2
df.dropna(inplace=True)
```

For regression between my measure text_based monetary policy shock and industry returns

In [ ]:

```
df_ind = df.merge(ret_D, left_on='date', right_on='date', how='inner')
```

In [ ]:

```
df_ind
```

Out[ ]:

	date	DFF	sent_score	sent_score_2	rsd_1	year	month	rsd_2	Agric	Food	...	Boxes	Trans	Whlsl	Rtail
0	2000-05-16	6.13	100.000000	-2.826711	39.761809	2000	5	0.109208	1.56	0.36	...	1.39	0.70	1.51	0.21
1	2000-06-28	6.50	75.000000	0.422014	5.129217	2000	6	2.694507	1.95	-0.58	...	0.66	2.03	0.79	-0.93
2	2000-08-22	6.42	100.000000	-1.925802	39.761809	2000	8	-1.832967	1.70	-1.15	...	0.08	0.08	0.60	1.42
3	2000-10-03	6.46	100.000000	0.978715	30.129217	2000	10	2.646763	1.39	-0.92	...	0.32	1.33	0.01	0.04
4	2000-11-15	6.61	40.000000	-1.043032	-29.870783	2000	11	-1.323703	3.06	0.83	...	0.73	1.20	1.55	1.66
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
184	2023-02-01	4.33	50.000000	4.366642	1.145782	2023	2	3.460164	0.16	0.38	...	-1.10	2.33	0.80	1.66
185	2023-03-22	4.58	27.272727	3.615172	-23.332871	2023	3	1.061449	-1.79	-0.96	...	-0.62	-2.12	-1.84	-1.62
186	2023-05-03	4.83	20.000000	2.620852	-21.848696	2023	5	0.571310	-0.91	0.14	...	-0.81	0.25	-0.39	-0.43
187	2023-06-14	5.08	18.181818	3.767323	-20.864669	2023	6	2.384897	-1.00	0.16	...	-0.72	1.13	-0.66	-0.03
188	2023-07-26	5.08	30.000000	2.715870	-8.345935	2023	7	0.564245	-0.62	-0.33	...	-1.53	2.60	-0.49	0.07

189 rows × 57 columns

In [ ]:

```
res_2 = industry_regression(df_ind, 'rsd_2')
```

In [ ]:

```
res_2.sort_values('Coefficient')
```

Out[ ]:

	Industry	Coefficient	P-value
26	Gold	-0.221870	0.077327
28	Coal	-0.205268	0.123765
34	Hardw	-0.185924	0.077338
46	RlEst	-0.181296	0.054973
19	FabPr	-0.162666	0.114916
47	Fin	-0.153430	0.176026
44	Banks	-0.148576	0.147907
0	Agric	-0.114618	0.160737
29	Oil	-0.109026	0.170358
20	Mach	-0.108235	0.180874
16	BldMt	-0.093940	0.217511
17	Cnstr	-0.092992	0.341838
18	Steel	-0.092067	0.374903
27	Mines	-0.091866	0.353337
48	Other	-0.089714	0.233266
21	ElcEq	-0.079000	0.354403
32	PerSv	-0.069050	0.341686
37	LabEq	-0.061263	0.439757
35	Softw	-0.059403	0.514526
6	Fun	-0.059118	0.619364
36	Chips	-0.050729	0.624061
31	Telcm	-0.047049	0.472748
25	Guns	-0.045059	0.541500
7	Books	-0.042590	0.530969
39	Boxes	-0.042065	0.575493

	Industry	Coefficient	P-value
45	Insur	-0.041660	0.593200
13	Chems	-0.041327	0.591736
38	Paper	-0.037036	0.560746
14	Rubbr	-0.035948	0.620806
33	BusSv	-0.030767	0.630224
43	Meals	-0.011144	0.866902
24	Ships	-0.007449	0.924922
10	Hlth	-0.004267	0.939662
12	Drugs	-0.003724	0.941574
40	Trans	-0.001547	0.984566
1	Food	0.003152	0.944985
8	Hshld	0.013687	0.786088
30	Util	0.017713	0.754192
23	Aero	0.021063	0.826316
2	Soda	0.022443	0.791995
9	Clths	0.024444	0.763992
42	Rtail	0.026094	0.711137
11	MedEq	0.027300	0.639448
41	Whlsl	0.041017	0.487970
3	Beer	0.050165	0.277450
15	Txtls	0.055392	0.553681
22	Autos	0.056090	0.555565
4	Smoke	0.065922	0.332384
5	Toys	0.076282	0.361569

The sectors like 'Gold,' 'Coal,' 'Hardw,' and 'RlEst' show a much higher negative coefficient compared to the previous analysis, suggesting these sectors might be more sensitive to monetary policy shocks. The larger negative value indicates that returns in these sectors might decrease more significantly when there's a tightening monetary policy, such as increasing interest rates.

There's a noticeable shift in which sectors are most sensitive to monetary policy shocks. For instance, 'Gold' and 'Hardw' were sensitive in both analyses, but they appear to be more so now. Conversely, sectors like 'Fin' and 'Banks' have moved slightly down the list, though they still show considerable negative sensitivity.

In [ ]:
```
res_2.set_index('Industry', inplace=True)
```

In [ ]:
```
res_M2 = industry_regression_beta(df_ind_M, res_2['Coefficient'], 'Coefficient')
```

In [ ]:
```
res_M2
```

Out[ ]:

	year-month	Coefficient	P-value
0	2000-03-01	9.330915	0.537789
1	2000-05-01	27.186023	0.062980
2	2000-06-01	-29.845789	0.035330
3	2000-08-01	-39.779238	0.005322
4	2000-10-01	54.986350	0.000241
...	...	...	...
179	2022-01-01	-6.114268	0.659596
180	2022-03-01	-36.666580	0.005381
181	2022-05-01	-12.167678	0.180794
182	2022-06-01	21.904086	0.016685
183	2022-07-01	7.107603	0.630910

184 rows × 3 columns

In [ ]:
```
res_M2['Coefficient'].describe()
```

Out[ ]:
```
count    184.000000
mean       0.690021
std       18.370789
min      -52.264785
25%      -13.501718
50%        0.027503
75%       11.846412
max       56.312005
Name: Coefficient, dtype: float64
```

In [ ]:
```
print(f"avg of λ: {res_M2['Coefficient'].describe()[1].round(2)}")
print(f"std of λ: {res_M2['Coefficient'].describe()[2].round(2)}")
```

```
avg of λ: 0.69
std of λ: 18.37
```

A lambda (λ) of 0.69 indicates that there is a positive impact on industry returns when there's an unexpected hawkish monetary policy change.

In [ ]:
```
lambda_mean = res_M2['Coefficient'].describe()[1]
lambda_std = res_M2['Coefficient'].describe()[2]
```

In [ ]:
```
standard_error = lambda_std / 184**0.5

t_statistic = lambda_mean / standard_error

print("t-statistic:", t_statistic.round(3))
```

```
t-statistic: 0.509
```

In [ ]:
```
print(f'10% level critical t-value:{stats.t.ppf(1 - (0.1/2), 183).round(3)}')
print(f'5% level critical t-value:{stats.t.ppf(1 - (0.05/2), 183).round(3)}')
print(f'1% level critical t-value:{stats.t.ppf(1 - (0.01/2), 183).round(3)}')
```

```
10% level critical t-value:1.653
5% level critical t-value:1.973
1% level critical t-value:2.603
```

Thus the risk premium is not significantly different from zero at all levels.