Fundamental Equity Risk Assessment

(MTH 800 Assignment 1)

Matthew Mercuri

The author is solely responsible for the content enclosed.

Contents

Introduction

This assignment demonstrates the most important that are to be applied in risk management. Various techniques to assess risk are shown, including, value-at-risk (VaR) and expected shortfall (ES). Legitimate data of stock price history for specific tickers was used in all computations. The objective of the assignment was to gain a significant understanding of different ways to assess risk. Throughout the report, notice is called to the benefits and disadvantages to the various models. Occasionally, commentary on why complications come about are provided. The project forces competently working with well established models that are in practice at the biggest financial institutions. Coding the models is good practice to develop intuition in their application. Care is also taken in constructing the models with different confidence levels. Thus, we discuss why metrics like VaR grow larger as alpha approaches one. To make calculations simple, the risk factors that are used are solely log-returns of various assets.

The project is segmented in 3 sections that have individual objectives. The first section shows how one calculates VaR and ES for a given ticker (AMD in this report). Interestingly though, it is done with 3 different methods: historical, parametric, and Monte Carlo. For the parametric and Monte Carlo methods, the risk factors are assumed to be Gaussian and then after they are drawn from a t-student distribution as well. In section 2, an analysis is done using a three-asset portfolio. The assets that are owned in the portfolio are Berskhire Hathaway Class B (BRK.B), Netflix (NFLX), and the S&P 500 (SPY). Using the constructed portfolio with different weights, VaR is calculated. In the last section, attention is focused on BRK.B log returns. Probability moments are computed from the historical data. Then, a Gaussian kernel is used to compute an empirical PDF that we compare to a Gaussian PDF. The Kolmogorov-Smirnov test is also used to compare the data to normal and t-student distributions. Please note that the code that was used to compute and generate results is shown in the appendix.

Section 1

First, two visualizations are produced using the stock data of AMD:

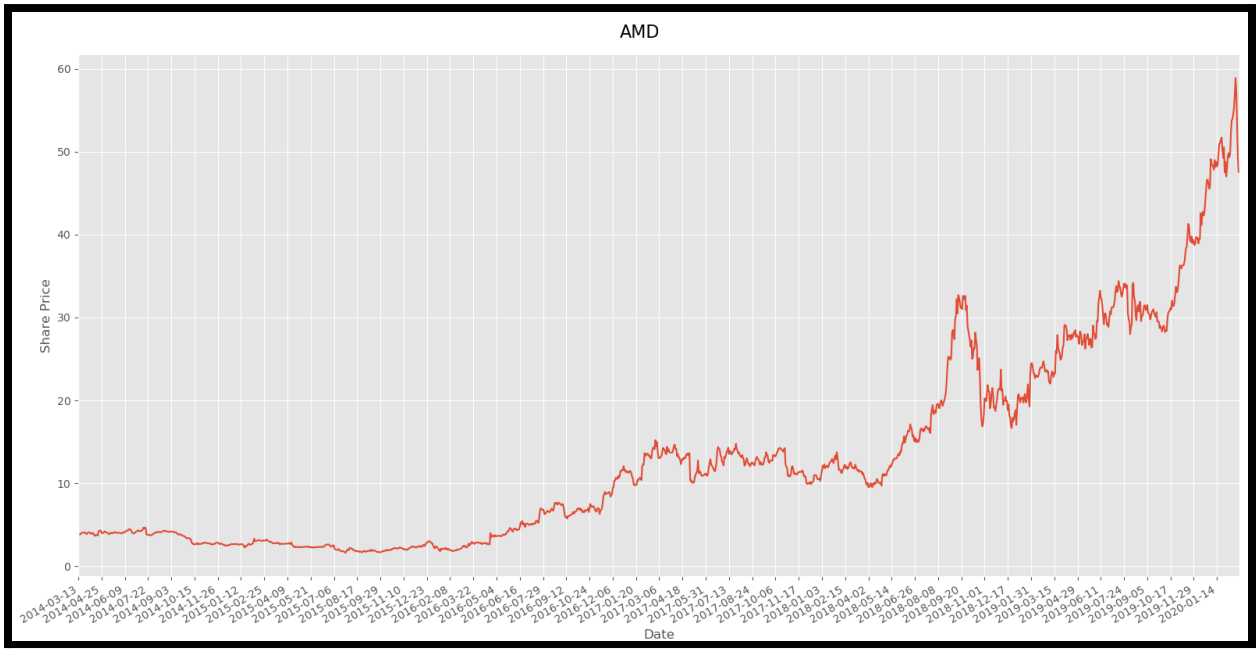

Figure 1 AMD stock price over time (using adjusted close prices)

