PROBLEM_A

February 4, 2023

0.1 Problem A

0.2 A1

[3]:
```
#import  depepndencies

import numpy as np
import pandas as pd
from pandas import Series, DataFrame
import matplotlib.pyplot as plt

import scipy
from scipy import stats

% matplotlib inline
```

```
UsageError: Line magic function `%` not found.
```

[4]:
```
# load the data

ford_model = pd.read_excel(r'/home/dsm/PART_A/FORD.csv')
tesla_model = pd.read_excel(r'/home/dsm/PART_A/TESLA.csv')

f = ford_model.head(252)
t = tesla_model.head(252)
```

[5]:
```
#print ford entries

print(f)
```

```
      Date       Open       High        Low      Close  Adj Close     Volume
0    44592  19.580000  20.330000  19.370001  20.299999  19.760141   91361900
1    44593  20.610001  20.850000  19.920000  20.660000  20.110567  117651800
2    44594  20.809999  21.049999  20.180000  20.629999  20.081366   95377600
3    44595  20.170000  20.620001  19.870001  19.889999  19.361044  103016000
4    44596  18.520000  18.590000  17.520000  17.959999  17.482370  211100500
..     …          …          …          …          …          …          …
247  44951  12.600000  12.850000  12.490000  12.790000  12.790000   37739000
248  44952  12.990000  13.070000  12.710000  12.920000  12.920000   48970900
```

```
249  44953  12.880000  13.370000  12.870000  13.270000  13.270000   62066500
250  44956  13.010000  13.200000  12.860000  12.890000  12.890000   64463300
251  44957  13.390000  13.570000  13.250000  13.445000  13.445000   50203979

[252 rows x 7 columns]
```

[6]:
```
#print tesla entries

print(t)
```

```
      Date        Open        High         Low       Close   Adj Close  \
0    44592  290.903320  312.663330  287.350006  312.239990  312.239990
1    44593  311.736664  314.566681  301.666656  310.416656  310.416656
2    44594  309.393341  310.500000  296.470001  301.886658  301.886658
3    44595  294.000000  312.333344  293.506653  297.046661  297.046661
4    44596  299.073334  312.166656  293.723328  307.773346  307.773346
..     ...         ...         ...         ...         ...         ...
247  44951  141.910004  146.410004  138.070007  144.429993  144.429993
248  44952  159.970001  161.419998  154.759995  160.270004  160.270004
249  44953  162.429993  180.679993  161.169998  177.899994  177.899994
250  44956  178.050003  179.770004  166.500000  166.660004  166.660004
251  44957  164.570007  173.748398  162.779999  171.285004  171.285004

        Volume
0    104436000
1     73138200
2     66792900
3     78855600
4     73625400
..         ...
247  192734300
248  234815100
249  305632100
250  230203200
251  143541028

[252 rows x 7 columns]
```

[40]:
```
# describe properties
f.describe()
```

[40]:
```
               Date        Open        High         Low       Close  \
count    252.000000  252.000000  252.000000  252.000000  252.000000
mean   44773.039683   14.114206   14.362817   13.856746   14.114702
std      105.714597    2.080363    2.114100    2.033197    2.070480
min    44592.000000   11.050000   11.210000   10.610000   10.950000
25%    44682.250000   12.375000   12.742500   12.187500   12.495000
```

```
50%    44774.500000  13.745000  13.875000  13.375000  13.655000
75%    44862.750000  15.577500  15.775000  15.332500  15.512500
max    44957.000000  20.809999  21.049999  20.180000  20.660000

        Adj Close        Volume
count  252.000000  2.520000e+02
mean    13.895848  6.544592e+07
std      1.962912  2.449133e+07
min     10.837501  1.298090e+07
25%     12.347353  5.077075e+07
50%     13.471792  5.956675e+07
75%     15.295121  7.603990e+07
max     20.110567  2.111005e+08
```

