Shea FA692-HW2

March 29, 2023

1 FA692 Homework 2

2 Due: Wednesday, March 29 @ 11:59PM

Name: Ryan Shea

Date: March 28, 2023

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Set seed of random number generator
CWID = 10445281 #Place here your Campus wide ID number, this will personalize
#your results, but still maintain the reproduceable nature of using seeds.
#If you ever need to reset the seed in this assignment, use this as your seed
#Papers that use -1 as this CWID variable will earn 0's so make sure you change
#this value before you submit your work.
personal = CWID % 10000
np.random.seed(personal)
```

2.1 Question 1 (10pt)

2.1.1 Question 1.1

Use the `yfinance` package (or other method of your choice) to obtain the daily adjusted close prices for the S&P500 (`SPY`) from January 1, 2023 to March 15, 2023. You should inspect the dates for your data to make sure you are including everything appropriately. Create a data frame (or array) of the daily log returns of this stock; you may concatenate this to your price data. Use the `print` command to display your data.

[ ]:
```
# Enter your code here
import yfinance as yf

spy = yf.download('SPY', start='2023-01-01', end='2023-03-15')

spy['log'] = np.log(spy['Adj Close'] / spy['Adj Close'].shift(1))
spy = spy[['Adj Close', 'log']].dropna()
print(spy)
```

```
[**********************100%**********************]  1 of 1 completed
             Adj Close       log
Date
2023-01-04  382.300964  0.007691
2023-01-05  377.937592 -0.011479
2023-01-06  386.604492  0.022673
2023-01-09  386.385345 -0.000567
2023-01-10  389.095001  0.006988
2023-01-11  394.016235  0.012569
2023-01-12  395.450745  0.003634
2023-01-13  396.984894  0.003872
2023-01-17  396.257660 -0.001834
2023-01-18  390.001556 -0.015914
2023-01-19  387.162415 -0.007306
2023-01-20  394.374878  0.018458
2023-01-23  399.106812  0.011927
2023-01-24  398.678436 -0.001074
2023-01-25  398.827881  0.000375
2023-01-26  403.211151  0.010930
2023-01-27  404.137604  0.002295
2023-01-30  399.066956 -0.012626
2023-01-31  404.934570  0.014596
2023-02-01  409.238129  0.010572
2023-02-02  415.195404  0.014452
2023-02-03  410.782257 -0.010686
2023-02-06  408.271820 -0.006130
2023-02-07  413.611450  0.012994
2023-02-08  409.088715 -0.010995
2023-02-09  405.542236 -0.008707
2023-02-10  406.488647  0.002331
2023-02-13  411.260406  0.011671
2023-02-14  411.071167 -0.000460
2023-02-15  412.406067  0.003242
2023-02-16  406.727722 -0.013864
2023-02-17  405.711609 -0.002501
2023-02-21  397.572662 -0.020265
2023-02-22  397.024750 -0.001379
2023-02-23  399.136688  0.005305
2023-02-24  394.872955 -0.010740
2023-02-27  396.217834  0.003400
2023-02-28  394.753418 -0.003703
2023-03-01  393.239197 -0.003843
2023-03-02  396.297516  0.007747
2023-03-03  402.653259  0.015911
2023-03-06  402.932220  0.000693
2023-03-07  396.755768 -0.015447
2023-03-08  397.403320  0.001631
2023-03-09  390.071289 -0.018622
```

```
2023-03-10  384.442780 -0.014535
2023-03-13  383.894836 -0.001426
2023-03-14  390.240662  0.016395