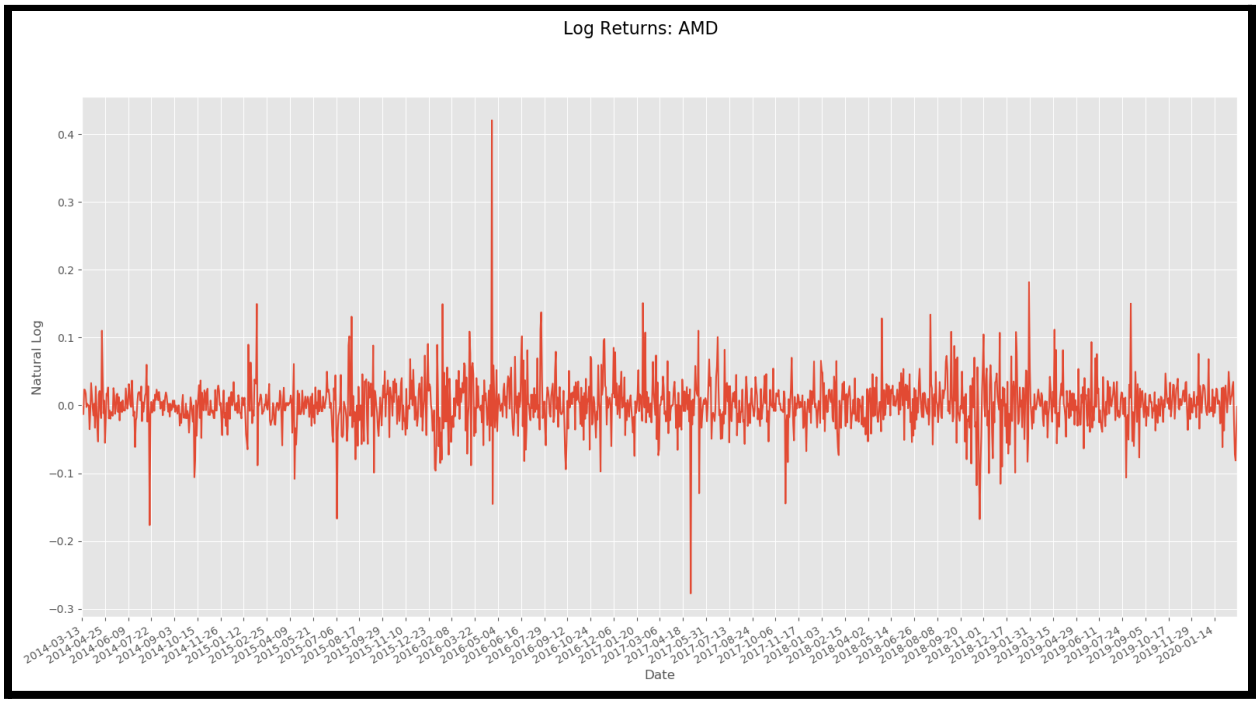

Figure 2 AMD log returns over time

The log returns (X_{t+1}) are calculated using a simple formula:

$$X_{t+1} = \ln ln\left(S_{t+1}\right) - \ln ln\left(S_t\right)$$

VaR and ES are calculated with their definitions, which are:

$$VaR_{\alpha} = F_L^{-1}(\alpha)$$

Where:

VaR_{α} is the maximum loss α% of the time

$F_L^{-1}(x)$ is the inverse cumulative distribution function

$$ES_{\alpha} = E\left[L|L > VaR_{\alpha}\right]$$

Where:

ES_{α} is the average loss when VaR_{α} is exceeded

L is the losses at time $t + 1$

Using the above definitions, it is possible to code the logic that is needed to compute VaR and ES. The initial portfolio value is $10,000,000. The chart below summarizes the results:

VaR (95% alpha)

Historical	Parametric (Gaussian)	Parametric (t-student)	Monte Carlo (Gaussian)	Monte Carlo (t-student)
$390,035.74	$463,694.89	$422,724.50	$462,982.36	$424,481.30

VaR (99% alpha)

Historical	Parametric (Gaussian)	Parametric (t-student)	Monte Carlo (Gaussian)	Monte Carlo (t-student)
$744,918.68	$666,889.86	$763,238.80	$666,177.22	$765,623.63

VaR is the maximum value that is lost alpha percent of the time. The first and arguably obvious observation is the fact that VaR at 99% alpha is greater than VaR 95%. This is consistent with the

loss distribution. The greater the value for alpha, the greater the loss. Simply put, when looking at the loss distribution PDF, the greater alpha is, the further it is to the right of the mean loss. Another important thing to notice is that the Monte Carlo (MC) simulations yield results like their respective distributions. This is simply the result of the law of large numbers. Essentially, so many simulations are performed that the distribution of losses approaches the distribution the random variables are drawn from. Last, is a focus on the differences between the VaRs given by the t-student and Gaussian distribution. Note that the degrees of freedom in this t-student distribution is 4. Notice that the 95% VaR is greater for the Gaussian than the t-student. Of course, the opposite is true for the 99% VaR. This is derived from the properties of the t-student distribution. The t-student has much "heavier tails" than the Gaussian distribution. This explains why alpha of 99% in the t-student distribution is further from the mean than it is in the Gaussian distribution. Also, a t-student distribution that approaches infinite degrees of freedom approximates a Gaussian distribution. Given this fact, it makes sense that for smaller values of alpha, smaller loss values are obtained than is the case of the Gaussian distribution. Perhaps the most important result, is the similarity between the t-student VaR and historical VaR. Thus, one can tell empirically that it better approximates the loss distribution.