[41]:
```
# describe properties
t.describe()
```

[41]:
```
               Date        Open        High         Low       Close  \
count    252.000000  252.000000  252.000000  252.000000  252.000000
mean   44773.039683  247.776852  253.925483  240.908320  247.206872
std      105.714597   62.848254   63.835486   61.767233   62.741874
min    44592.000000  103.000000  111.750000  101.809998  108.099998
25%    44682.250000  209.692493  220.552502  205.705002  212.637505
50%    44774.500000  251.911667  257.413330  242.483330  251.758331
75%    44862.750000  296.651672  302.702499  288.214173  293.966667
max    44957.000000  378.766663  384.290009  362.433319  381.816681

        Adj Close        Volume
count  252.000000  2.520000e+02
mean   247.206872  9.464496e+07
std     62.741874  4.181675e+07
min    108.099998  4.186470e+07
25%    212.637505  6.702772e+07
50%    251.758331  8.411670e+07
75%    293.966667  1.025714e+08
max    381.816681  3.056321e+08
```

[8]:
```
f.sum()
```

[8]:
```
Date         1.128281e+07
Open         3.556780e+03
High         3.619430e+03
Low          3.491900e+03
Close        3.556905e+03
Adj Close    3.501754e+03
Volume       1.649237e+10
dtype: float64
```

[9]:
```
t.sum()
```

[9]:
```
Date         1.128281e+07
Open         6.243977e+04
High         6.398922e+04
Low          6.070890e+04
Close        6.229613e+04
Adj Close    6.229613e+04
Volume       2.385053e+10
dtype: float64
```

[10]:
```
f.sum(axis=1)
```

[10]:
```
0      9.140659e+07
1      1.176965e+08
2      9.542230e+07
3      1.030607e+08
4      2.111452e+08
           …
247    3.778401e+07
248    4.901592e+07
249    6.211152e+07
250    6.450832e+07
251    5.024900e+07
Length: 252, dtype: float64
```

[11]:
```
t.sum(axis=1)
```

[11]:
```
0      1.044821e+08
1      7.318434e+07
2      6.683901e+07
3      7.890169e+07
4      7.367152e+07
           …
247    1.927800e+08
248    2.348608e+08
249    3.056779e+08
250    2.302490e+08
251    1.435868e+08
Length: 252, dtype: float64
```

[12]:
```
# median
f.median()
```

[12]:
```
Date         4.477450e+04
Open         1.374500e+01
High         1.387500e+01
```

```
Low          1.337500e+01
Close        1.365500e+01
Adj Close    1.347179e+01
Volume       5.956675e+07
dtype: float64
```

[13]:
```python
# median
t.median()
```

[13]:
```
Date         4.477450e+04
Open         2.519117e+02
High         2.574133e+02
Low          2.424833e+02
Close        2.517583e+02
Adj Close    2.517583e+02
Volume       8.411670e+07
dtype: float64
```

[14]:
```python
# mean
f.mean()
```

[14]:
```
Date         4.477304e+04
Open         1.411421e+01
High         1.436282e+01
Low          1.385675e+01
Close        1.411470e+01
Adj Close    1.389585e+01
Volume       6.544592e+07
dtype: float64
```

[15]:
```python
# mean
f.mean()
```

[15]:
```
Date         4.477304e+04
Open         1.411421e+01
High         1.436282e+01
Low          1.385675e+01
Close        1.411470e+01
Adj Close    1.389585e+01
Volume       6.544592e+07
dtype: float64
```

[16]:
```python
# media
t.mean()
```

[16]:
```
Date         4.477304e+04
Open         2.477769e+02
```

```
High         2.539255e+02
Low          2.409083e+02
Close        2.472069e+02
Adj Close    2.472069e+02
Volume       9.464496e+07
dtype: float64
```

