2020

ANALYSIS OF VAR/ES ESTIMATION FOR PORTFOLIO IN DIFFERENT TIME PERIODS

Final Research Report

Group 7

Xinyi Liu

Junjie Liang

Zhijiao Chen

Fangzhou Xiang

1. Introduction

Risk measurement is the basis of risk management. Markowitz[1] was the first to propose a measure of financial risk. He suggested using variance as a measure of financial risk. Variance has been popular since the 1950s and is the cornerstone of modern investment theory. However, it is not a satisfactory measure of risk for two reasons: it penalizes profit and loss in the same way, and is not directly related to loss. In the 1990s, Morgan's risk managers proposed a new risk measurement method-Value at risk (VaR)[2].

The VaR approach is attractive because it is easy to understand (VaR is measured in monetary units) and it provides an estimate of the amount of capital that is needed to support a certain level of risk. Another advantage of this measure is the ability to incorporate the effects of portfolio diversification[3]. Currently, Value at Risk is the main measurement tool for market risk management of bank securities, insurance companies, and investment funds in countries around the world. Scholars at home and abroad have conducted extensive and in-depth research on VaR, providing a large number of theoretical foundations and operability for risk management of financial products.

In this project, we estimate Value-at-risk (VaR) and Expected Shortfall (ES) of a portfolio between normal period and coronavirus period that contains several assets which include US stock, US index, with the US stocks hedged with stock put option. The assets we used are Amazon stock price (AMZN), Boeing stock price (BA) and SP500. Our data starts at 1997-05-15, which is also the date when amazon's stock price starts. We compare the results in the portfolio with fully-hedged, partially hedged or without the hedging. This topic is inspired by the current issue of COVID-19 and by our discussions in class about options and partial risk hedges (Section 8.1 – 8.2).

Since coronavirus has just occurred, we set our normal time period as 2020-01-02 to 2020-02-21 and extreme time period as 2020-02-22 to 2020-4-13. We used a rolling window with a 50 days window size, and estimated the VaR and ES (p=1%) for no option hedged, single stock option hedged, double stock option hedged, and fully stocks option hedged in normal period and extreme period.

The rest of the report is structured as follows. In Section 2, we conduct a literature review on risk management using options and current VaR approaches. Section 3 introduces the methodology of our project. Section 4 includes an overview of the data and presents the results of the full data sample. Section 5 is the reflection of the entire Hedge process. Section 6 explains our final results. Finally, we made some conclusions and suggestion in Section 7.

1 Markowitz H. Portfolio selection[J]. The journal of finance, 1952, 7(1): 77-91
2 Morgan J P. Risk Metrics TM, Technical Document. Risk Metrics Group[J]. 1996.
3 RAAJI G D, Raunig B. A COMPARISON OF VALUE AT RISK APPROACHES AND THEIR IMPLICATIONS[J]. 1998.

2. Literature Review

This section introduces the literature related to risk management using options, and current VaR approaches.

2.1. Risk Management Using Options

Seeking the optimal hedging strategy has always been an interesting and important issue in the field of finance. For example, as we all know, under the assumption of market completeness, we can construct a perfect hedging strategy that perfectly replicates or pays out equity with a self-funded investment portfolio. However, when the market is incomplete, perfect hedging is usually impossible. Therefore, some people advocate a compromise strategy, the so-called partial hedging.

Partial hedging does not require full hedging of all future debt, but relaxes this strict condition by only partially hedging future debt. The cost of partial hedging (that is, hedging budget) is usually much smaller than the corresponding full hedging strategy, which also leads to profitable strategies. An example of a partial hedging strategy is the quantile hedging which is proposed by Follmer and Leukert (1999)[4]. The quantile hedging ensures that, for a given budget constraint, the probability of meeting the future obligation is optimally maximized.

2.2. Current VaR Approaches

It is also worth noting that various of VaR approaches have been proposed in the literature. In general, there are variance-covariance approach, historical simulations and Monte Carlo methods.