ES shows similar observations:

ES (95% alpha)

Historical	Parametric (Gaussian)	Parametric (t-student)	Monte Carlo (Gaussian)	Monte Carlo (t-student)
$626,272.66	$588,284.18	$728,477.21	$587,476.59	$649,131.89

ES (99% alpha)

Historical	Parametric (Gaussian)	Parametric (t-student)	Monte Carlo (Gaussian)	Monte Carlo (t-student)
$881,997.16	$767,926.58	$2,038,376.07	$766,496.86	$1,069,740.19

ES is explained as the average loss when VaR at alpha is exceeded. Looking at the above results, it is easy to see that all the numbers exceed their respective values of VaR. This is consistent

with the very definition of ES. The difference between VaRs is greater in the t-student distributions than it is in the Gaussian distribution. This is again attributed to the fact that the t-student has heavier tails. The same observations regarding the relative size amongst the distributions is to be made from VaR (same relationships).

The next goal for this section is to back-test the results of our estimations for VaR. The test statistics are as follows:

$$H_0: P\left(L_{t+1} > \hat{VaR}_\alpha\right) = 1 - \alpha$$

$$H_a: P\left(L_{t+1} > \hat{VaR}_\alpha\right) \neq 1 - \alpha$$

Our back-test statistic is found by:

$$BT = \sum_{j=1}^{n} I_j \text{ where } I_j = 1_{\left[L_{t+1} > \hat{VaR}_\alpha\right]}$$

$$BTS = \frac{BT - n\alpha(1-\alpha)}{\sqrt{n\alpha(1-\alpha)}}$$

Reject the VaR if:

$$|BTS| \geq Z_{\frac{\alpha}{2}}$$

$Z_{\frac{\alpha}{2}}$ is equal to 1.96 for our 95% VaR and 2.576 for 99% VaR

The result for the test statistic given calculated estimations of VaR are summarized below:

VaR (95% alpha)

Historical	Parametric (Gaussian)	Parametric (t-student)	Monte Carlo (Gaussian)	Monte Carlo (t-student)
0.1156	-0.4625	-0.1734	-0.4625	-0.1734

VaR (99% alpha)

Historical	Parametric (Gaussian)	Parametric (t-student)	Monte Carlo (Gaussian)	Monte Carlo (t-student)
0.3039	0.937	0.3039	0.937	0.3039

As all the absolute values of the test statistics are below the critical value of $Z_{\frac{\alpha}{2}}$, one cannot reject the null hypothesis.

Section 2

A portfolio is constructed of three assets, with the goal of being able to calculate VaR given defined portfolio weights. Estimating the mean and covariance matrix can be achieved using code that abides by the following two formulas for mean and covariance matrix elements:

$$\hat{\mu}_k = \frac{1}{n}\sum_{j=1}^{n} X_j^{(k)} where\ k = 1,\ 2,\ \cdots,\ d$$

$$\hat{\sigma}_{ij} = \frac{1}{n}\sum_{j=1}^{n}\left(X_j^{(i)} - \hat{\mu}_i\right)\left(X_j^{(k)} - \hat{\mu}_k\right) where\ k = 1,\ 2,\ \cdots,\ d$$

Where:

$\hat{\mu}_k$ is the mean log return of asset *k*

$X_j^{(k)}$ is the log-return of asset *k* at *j*

$\hat{\sigma}_{ij}$ is the covariance between asset *i* and *j*

The Var is then defined by the matrix formula:

$$VaR_\alpha = -V_t w\mu^T + V_t^2 (w\sum_x w^t)^{\frac{1}{2}} Z_\alpha$$

Where:

V_t is the portfolio value at time *t*

w and w^t is the weights vector and transposed weights vector

μ^T is the transposed vector of average log returns

Σ is the covariance matrix

Z_α is the z-value (distance from 0 in standard normal distribution)

Portfolio one is given weights of $w = \left[w_1 = \frac{1}{3},\ w_2 = \frac{1}{3},\ w_3 = \frac{1}{3}\right]$ to SPY, BRK.B, and NFLX, respectively. Portfolio two is given weights $w = \left[w_1 = \frac{1}{3},\ w_2 = \frac{1}{3},\ w_3 = \frac{1}{3}\right]$ amongst the assets. The VaR yielded by the construction of the two portfolios is (assuming multivariate normal distribution):

Portfolio One

VaR (95% alpha)	VaR (99% alpha)
$195,639.64	$279,377.79

Portfolio Two

VaR (95% alpha)	VaR (99% alpha)
$196,269.18	$280,007.33

Looking at the values for VaR in the two portfolios, a rational investor would choose the weighting in portfolio one. Though, there is not much of a difference given the fact that the weights are only slightly different. One can not surprisingly notice that the VaR of a portfolio with multiple assets (and various covariances), is quantifiably less risky.

Section 3

In this section, using BRK.B (from the previous section) to compute the attributes of a distribution and then later the empirical pdf. One can compute the “moments” using an array of its log-returns as inputs to statistical definitions that can be programed. Essentially, use:

$$E\left[X^n\right] = \int_{-\infty}^{\infty} x^n dF(x)$$

$E\left[X^1\right]$ is the distribution's mean

$E\left[X^n\right]$ is the distribution's variance

$E\left[X^n\right]$ is the distribution's skewness

$E\left[X^n\right]$ is the distribution's kurtosis

The moments generated by the log returns of BRK.B are as follows:

Mean	0.000346
Variance	0.000104
Skewness	-0.324099
Kurtosis	4.305584

Looking at the computed values of the moments, there are some important observations to be made. One can clearly see that the mean return is close to zero like it is in the standard normal distribution. BRK.B is much less variant than a standard normal distribution is, though. Skewness is negative which means the left tail of the distribution contains more probability than the right. Kurtosis is quite a large value, which means the data is not symmetrical. This is in line with the understanding that returns are hardly normal. There is a lot of probability contained in the tails compared to standard normal.