[17]:
```
# standard deviation
f.std()
```

[17]:
```
Date         1.057146e+02
Open         2.080363e+00
High         2.114100e+00
Low          2.033197e+00
Close        2.070480e+00
Adj Close    1.962912e+00
Volume       2.449133e+07
dtype: float64
```

[18]:
```
# standard deviation
t.std()
```

[18]:
```
Date         1.057146e+02
Open         6.284825e+01
High         6.383549e+01
Low          6.176723e+01
Close        6.274187e+01
Adj Close    6.274187e+01
Volume       4.181675e+07
dtype: float64
```

[19]:
```
# skewness
f.skew()
```

[19]:
```
Date        -0.000113
Open         0.627788
High         0.641640
Low          0.591521
Close        0.640540
Adj Close    0.582377
Volume       2.021585
dtype: float64
```

[20]:
```
# skewness
t.skew()
```

[20]:
```
Date         -0.000113
Open         -0.405063
High         -0.422329
Low          -0.388196
Close        -0.401201
Adj Close    -0.401201
Volume        1.862001
dtype: float64
```

[21]:
```
# Kurtosis
f.kurtosis()
```

[21]:
```
Date         -1.192528
Open         -0.155128
High         -0.132661
Low          -0.246695
Close        -0.110457
Adj Close    -0.123741
Volume        7.840863
dtype: float64
```

[22]:
```
# kurtosisi
t.kurtosis()
```

[22]:
```
Date         -1.192528
Open         -0.483477
High         -0.477637
Low          -0.499871
Close        -0.491002
Adj Close    -0.491002
Volume        3.985554
dtype: float64
```

[23]:
```
# correlation coeffient of ford
f.corr()
```

[23]:
```
               Date      Open      High       Low     Close  Adj Close  \
Date       1.000000 -0.641805 -0.650803 -0.625817 -0.634419  -0.590937
Open      -0.641805  1.000000  0.995655  0.994575  0.988481   0.986130
High      -0.650803  0.995655  1.000000  0.996067  0.995167   0.992172
Low       -0.625817  0.994575  0.996067  1.000000  0.995775   0.994726
Close     -0.634419  0.988481  0.995167  0.995775  1.000000   0.998316
Adj Close -0.590937  0.986130  0.992172  0.994726  0.998316   1.000000
Volume    -0.353874  0.310933  0.322564  0.267250  0.291683   0.276329

             Volume
Date      -0.353874
```

```
Open       0.310933
High       0.322564
Low        0.267250
Close      0.291683
Adj Close  0.276329
Volume     1.000000
```

[24]:
```python
# correlation coeffient of teslt
t.corr()
```

[24]:
```
               Date      Open      High       Low     Close  Adj Close  \
Date       1.000000 -0.770786 -0.782781 -0.764027 -0.774274  -0.774274
Open      -0.770786  1.000000  0.996463  0.995588  0.989473   0.989473
High      -0.782781  0.996463  1.000000  0.996513  0.995556   0.995556
Low       -0.764027  0.995588  0.996513  1.000000  0.996384   0.996384
Close     -0.774274  0.989473  0.995556  0.996384  1.000000   1.000000
Adj Close -0.774274  0.989473  0.995556  0.996384  1.000000   1.000000
Volume     0.516943 -0.696407 -0.683205 -0.708456 -0.693845  -0.693845

             Volume
Date       0.516943
Open      -0.696407
High      -0.683205
Low       -0.708456
Close     -0.693845
Adj Close -0.693845
Volume     1.000000
```

0.2.1 A2

[25]:
```python
import matplotlib.pyplot as plt
```

[26]:
```python
# Discard unneeded data

f_close = pd.DataFrame(f.Close)
```

[27]:
```python
 # Discard unneeded data

t_close = pd.DataFrame(t.Close)
```

[28]:
```python
# use rolling method to calculate and plot MOVING AVERAGE



f_close['MA_9'] = f_close.Close.rolling(9).mean().shift()
f_close['MA_20'] = f_close.Close.rolling(20).mean()
```

```
t_close['MA_9'] = t_close.Close.rolling(9).mean().shift()
t_close['MA_20'] = t_close.Close.rolling(20).mean()
```