The variance-covariance method is usually estimated based on the daily historical time series of related risk factors, using an equal weight moving average (usually assumed that the mean is zero)[5]:

$$\sigma_t^2 = \frac{1}{M}\sum_{j=1}^{M} R_{t,j}^2$$

Another frequently used estimator is the exponentially weighted moving average (EWMA). Compared with the equal weight moving average, the exponentially weighted moving average, when calculating the conditional variance (covariance), weighs the current observations more than the past observations. Exponentially weighted moving average (EWMA) framework proposed by J.P Morgan's RiskMetrics assigns geometrically declining weights on past observations with the highest weight been attributed to the latest observation. Other methods in this direction are the famous ARCH and GARCH models,

[4] Föllmer H, Leukert P. Efficient hedging: cost versus shortfall risk[J]. Finance and Stochastics, 2000, 4(2): 117-146.
[5] Figlewski S. Forecasting volatility using historical data[J]. 1994.

which were proposed by Engle (1982) and Bollerslev (1986), respectively. So and Yu (2006) [6]used the GARCH model to predict the stock index of 12 countries and the VaR of 5 exchange rates. They confirmed that the distribution of stock index and exchange rate returns has a thick-tailed characteristic at a confidence level of 99%. The VaR estimated based on the GARCH model at% is very effective in risk management.

The second method is historical simulation. Compared with the previous method, this method does not make specific allocation assumptions for individual market risk factors (ie, returns), nor does it estimate variance or covariance. On the contrary, we only assume that the distribution of relevant market returns over the sample period is constant.

The third method is to convert the original data into result data that is normally distributed, and to take advantage of the convenience of the normal distribution. Xiaoyu Wang and Dejun Xie etc. proposed a Monte Carlo simulation-based approach for measuring Value-at-Risk of a portfolio consisting of options and bonds.[7] A prominent advantage of this method is that its implementation does not require prior knowledge of the joint distribution of risk factors or other statistical characteristics.

3. Methodology

3.1. Markowitz mean-variance portfolio theory and Sharpe ratio

To estimate the proportions of budget that will be allocated to each asset we invest, Markowitz's mean-variance view of portfolio is a good theory to measure the variance of stocks. To start with, Markowitz mean-variance portfolio theory is meant to assess a portfolio's expected return under a given volatility level. The expected return of our assets can be written as:

$$E(\mathrm{R}_p) = \sum_{1}^{i} w_i * E(R_i)$$

in which w_i and R_i refer to the percentage and annual(250-day) mean return of each asset. The volatility of our portfolio is measured by the yearly standard deviation:

$$\sigma_p = \sqrt{\sum_i \sum_j w_i w_j \sigma_i \sigma_j \rho_{ij}}$$

in which ρ_{ij} is the correlation coefficient between the return of asset i and j. For the convenience of

[6] So M K P, Philip L H. Empirical analysis of GARCH models in value at risk estimation[J]. Journal of International Financial Markets, Institutions and Money, 2006, 16(2): 180-197.
[7] Wang X, Xie D, Jiang J, et al. Value-at-Risk estimation with stochastic interest rate models for option-bond portfolios[J]. Finance Research Letters, 2017, 21: 10-20.

calculation in python, matrix form $\sigma_p = \sqrt{W^T * Covariance\ matrix * W}$ is introduced to align with the Numpy and Pandas data structure, where W^T is the transposed matrix of W and covariance matrix is: $\begin{bmatrix} \sigma_{11} & \cdots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{n1} & \cdots & \sigma_{nn} \end{bmatrix}$.

Sharpe ratio is an indicator to evaluate the risk premium in relative to the standard deviation of return, in other word it measures the volatility adjusted return of the portfolio. The formula can be expressed as:

$$\text{Sharpe ratio} = \frac{E(R_p) - Rf}{\sigma_p}$$

Maximized Sharpe ratio portfolio optimization will enable us to find out the investment mix that has the highest return while considering the risk as well.