Using a Gaussian kernel, one can derive an empirical probability distribution using the BRK.B data. This is useful in demonstrating just how non-normal returns can be. Below is a comparison between the empirical PDF obtained to a standard normal:

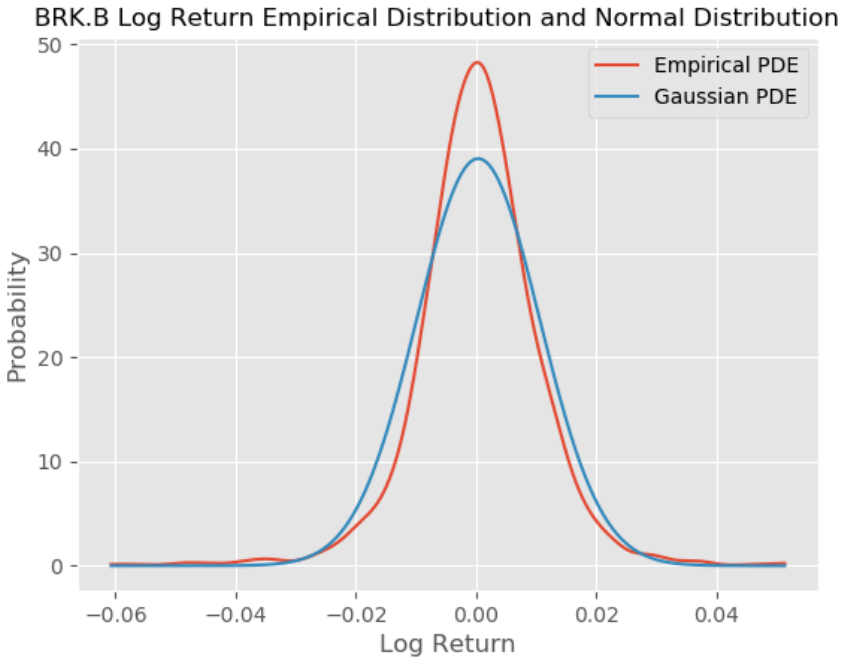

Figure 3 Compraing BRK.B empirical PDF to standard normal PDF

Figure 3 confirms the observations of the computed moments. It is evident by looking at the figure that the returns are in fact less variant than the normal distribution. The empirical PDF also confirms that there is much more area (probability) contained in the tail than in the normal PDF. This confirms observations made before that the t-student distribution provides a better estimate of returns.

The final objective of this section was to produce a quantifiable way to prove that the returns are highly non-normal. To achieve this, one can use the Kolmogorov-Smirnov test. There are many programming languages that have this test as a built-in tool. Performing the KS test on the BRK.B log returns yields a p-value that is far to small to accept that the returns can be accurately estimated from both the normal and t-student distribution

Conclusion

Overall, this project went a long way in building a sufficient understanding of important risk metrics. Learning to apply the definitions of VaR and ES to code will go a long way in becoming a

practitioner. Applying the concepts to both individual equities and portfolios is also useful in understanding how to account for risk and diversify unnecessary risk. This assignment was particularly proficient in systematically identifying exactly why a normal distribution is not a good fit for returns. Using different methods to calculate the risk metrics also solidifies understanding and helps identify short comings in distributions.

Appendix A – Code Console Output

Section 1

```
The historical one year 95% VaR for AMD is $390035.74
The historical one year 99% VaR for AMD is $744918.68
================================================================
The normal parametric one year 95% VaR for AMD is $463694.89
The normal parametric one year 99% VaR for AMD is $666889.86
================================================================
The t-student parametric one year 95% VaR for AMD is $422724.5
The t-student parametric one year 99% VaR for AMD is $763238.8
================================================================
The MC normal estimated one year 95% VaR for AMD is $462982.36
The MC normal estimated one year 99% VaR for AMD is $666177.22
================================================================
The MC t-student estimated one year 95% VaR for AMD is $424481.3
The MC t-student estimated one year 99% VaR for AMD is $765623.63
================================================================
The historical one year 95% ES for AMD is $626272.66
The historical one year 99% ES for AMD is $881997.16
================================================================
The normal parametric one year 95% ES for AMD is $588284.18
The normal parametric one year 99% ES for AMD is $767926.58
================================================================
The t-student parametric one year 95% ES for AMD is $728477.21
The t-student parametric one year 99% ES for AMD is $2038376.07
================================================================
The MC normal estimated one year 95% ES for AMD is $587476.59
The MC normal estimated one year 99% ES for AMD is $766496.86
================================================================
The MC t-student estimated one year 95% ES for AMD is $649131.89
The MC t-student estimated one year 99% ES for AMD is $1069740.19
================================================================
Our test statistic for historic 95% VaR is 0.1156, so we do not reject our null hypothesis.
Our test statistic for historic 99% VaR is 0.3039, so we do not reject our null hypothesis.
================================================================
Our test statistic for normal 95% VaR is -0.4625, so we do not reject our null hypothesis.
Our test statistic for normal 99% VaR is 0.937, so we do not reject our null hypothesis.
================================================================
Our test statistic for t-student 95% VaR is -0.1734, so we do not reject our null hypothesis.
Our test statistic for t-student 99% VaR is 0.3039, so we do not reject our null hypothesis.
================================================================
```

```
Our test statistic for MC (Gaussian) 95% VaR is -0.4625, so we do not reject our null hypothesis.
Our test statistic for MC (Gaussian) 99% VaR is 0.937, so we do not reject our null hypothesis.
================================================================
Our test statistic for MC (t-student) 95% VaR is -0.1734, so we do not reject our null hypothesis.
Our test statistic for MC (t-student) 99% VaR is 0.3039, so we do not reject our null hypothesis.
================================================================
```