[29]:
```
f_close['MA_20'].head(20)
```

[29]:
```
0        NaN
1        NaN
2        NaN
3        NaN
4        NaN
5        NaN
6        NaN
7        NaN
8        NaN
9        NaN
10       NaN
11       NaN
12       NaN
13       NaN
14       NaN
15       NaN
16       NaN
17       NaN
18       NaN
19    18.247
Name: MA_20, dtype: float64
```

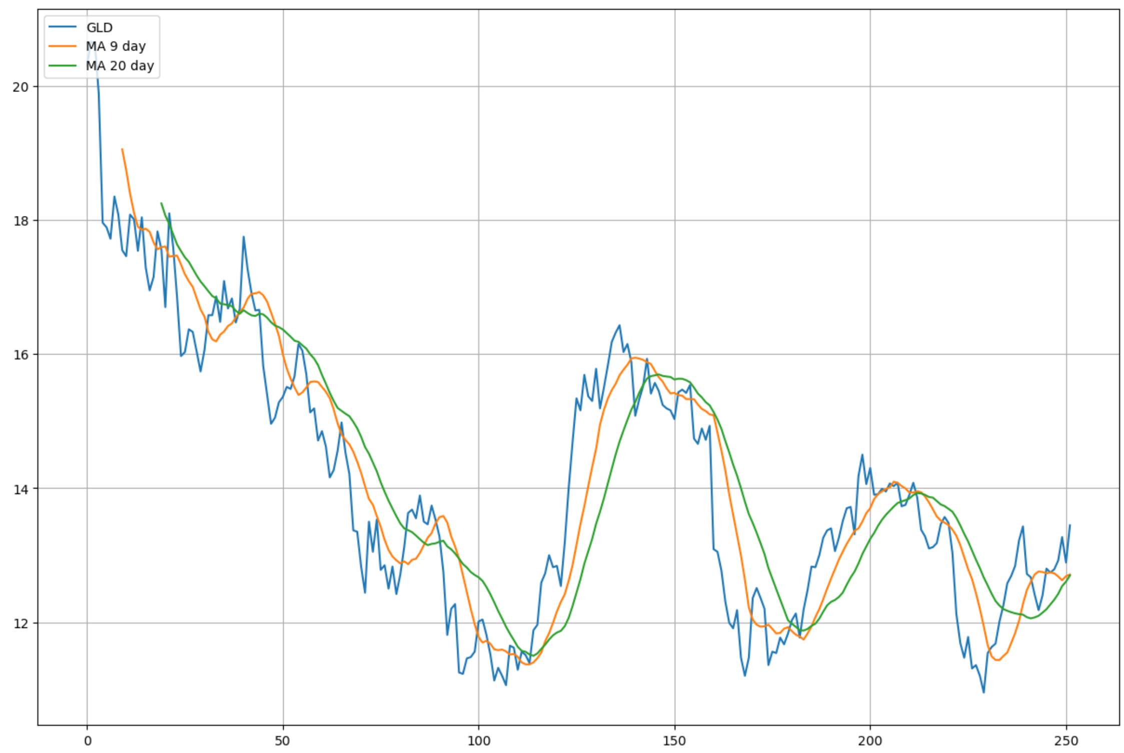
[30]:
```
t_close['MA_20'].head(20)
```

[30]:
```
0            NaN
1            NaN
2            NaN
3            NaN
4            NaN
5            NaN
6            NaN
7            NaN
8            NaN
9            NaN
10           NaN
11           NaN
12           NaN
13           NaN
14           NaN
15           NaN
16           NaN
17           NaN
```

```
18           NaN
19    293.925497
Name: MA_20, dtype: float64
```