             Adj Close       log
Date
2023-01-04  382.300964  0.007691
2023-01-05  377.937592 -0.011479
2023-01-06  386.604492  0.022673
2023-01-09  386.385345 -0.000567
2023-01-10  389.095001  0.006988
2023-01-11  394.016235  0.012569
2023-01-12  395.450745  0.003634
2023-01-13  396.984894  0.003872
2023-01-17  396.257660 -0.001834
2023-01-18  390.001556 -0.015914
2023-01-19  387.162415 -0.007306
2023-01-20  394.374878  0.018458
2023-01-23  399.106812  0.011927
2023-01-24  398.678436 -0.001074
2023-01-25  398.827881  0.000375
2023-01-26  403.211151  0.010930
2023-01-27  404.137604  0.002295
2023-01-30  399.066956 -0.012626
2023-01-31  404.934570  0.014596
2023-02-01  409.238129  0.010572
2023-02-02  415.195404  0.014452
2023-02-03  410.782257 -0.010686
2023-02-06  408.271820 -0.006130
2023-02-07  413.611450  0.012994
2023-02-08  409.088715 -0.010995
2023-02-09  405.542236 -0.008707
2023-02-10  406.488647  0.002331
2023-02-13  411.260406  0.011671
2023-02-14  411.071167 -0.000460
2023-02-15  412.406067  0.003242
2023-02-16  406.727722 -0.013864
2023-02-17  405.711609 -0.002501
2023-02-21  397.572662 -0.020265
2023-02-22  397.024750 -0.001379
2023-02-23  399.136688  0.005305
2023-02-24  394.872955 -0.010740
2023-02-27  396.217834  0.003400
2023-02-28  394.753418 -0.003703
2023-03-01  393.239197 -0.003843
2023-03-02  396.297516  0.007747
2023-03-03  402.653259  0.015911
2023-03-06  402.932220  0.000693
```

```
2023-03-07  396.755768 -0.015447
2023-03-08  397.403320  0.001631
2023-03-09  390.071289 -0.018622
2023-03-10  384.442780 -0.014535
2023-03-13  383.894836 -0.001426
2023-03-14  390.240662  0.016395
```

2.2 Question 2 (40pt)

2.2.1 Question 2.1

Scrape data from the Bloomberg `@business` Twitter account from January 1, 2023 to March 15, 2023. Save this data to a Data Frame with time stamps. Additionally, save all the collected data to a text file with time stamps. You will need to submit the text file along with your work (-5 points if not submitted).

Note: Bloomberg tweets sometimes include the pipe "|". I recomment using tilde "~" as a delimiter instead.

Hint: Because saving the tweets can take a long time, you can comment that code out before exporting to pdf.

[ ]:
```
# Enter your code here
import snscrape.modules.twitter as tw

# f = open('business.txt', 'w', encoding='utf-8')

# for tweet in tw.TwitterSearchScraper(query="(from:business) since:2023-01-01␣
 ↪until:2023-03-15").get_items():
#     date_str = tweet.date.strftime("%Y-%m-%d %H:%M:%S%z")
#     date_str = date_str[:-2] + ":" + date_str[-2:]
#     #f.write(date_str + "|" + tweet.content + "\n")
#     f.write(date_str + "~" + tweet.rawContent + "\n")
# f.close()
```

[ ]:
```
from datetime import datetime as dt
import pytz

business = []
dates = []
f = open('business.txt', 'r', encoding='utf-8')

for l in f:
    line = l.split('~')
    date_str = line[0]
    try:
        date_time = dt.fromisoformat(date_str)
        date_time = date_time.astimezone(pytz.timezone("US/Eastern"))
        line[0] = date_time
```

```
        line[1] = line[1][:-1]
        business.append(line)
        dates.append(date_time.date())
    except:
        business[-1][1] += " "+l[:-1]
f.close()

business = pd.DataFrame(business, columns=['Time', 'Tweet'])
business['Date'] = dates

business
```

[ ]:

```
                          Time  \
0     2023-03-14 19:40:29-04:00
1     2023-03-14 19:40:29-04:00
2     2023-03-14 19:35:41-04:00
3     2023-03-14 19:31:07-04:00
4     2023-03-14 19:25:09-04:00
…                           …
26813 2022-12-31 19:00:09-05:00
26814 2022-12-31 19:00:09-05:00
26815 2022-12-31 19:00:09-05:00
26816 2022-12-31 19:00:09-05:00
26817 2022-12-31 19:00:08-05:00

                                                   Tweet        Date
0      One Japanese fintech firm is making it compuls…  2023-03-14
1      An unlikely startup guru has emerged in Japan,…  2023-03-14
2      Some US cities are late in making financial di…  2023-03-14
3      The shipping industry is looking to rethink ev…  2023-03-14
4      A biotech wants to cut fashion waste by using …  2023-03-14
…                                                    …          …
26813  Toymakers have found a new group of customers:…  2022-12-31
26814  Belarusian hackers and dissidents determined t…  2022-12-31
26815  Landlords are taking out millions in loans to …  2022-12-31
26816  It took a pandemic to make a dent in US inequa…  2022-12-31
26817  A planned train line in Mexico is billions ove…  2022-12-31