Section 2

```
The multivariate normal 95% VaR for Port 1 is: $195639.64
The multivariate normal 99% VaR for Port 1 is: $279377.79
************************************************************
The multivariate normal 95% VaR for Port 2 is: $196269.18
The multivariate normal 99% VaR for Port 2 is: $280007.33
```

Section 3

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The mean of BRK.B log returns is: 0.000346
The variance of BRK.B log returns is: 0.000104
The skewness of BRK.B log returns is: -0.324099
The kurtosis of BRK.B log returns is: 4.305584
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
The p-value for the normal KS Test is 1e-322. Thus, reject null hypothesis.
The p-value for the t-student KS Test is 1e-322. Thus, reject null hypothesis.
```

Append B – Code

```
1.  import numpy as np
2.  import matplotlib.pyplot as plt
3.  import pandas as pd
4.  from scipy import stats
5.  from scipy.stats import norm, t
6.
7.  plt.style.use('ggplot')
8.  MC_SIMS = 1000000
9.  LAST_N_DAYS = 1500
10. INITIAL_PORT_VALUE = 10000000
11.
12. amd_df = pd.read_csv('Data/AMD.csv', index_col=0)  # reading in stock data
13. brk_df = pd.read_csv('Data/BRK.B.csv', index_col=0)
14. spy_df = pd.read_csv('Data/SPY.csv', index_col=0)
15. nflx_df = pd.read_csv('Data/NFLX.csv', index_col=0)
16.
17. # Calculating log returns
18. amd_df['Log Returns'] = np.log(amd_df['Adjusted Close']) - np.log(amd_df['Adjusted Clos
    e'].shift(1))
19. brk_df['Log Returns'] = np.log(brk_df['Adjusted Close']) - np.log(brk_df['Adjusted Clos
    e'].shift(1))
```

```
20. spy_df['Log Returns'] = np.log(spy_df['Adjusted Close']) - np.log(spy_df['Adjusted Clos
    e'].shift(1))
21. nflx_df['Log Returns'] = np.log(nflx_df['Adjusted Close']) - np.log(nflx_df['Adjusted C
    lose'].shift(1))
22.
23. # Cleaning dataset
24. amd_df.drop(columns=['Open', 'High', 'Low', 'Close', 'Volume',
25.                      'Dividend Amount', 'Split Coefficient'], inplace=True)
26. amd_df.drop(amd_df.tail(1).index, inplace=True)
27. pd.to_datetime(amd_df.index)
28. amd_df = amd_df.tail(LAST_N_DAYS)
29.
30. brk_df.drop(columns=['Open', 'High', 'Low', 'Close', 'Volume',
31.                      'Dividend Amount', 'Split Coefficient'], inplace=True)
32. pd.to_datetime(brk_df.index)
33. brk_df = brk_df.tail(LAST_N_DAYS)
34.
35. spy_df.drop(columns=['Open', 'High', 'Low', 'Close', 'Volume',
36.                      'Dividend Amount', 'Split Coefficient'], inplace=True)
37. pd.to_datetime(spy_df.index)
38. spy_df = spy_df.tail(LAST_N_DAYS)
39.
40. nflx_df.drop(columns=['Open', 'High', 'Low', 'Close', 'Volume',
41.                       'Dividend Amount', 'Split Coefficient'], inplace=True)
42. pd.to_datetime(nflx_df.index)
43. nflx_df = nflx_df.tail(LAST_N_DAYS)
44.
45.
46. def plot_amd_returns():
47.     # To Do: graph prices
48.     # Plotting log returns over time
49.
50.     fig1, ax1 = plt.subplots()
51.     # ax1.plot(amd_df.index, amd_df['Log Returns'])
52.     ax1.plot(amd_df.index, amd_df['Adjusted Close'])
53.     ax1.xaxis.set_major_locator(plt.MaxNLocator(50))
54.     ax1.set_xlim(amd_df.index.min(), amd_df.index.max())
55.     fig1.autofmt_xdate()
56.
57.     fig1.suptitle('AMD', fontsize=16, y=0.92)
58.     ax1.set_xlabel('Date')
```