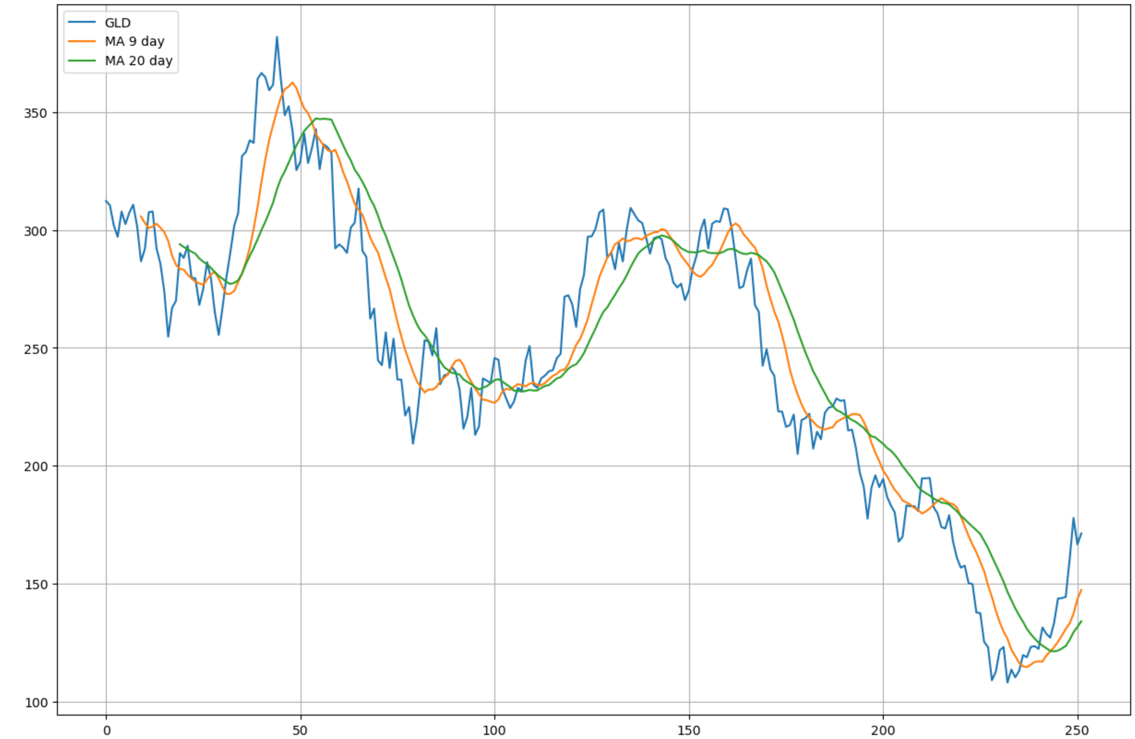
[31]:
```python
plt.figure(figsize=(15,10))
plt.grid(True)
plt.plot(f_close['Close'],label='GLD')
plt.plot(f_close['MA_9'], label='MA 9 day')
plt.plot(f_close['MA_20'], label='MA 20 day')
plt.legend(loc=2)
```

[31]: <matplotlib.legend.Legend at 0x7fba8ca56d40>



[32]:
```python
plt.figure(figsize=(15,10))
plt.grid(True)
plt.plot(t_close['Close'],label='GLD')
plt.plot(t_close['MA_9'], label='MA 9 day')
plt.plot(t_close['MA_20'], label='MA 20 day')
plt.legend(loc=2)
```