3.2. Black–Scholes–Merton option pricing model

European put option is the hedging tool we use to hedge the potential downturn loss of our underlying assets. The price of European put option we need to pay in the beginning of hedging derives from B-S-M model:

$$\text{p} = \text{Ke}^{-rf*T}N(-d_2) - S_0N(-d_1)$$

S_0 is the current price of underlying asset, K is strike price we set for hedging and T is the hedging period we need to cover the down-turn risks. The d_1 and d_2 are written as:

$$d_1 = \frac{\ln\left(\frac{S_0}{K}\right) + \left(rf + \frac{\sigma^2}{2}\right) * T}{\sigma\sqrt{T}}, \qquad d_2 = d_1 - \sigma\sqrt{T}$$

and N is the cumulative density function of standard normal distribution. When it comes the end of the hedging period, $\text{p}_T = \max[\text{K} - \text{S}_T, 0]$, the profit we gain from hedging will be $\text{p}_T - p$.

3.3. Delta hedging

Delta hedging is a dynamic options trading strategy aiming to hedge the directional risk associated with price movements in the underlying asset. From B-S-M formula we already know option price is connected to the stock price, the partial derivative of option price to underlying asset price is:

$$\text{delta} = \frac{\partial p}{\partial S}$$

That means every 1 unit change in the stock price will bring about delta unit change in option price. The idea of delta hedging is to dynamic maintain the delta-neutral status of portfolio during the hedging process, in other words to we will keep $\frac{Shares\ of\ underlying\ assets}{shares\ of\ options} = delta$ and once the position

deviates away from this balance we will readjust our portfolio. (here we assume each share of option is against 1 share of underlying asset.) In this case, theoretically the value in portfolio will not be susceptible to the fluctuation of underlying asset,

$$\Delta\text{portfolio} = \Delta S * \text{ns} - \Delta p * \text{np} = \Delta S * \text{delta} * \text{np} - \Delta p * \text{np} = 0$$

This strategy could be feasible in coronavirus outbreak time since stock prices will be highly uncertain.

4. Data Preparation

4.1. Portfolio Components and Time Horizon

For the application of our methodology, we want to select three financial time series that their volatilities are not too low or even too high and similar to each other, considering that the ratio of each component for our optimal portfolio should not be too large or too small. We choose S&P500, Amazon and Boeing as our portfolio's components and download their historical daily price from Yahoo Finance.

To make sure that we our simulation and analysis using historical data well reflect what would happen in the future, we need to select a time horizon that contains enough extreme conditions, then we could get a relative reasonable simulation result. After trying some different time horizon, plotting and comparing among them, we finally choose time series data from 1997-05-15 to 2020-02-21 as historical population for normal period simulation and time series data from 1997-05-15 to 2020-04-13 as historical population for extreme period, since 1997-05-15 is the starting day of Amazon's data (the last starting day among three stocks). We set the normal trading period from 2020-01-02 to 2020-02-21 and the extreme trading period (COVID-19 period) from 2020-02-24 to 2020-04-13, so our portfolio trading periods are 35 days. We take the data from whole period (1997-05-15 to 2020-04-13) to calculate the stock's yearly volatility and take the stock prices one day before each trading period (2019-12-31 & 2020-02-21) as our stock start prices. We assume yearly risk-free return as 3%.

Figure1: Stock prices in two trading periods

4.2. Weight Calculation

We get log return for each component of our portfolio and calculate their return and volatility in order to get the optimal portfolio. Basic information (daily) of 3 returns are shown below:

	S&P500	Amazon	Boeing
Mean	0.000216	0.001070	0.000343
Std	0.011917	0.036090	0.019484

Table1: Stock daily return and volatility

According to the assumption of Markowitz Portfolio Theory, returns of all components should be normally distributed. Therefore, before calculating weights, we conduct normal test and 3 stocks are passed at a high significance level. Next, we randomly generate 1000 combinations for weights and find the one with the highest Sharpe-ratio. Below, we visualize our result of this generation. We plot Expected Return against Expected Volatility, and use the lighter color standing for higher Sharpe-ratio.