```
59.     ax1.set_ylabel('Share Price')
60.
61.     plt.show()
62.
63.
64. def calculate_loss_stats_amd():
65.     losses_array = np.array(amd_df['Log Returns']*-INITIAL_PORT_VALUE)
66.     losses_array = losses_array[len(losses_array)-252:]  # using last years worth of da
    ta
67.
68.     # historical approach
69.     historical_var = np.quantile(losses_array, [0.95, 0.99])
70.     print(f'The historical one year 95% VaR for AMD is ${round(historical_var[0], 2)}')
71.     print(f'The historical one year 99% VaR for AMD is ${round(historical_var[1], 2)}')
72.     print("=====================================================================")
73.
74.     # parametric approach (Gaussian)
75.     mu = amd_df['Log Returns'].tail(252).mean()
76.     sigma = amd_df['Log Returns'].tail(252).std()
77.     varnorm95 = (-INITIAL_PORT_VALUE*mu) + (INITIAL_PORT_VALUE*sigma) * norm.ppf(0.95)
78.     varnorm99 = (-INITIAL_PORT_VALUE*mu) + (INITIAL_PORT_VALUE*sigma) * norm.ppf(0.99)
79.     print(f'The normal parametric one year 95% VaR for AMD is ${round(varnorm95, 2)}')
80.     print(f'The normal parametric one year 99% VaR for AMD is ${round(varnorm99, 2)}')
81.     print("=====================================================================")
82.
83.     # parametric approach (t-student)
84.     nu = 4
85.     vart95 = (-INITIAL_PORT_VALUE*mu) + (INITIAL_PORT_VALUE*sigma) * np.sqrt((nu-2) / n
    u) * t.ppf(0.95, nu)
86.     vart99 = (-INITIAL_PORT_VALUE*mu) + (INITIAL_PORT_VALUE*sigma) * np.sqrt((nu-2) / n
    u) * t.ppf(0.99, nu)
87.     print(f'The t-student parametric one year 95% VaR for AMD is ${round(vart95, 2)}')
88.     print(f'The t-student parametric one year 99% VaR for AMD is ${round(vart99, 2)}')
```

```
    print("=====================================================================")

    # monte-carlo method (Gaussian)
    normal_gen_losses_array = np.random.normal(-INITIAL_PORT_VALUE*mu, INITIAL_PORT_VALUE*sigma, MC_SIMS)
    var_norm_mc = np.quantile(normal_gen_losses_array, [0.95, 0.99])
    print(f'The MC normal estimated one year 95% VaR for AMD is ${round(var_norm_mc[0], 2)}')
    print(f'The MC normal estimated one year 99% VaR for AMD is ${round(var_norm_mc[1], 2)}')
    print("=====================================================================")

    # monte-carlo method (t-student)
    t_gen_losses_array = (-INITIAL_PORT_VALUE*mu) + (INITIAL_PORT_VALUE*sigma) * np.sqrt((nu-2)/nu) * np.random.standard_t(nu, size=MC_SIMS)
        var_t_mc = np.quantile(t_gen_losses_array, [0.95, 0.99])
        print(f'The MC t-student estimated one year 95% VaR for AMD is ${round(var_t_mc[0], 2)}')
        print(f'The MC t-student estimated one year 99% VaR for AMD is ${round(var_t_mc[1], 2)}')
        print("=====================================================================")

        # ES using historical method
        es95_array = losses_array > historical_var[0]
        es99_array = losses_array > historical_var[1]
        es95_hist = sum(losses_array * es95_array) / sum(es95_array)
        es99_hist = sum(losses_array * es99_array) / sum(es99_array)
        print(f'The historical one year 95% ES for AMD is ${round(es95_hist, 2)}')
        print(f'The historical one year 99% ES for AMD is ${round(es99_hist, 2)}')
        print("=====================================================================")

        # ES using parametric approach (Gaussian)
        es95_normal = (-INITIAL_PORT_VALUE*mu) + ((INITIAL_PORT_VALUE*sigma*norm.pdf(norm.ppf(0.95))) / 0.05)
        es99_normal = (-INITIAL_PORT_VALUE*mu) + ((INITIAL_PORT_VALUE*sigma*norm.pdf(norm.ppf(0.99))) / 0.01)
        print(f'The normal parametric one year 95% ES for AMD is ${round(es95_normal, 2)}')
        print(f'The normal parametric one year 99% ES for AMD is ${round(es99_normal, 2)}')
        print("=====================================================================")
```

```
120.
121.        # ES using parametric approach (t-student)
122.        es95_t = (-INITIAL_PORT_VALUE*mu) + (INITIAL_PORT_VALUE*sigma*(nu+(t.ppf(0.95,
    nu))**2)) / (((nu-1) * t.pdf(t.ppf(0.95, nu), nu)) / 0.05)
123.        es99_t = (-INITIAL_PORT_VALUE*mu) + (INITIAL_PORT_VALUE*sigma*(nu+(t.ppf(0.99,
    nu))**2)) / (((nu-1) * t.pdf(t.ppf(0.99, nu), nu)) / 0.01)
124.        print(f'The t-student parametric one year 95% ES for AMD is ${round(es95_t, 2)}
    ')
125.        print(f'The t-student parametric one year 99% ES for AMD is ${round(es99_t, 2)}
    ')
126.        print("=================================================================")
127.
128.        # ES monte-carlo method (Gaussian)
129.        excess_var_95_losses = normal_gen_losses_array > var_norm_mc[0]
130.        es95_mc_norm = sum(normal_gen_losses_array * excess_var_95_losses) / sum(excess
    _var_95_losses)
131.        excess_var_99_losses = normal_gen_losses_array > var_norm_mc[1]
132.        es99_mc_norm = sum(normal_gen_losses_array * excess_var_99_losses) / sum(excess
    _var_99_losses)
133.        print(f'The MC normal estimated one year 95% ES for AMD is ${round(es95_mc_norm
    , 2)}')
134.        print(f'The MC normal estimated one year 99% ES for AMD is ${round(es99_mc_norm
    , 2)}')
135.        print("=================================================================")
136.
137.        # ES monte-carlo method (Gaussian)
138.        excess_var_95_losses_t = t_gen_losses_array > var_t_mc[0]
139.        es95_mc_t = sum(t_gen_losses_array * excess_var_95_losses_t) / sum(excess_var_9
    5_losses_t)
140.        excess_var_99_losses_t = t_gen_losses_array > var_t_mc[1]
141.        es99_mc_t = sum(t_gen_losses_array * excess_var_99_losses_t) / sum(excess_var_9
    9_losses_t)
142.        print(f'The MC t-student estimated one year 95% ES for AMD is ${round(es95_mc_t
    , 2)}')
143.        print(f'The MC t-student estimated one year 99% ES for AMD is ${round(es99_mc_t
    , 2)}')
144.        print("=================================================================")
145.
146.        # ==================== Backtesting VaR ====================
147.        # Historical VaR
148.        exceeded_95_hist_array = losses_array > historical_var[0]
```