[32]: <matplotlib.legend.Legend at 0x7fba83bf4c70>



0.2.2 A3

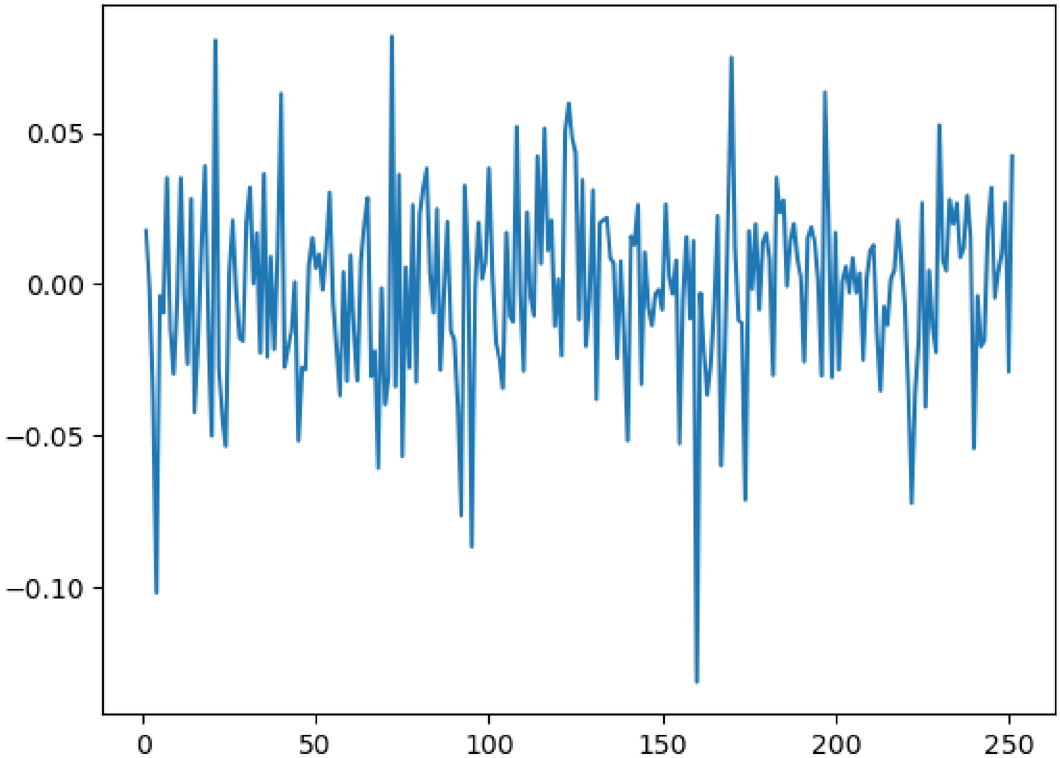
[33]:
```
# Compute log change (instantaneous rate of return)
f_close["change"] = np.log(f_close["Close"] / f_close["Close"].shift())
```

[34]:
```
# Compute log change (instantaneous rate of return)
t_close["change"] = np.log(t_close["Close"] / t_close["Close"].shift())
```

[35]:
```
# Plot reveals noisy data centered around 0

plt.plot(f_close.change)
```