[26818 rows x 3 columns]
```

2.2.2 Question 2.2

Using your favorite sentiment analyzer (e.g., `vaderSentiment`), find the average sentiment for the headlines on each date that data was collected. Concatenate this sentiment score to your data frame of log returns. Use the `print` command to display your data.

[ ]:
```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

sentiment = []
analyzer = SentimentIntensityAnalyzer()
for tweet in business.Tweet:
    vs = analyzer.polarity_scores(tweet)
    sentiment.append(vs["compound"])

business['Sentiment'] = sentiment

spy['Sentiment'] = business.pivot_table(index='Date', values='Sentiment',␣
 ↪aggfunc='mean')
print(spy)
```

```
             Adj Close       log  Sentiment
Date
2023-01-04  382.300964  0.007691  -0.010883
2023-01-05  377.937592 -0.011479  -0.032084
2023-01-06  386.604492  0.022673  -0.001990
2023-01-09  386.385345 -0.000567   0.048762
2023-01-10  389.095001  0.006988  -0.051226
2023-01-11  394.016235  0.012569  -0.006794
2023-01-12  395.450745  0.003634   0.012535
2023-01-13  396.984894  0.003872  -0.017501
2023-01-17  396.257660 -0.001834   0.013547
2023-01-18  390.001556 -0.015914  -0.002899
2023-01-19  387.162415 -0.007306  -0.019422
2023-01-20  394.374878  0.018458  -0.052442
2023-01-23  399.106812  0.011927   0.030476
2023-01-24  398.678436 -0.001074  -0.031936
2023-01-25  398.827881  0.000375  -0.036038
2023-01-26  403.211151  0.010930  -0.000502
2023-01-27  404.137604  0.002295   0.011412
2023-01-30  399.066956 -0.012626   0.009724
2023-01-31  404.934570  0.014596   0.018422
2023-02-01  409.238129  0.010572   0.052543
2023-02-02  415.195404  0.014452   0.040621
2023-02-03  410.782257 -0.010686  -0.029209
2023-02-06  408.271820 -0.006130  -0.006613
2023-02-07  413.611450  0.012994  -0.010283
2023-02-08  409.088715 -0.010995   0.030015
2023-02-09  405.542236 -0.008707   0.025923
2023-02-10  406.488647  0.002331   0.049027
2023-02-13  411.260406  0.011671   0.037702
2023-02-14  411.071167 -0.000460   0.045768
2023-02-15  412.406067  0.003242  -0.009137
2023-02-16  406.727722 -0.013864   0.033467
```

```
2023-02-17  405.711609 -0.002501   0.001133
2023-02-21  397.572662 -0.020265   0.040426
2023-02-22  397.024750 -0.001379   0.037188
2023-02-23  399.136688  0.005305  -0.003498
2023-02-24  394.872955 -0.010740  -0.021110
2023-02-27  396.217834  0.003400   0.018866
2023-02-28  394.753418 -0.003703  -0.011442
2023-03-01  393.239197 -0.003843   0.061795
2023-03-02  396.297516  0.007747  -0.012462
2023-03-03  402.653259  0.015911   0.024269
2023-03-06  402.932220  0.000693   0.027701
2023-03-07  396.755768 -0.015447   0.026721
2023-03-08  397.403320  0.001631   0.052365
2023-03-09  390.071289 -0.018622  -0.001680
2023-03-10  384.442780 -0.014535  -0.054424
2023-03-13  383.894836 -0.001426  -0.098203
2023-03-14  390.240662  0.016395  -0.032399
```

2.3 Question 3 (10pt)

2.3.1 Question 3.1

Determine the correlation between `SPY` returns and `@business` headlines. Statistically test whether this correlation is significant or not. Comment on the results and how you may be able to improve them.

Hint: The standard error for the correlation coefficient ρ is given by $\sqrt{\frac{1-\rho^2}{N-2}}$ when using N data points.

[ ]:
```
# Enter your code here
print(f"Corr:           {spy['log'].corr(spy['Sentiment'])}")
st_error = np.sqrt((1 - spy['log'].corr(spy['Sentiment'])**2) / (spy.shape[0] -␣
↪2))
print(f"Standard Error: {st_error}")
```

```
Corr:           -0.009485990846686164
Standard Error: 0.14743532229551995
```

You can see that the correlation between spy and the headlines is -0.00948, which is very insignificant. The standard error is 0.14, so there is a relatively large difference between the "population correlation" and the sample correlation. This could be because there is only 48 samples which is technically statistically significant but it is still not a large enough sample to be confident with the results.