```
149.        exceeded_95_hist = sum(exceeded_95_hist_array)
150.        exceeded_95_hist_s = (exceeded_95_hist - (len(losses_array)*0.05)) / np.sqrt(le
    n(losses_array)*0.95*0.05)
151.        print(f'Our test statistic for historic 95% VaR is {round(exceeded_95_hist_s, 4
    )}, so we do not reject our null hypothesis.')
152.        exceeded_99_hist_array = losses_array > historical_var[1]
153.        exceeded_99_hist = sum(exceeded_99_hist_array)
154.        exceeded_99_hist_s = (exceeded_99_hist - (len(losses_array)*0.01)) / np.sqrt(le
    n(losses_array)*0.99*0.01)
155.        print(f'Our test statistic for historic 99% VaR is {round(exceeded_99_hist_s, 4
    )}, so we do not reject our null hypothesis.')
156.        print("=================================================================")
157.
158.        # Parametric (Gaussian)
159.        exceeded_95_normal_array = losses_array > varnorm95
160.        exceeded_95_norm = sum(exceeded_95_normal_array)
161.        exceeded_95_norm_s = (exceeded_95_norm - (len(losses_array)*0.05)) / np.sqrt(le
    n(losses_array)*0.95*0.05)
162.        print(f'Our test statistic for normal 95% VaR is {round(exceeded_95_norm_s, 4)}
    , so we do not reject our null hypothesis.')
163.        exceeded_99_normal_array = losses_array > varnorm99
164.        exceeded_99_norm = sum(exceeded_99_normal_array)
165.        exceeded_99_norm_s = (exceeded_99_norm - (len(losses_array)*0.01)) / np.sqrt(le
    n(losses_array)*0.99*0.01)
166.        print(f'Our test statistic for normal 99% VaR is {round(exceeded_99_norm_s, 4)}
    , so we do not reject our null hypothesis.')
167.        print("=================================================================")
168.
169.        # Parametric t-student
170.        exceeded_95_t_array = losses_array > vart95
171.        exceeded_95_t = sum(exceeded_95_t_array)
172.        exceeded_95_t_s = (exceeded_95_t - (len(losses_array)*0.05)) / np.sqrt(len(loss
    es_array)*0.95*0.05)
173.        print(f'Our test statistic for t-student 95% VaR is {round(exceeded_95_t_s, 4)}
    , so we do not reject our null hypothesis.')
174.        exceeded_99_t_array = losses_array > vart99
175.        exceeded_99_t = sum(exceeded_99_t_array)
176.        exceeded_99_t_s = (exceeded_99_t - (len(losses_array)*0.01)) / np.sqrt(len(loss
    es_array)*0.99*0.01)
177.        print(f'Our test statistic for t-student 99% VaR is {round(exceeded_99_t_s, 4)}
    , so we do not reject our null hypothesis.')
```