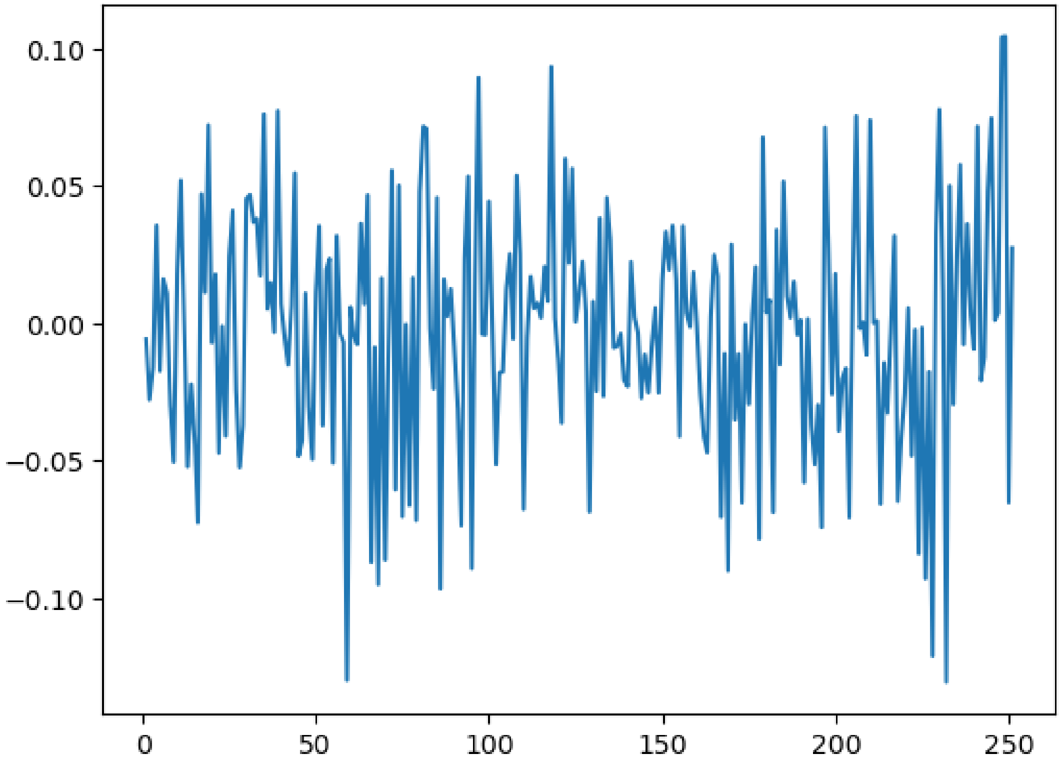
[35]:
```
[<matplotlib.lines.Line2D at 0x7fba83c832e0>]
```



[36]:
```
# Plot reveals noisy data centered around 0

plt.plot(t_close.change)
```

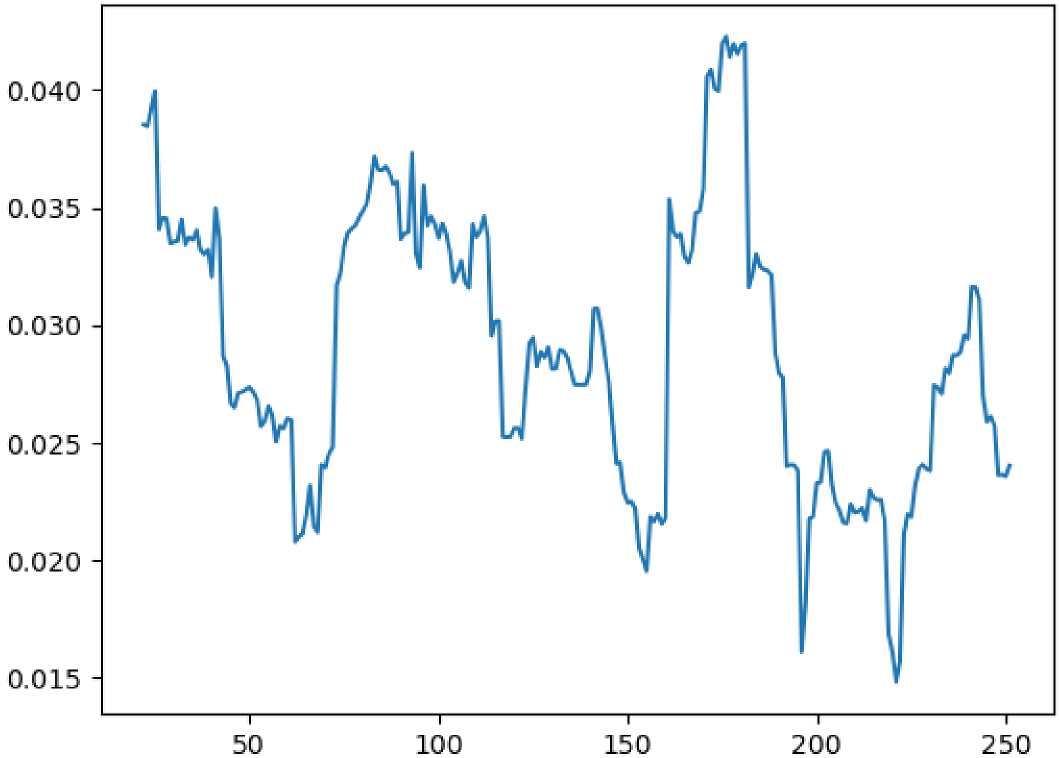
[36]:
```
[<matplotlib.lines.Line2D at 0x7fba8492d870>]
```



