# Nicholas Colonna

## Midterm Exam

### "I pledge my honor that I have abided by the Stevens Honor System." - ncolonna

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
nflx = pd.read_csv('NFLX-2013_2018.csv')
nflx.head()
```

Out[2]:

| | Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|---|
| **0** | 2013-10-22 | 55.405716 | 55.594284 | 45.928570 | 46.074287 | 46.074287 | 181099800 |
| **1** | 2013-10-23 | 45.331429 | 47.884285 | 45.285713 | 47.177143 | 47.177143 | 58376500 |
| **2** | 2013-10-24 | 47.348572 | 48.121429 | 46.237144 | 47.317142 | 47.317142 | 33559400 |
| **3** | 2013-10-25 | 47.285713 | 48.171429 | 46.558571 | 46.861427 | 46.861427 | 24062500 |
| **4** | 2013-10-28 | 46.430000 | 47.279999 | 44.544285 | 44.857143 | 44.857143 | 34260800 |

In this section, I calculate the log returns for NFLX as well as beta.

In [10]:
```
n=22
nflx['LogRet'] = np.log(nflx['Adj Close']) - np.log(nflx['Adj Close'].shift(1
))


beta = 2 / (n + 1)

print('Beta =', beta)
print('\nLog Returns:')
nflx['LogRet']
```

Beta = 0.08695652173913043

Log Returns:

```
Out[10]: 0             NaN
         1        0.023654
         2        0.002963
         3       -0.009678
         4       -0.043712
         5        0.041484
         6       -0.028386
         7        0.013550
         8        0.020837
         9        0.024984
         10       0.011486
         11      -0.017338
         12      -0.026477
         13       0.024300
         14       0.008948
         15      -0.012447
         16       0.004634
         17       0.021510
         18       0.020771
         19      -0.023109
         20      -0.013195
         21       0.006590
         22       0.026105
         23      -0.001867
         24       0.006847
         25       0.014062
         26       0.020316
         27       0.009090
         28      -0.005153
         29      -0.002697
                    ...
         1229    -0.000775
         1230     0.021354
         1231     0.038634
         1232    -0.004877
         1233    -0.009799
         1234    -0.039758
         1235     0.048199
         1236    -0.001879
         1237    -0.004370
         1238    -0.011479
         1239     0.023044
         1240    -0.000487
         1241     0.022615
         1242     0.007461
         1243    -0.017435
         1244     0.019324
         1245    -0.011311
         1246    -0.000239
         1247    -0.036186
         1248    -0.034409
         1249    -0.006424
         1250     0.018757
         1251    -0.087556
         1252    -0.014807
         1253     0.055898
         1254    -0.019118
```

```
1255    0.039061
1256    0.051481
1257   -0.050586
1258   -0.041338
Name: LogRet, Length: 1259, dtype: float64
```

(a) In this section, I implement the EWMA forcasting method to estimate the expected volatility.

In [26]:
```
beta_array = np.array([(2/(n+1))**i for i in range(1, n)])
sigma_array = []
for i in range(0, len(nflx['LogRet'][0:-22])):
    sigma_array.append(np.array([(((nflx['LogRet'][-(n+i):-i] - nflx['LogRet']
[-(n+i):-i].mean()) ** 2).mean()) ** (1/2) for i in range(1,n)]))

EWMA_series = []
for i in range(0, len(sigma_array)):
    EWMA_series.append(((beta_array * sigma_array[i]).sum()) / beta_array.sum
())

EWMA = pd.DataFrame({'vol':EWMA_series})
EWMA['vol'].describe()
```

Out[26]:
```
count    1.237000e+03
mean     3.224598e-02
std      7.288785e-16
min      3.224598e-02
25%      3.224598e-02
50%      3.224598e-02
75%      3.224598e-02
max      3.224598e-02
Name: vol, dtype: float64
```

(b) Using Roll's model, I calculate gamma0 and gamma1 in order to calculate the volatility.

In [27]:
```
covar = np.cov((nflx['Adj Close'].shift(1) - nflx['Adj Close'].shift(2))[2:],
(nflx['Adj Close'] - nflx['Adj Close'].shift(1))[2:])
print(covar, '\n')
gamma0 = covar[1,1]
gamma1 = covar[0,1]

print('Gamma 0:', gamma0)
print('Gamma 1:', gamma1)
```

```
[[17.79506101  0.3388176 ]
 [ 0.3388176  17.95665788]]

Gamma 0: 17.956657877162414
Gamma 1: 0.3388176029651567
```

In this step, I calculate the bid-ask spread and the fundamental volatility of NFLX.

In [30]:

```
c = np.sqrt(np.abs(gamma1))

bid_ask = 2 * c
fund_vol = gamma0 + 2*gamma1

print('Bid-Ask Spread:', bid_ask)
print('Fundamental Volatility:', fund_vol)
```

```
Bid-Ask Spread: 1.1641608187276475
Fundamental Volatility: 18.63429308309273
```

(c) The mean EWMA expected volatility was 0.0322459, while the Rolls model got a value for gamma0 of 0.338818. As you can see, the EWMA model gave a slightly lower value than that of the Roll Model. Gamma0 in the Roll model is the variance, while the EWMA expected volatility was a standard deviation calculation.

Sigma^2 is the fundamental volatility due to Roll's model, which is different than gamma0, which is the lag 0 autocovariance of the price changes, also known as variance. Gamma0 is a known value that can be calculated from the data, while Sigma^2 is a parameter to the Roll model that is estimated using a combination of Gamma0 and Gamma1, the first order autocovariance of the price changes. As you can see from the results, the fundamental volatility is close to the value of Gamma0.