```
178.        print("=====================================================================")
179.
180.        # MC Gaussian
181.        exceeded_95_mcg_array = losses_array > var_norm_mc[0]
182.        exceeded_95_mcg = sum(exceeded_95_mcg_array)
183.        exceeded_95_mcg_s = (exceeded_95_mcg - (len(losses_array)*0.05)) / np.sqrt(len(
    losses_array)*0.95*0.05)
184.        print(f'Our test statistic for MC (Gaussian) 95% VaR is {round(exceeded_95_mcg_
    s, 4)}, so we do not reject our null hypothesis.')
185.        exceeded_99_mcg_array = losses_array > var_norm_mc[1]
186.        exceeded_99_mcg = sum(exceeded_99_mcg_array)
187.        exceeded_99_mcg_s = (exceeded_99_mcg - (len(losses_array)*0.01)) / np.sqrt(len(
    losses_array)*0.99*0.01)
188.        print(f'Our test statistic for MC (Gaussian) 99% VaR is {round(exceeded_99_mcg_
    s, 4)}, so we do not reject our null hypothesis.')
189.        print("=====================================================================")
190.
191.        # MC t-student
192.        exceeded_95_mct_array = losses_array > var_t_mc[0]
193.        exceeded_95_mct = sum(exceeded_95_mct_array)
194.        exceeded_95_mct_s = (exceeded_95_mct - (len(losses_array)*0.05)) / np.sqrt(len(
    losses_array)*0.95*0.05)
195.        print(f'Our test statistic for MC (t-student) 95% VaR is {round(exceeded_95_mct
    _s, 4)}, so we do not reject our null hypothesis.')
196.        exceeded_99_mct_array = losses_array > var_t_mc[1]
197.        exceeded_99_mct = sum(exceeded_99_mct_array)
198.        exceeded_99_mct_s = (exceeded_99_mct - (len(losses_array)*0.01)) / np.sqrt(len(
    losses_array)*0.99*0.01)
199.        print(f'Our test statistic for MC (t-student) 99% VaR is {round(exceeded_99_mct
    _s, 4)}, so we do not reject our null hypothesis.')
200.        print("=====================================================================")
201.
202.
203.    def port_construction():
204.        ret_array = np.column_stack((spy_df['Log Returns'], brk_df['Log Returns'], nflx
    _df['Log Returns']))
205.        mean = np.mean(ret_array, axis=0)
206.        cov = np.cov(np.transpose(ret_array))
207.
208.        weights_1 = np.array([(1/3), (1/3), (1/3)])
209.        weights_2 = np.array([0.5, 0.25, 0.25])
```

```
210.
211.        mean_ret_port1 = np.dot(weights_1, np.transpose(mean))
212.        mean_ret_port2 = np.dot(weights_2, np.transpose(mean))
213.
214.        vol_port_1 = np.sqrt(np.dot(np.dot(weights_1, cov), np.transpose(weights_1)))
215.        vol_port_2 = np.sqrt(np.dot(np.dot(weights_2, cov), np.transpose(weights_2)))
216.
217.        norm_var95_port1 = np.dot(-INITIAL_PORT_VALUE*weights_1, mean)+(vol_port_1*INIT
    IAL_PORT_VALUE*norm.ppf(0.95))
218.        norm_var99_port1 = np.dot(-INITIAL_PORT_VALUE*weights_1, mean)+(vol_port_1*INIT
    IAL_PORT_VALUE*norm.ppf(0.99))
219.        print(f'The multivariate normal 95% VaR for Port 1 is: ${round(norm_var95_port1
    , 2)}')
220.        print(f'The multivariate normal 99% VaR for Port 1 is: ${round(norm_var99_port1
    , 2)}')
221.        print('*****************************************************************')
222.
223.        norm_var95_port2 = np.dot(-INITIAL_PORT_VALUE*weights_2, mean)+(vol_port_1*INIT
    IAL_PORT_VALUE*norm.ppf(0.95))
224.        norm_var99_port2 = np.dot(-INITIAL_PORT_VALUE*weights_2, mean)+(vol_port_1*INIT
    IAL_PORT_VALUE*norm.ppf(0.99))
225.        print(f'The multivariate normal 95% VaR for Port 2 is: ${round(norm_var95_port2
    , 2)}')
226.        print(f'The multivariate normal 99% VaR for Port 2 is: ${round(norm_var99_port2
    , 2)}')
227.
228.
229.    def brk_analysis():
230.
231.        # Finding momements of the log returns
232.        brk_mean = brk_df['Log Returns'].mean()
233.        brk_var = brk_df['Log Returns'].var()
234.        brk_skew = brk_df['Log Returns'].skew()
235.        brk_kur = brk_df['Log Returns'].kurtosis()
236.        print('+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++')
237.        print(f'The mean of BRK.B log returns is: {round(brk_mean, 6)}')
238.        print(f'The variance of BRK.B log returns is: {round(brk_var, 6)}')
239.        print(f'The skewness of BRK.B log returns is: {round(brk_skew, 6)}')
240.        print(f'The kurtosis of BRK.B log returns is: {round(brk_kur, 6)}')
241.        print('+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++')
242.
```

```
243.        # Gaussian kernel density function
244.        data = brk_df['Log Returns']
245.        kde = stats.gaussian_kde(data)
246.        x = np.linspace(data.min(), data.max(), 1000)
247.        p = kde(x)
248.
249.        fig2, ax2 = plt.subplots()
250.        ax2.plot(x, p)
251.        ax2.plot(x, stats.norm.pdf(x, brk_mean, np.sqrt(brk_var)))
252.        fig2.suptitle('BRK.B Log Return Empirical Distribution and Normal Distribution'
    , y=0.92)
253.        ax2.set_xlabel('Log Return')
254.        ax2.set_ylabel('Probability')
255.        ax2.legend(['Empirical PDE', 'Gaussian PDE'])
256.        plt.show()
257.
258.        # KS Tests
259.        normal_d, normal_p_value = stats.kstest(brk_df['Log Returns'], 'norm')
260.        print(f'The p-value for the normal KS Test is {normal_p_value}. Thus, reject nu
    ll hypothesis.')
261.        t_d, t_p_value = stats.kstest(brk_df['Log Returns'], 't', args=(4, brk_mean))
262.        print(f'The p-value for the t-student KS Test is {normal_p_value}. Thus, reject
     null hypothesis.')
263.
264.
265.    brk_analysis()
266.    port_construction()
267.    calculate_loss_stats_amd()
268.    plot_amd_returns()
```