[37]:
```
# Compute rolling historical volatility, offset using .shift() method

f_close['Volatility'] = f_close.change.rolling(21).std().shift()
f_close['Volatility'].plot()
```

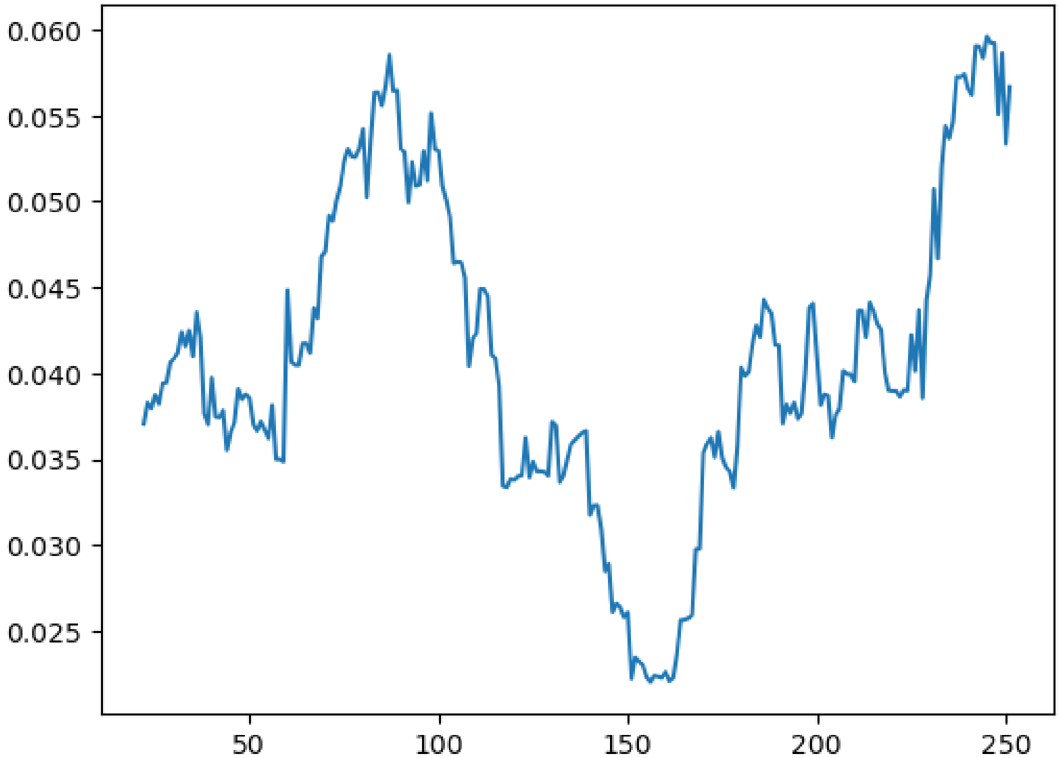
[37]: <AxesSubplot:>



[38]:
```
# Compute rolling historical volatility, offset using .shift() method

t_close['Volatility'] = t_close.change.rolling(21).std().shift()
t_close['Volatility'].plot()
```

[38]: <AxesSubplot:>


0.060
0.055
0.050
0.045
0.040
0.035
0.030
0.025
50
100
150
200
250