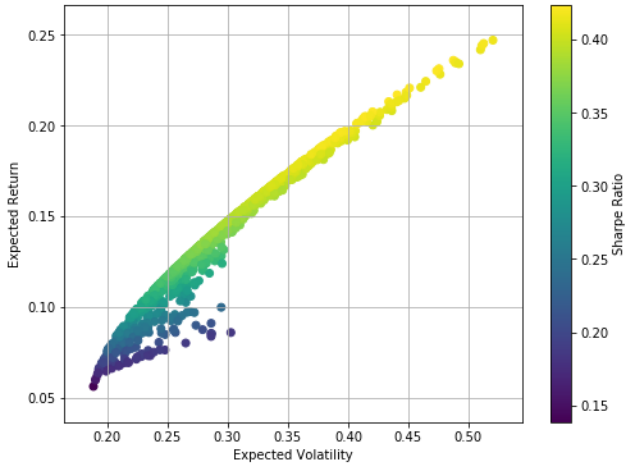

Figure2: 1000 randomly setting portfolios

We use “optimize” function to maximize sharpe ratio and get our optimal portfolio. However, we find that the portfolio doesn’t include S&P 500 due to its relative worse performance among three stocks. Besides, this portfolio is too optimal to give us a positive VaR during the normal period. This fact also proves that Markowitz Portfolio Theory is effective to minimize the basic risk at normal period. We then adjust our limit condition for our optimal portfolio - each weight in the range of (0.2,1), since that in Markowitz Portfolio Theory, every investor may have his/her own preference so not everyone can reach optimal. Our preference here is to include three stocks in our investment and each weight should be at least 20%. Based on our adjustment, we get our final optimal weights and the result is shown below.

Components	Optimal Weights	Optimal Weights after Adjustment
S&P500	0	0.2
Amazon	0.717035	0.6
Boeing	0.282965	0.2

Table2: Portfolio optimal weights

4.3. Time Series Preprocess

Prepared for conducting simulation, we want to further preprocess our data to make our stock returns not that volatile, considering that our single time of simulation (we intent to use boostrap method here) could be easily influenced by those extreme values in stock returns but this is actually not a common thing. For example, we may get a relative larger VaR in the extreme period. Therefore, we use rolling windows with a window size of 50, that is we use rolling mean for 50-day, to make a moving average towards our whole period of stock returns. We think 50-day is a suitable length to adjust the volatility but not to make it too smoother to get a better VaR. By using rolling mean as our new stock returns, we get a relative smoother time series plot and control extreme value to some extent.

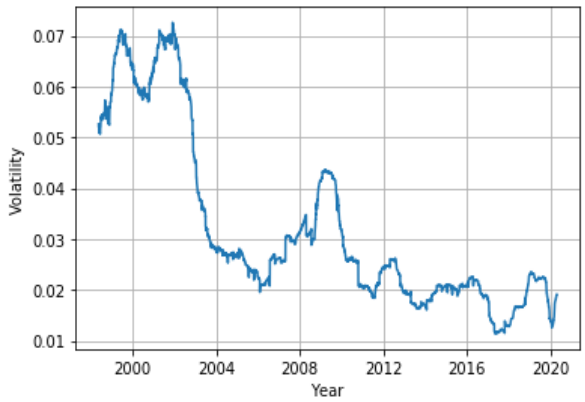

Figure3: Amazon's 250-day rolling return

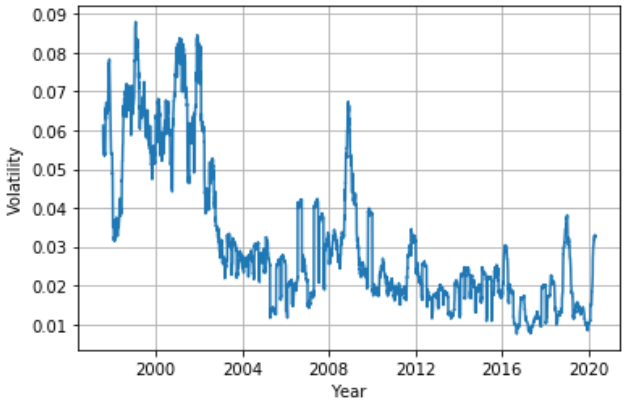

Figure4: Amazon's 50-day rolling return

5. Demonstration of VaR/ES Estimation

In this section, we present an empirical example with a hypothetical portfolio to demonstrate our model.

5.1. BSM model set-up for option valuation

- Price: stock start price
- Strike: price (at the money)
- Vol: stock yearly volatility
- rf: 3%

- tmat: 35/250

We use BSM model to estimate put option values of our stocks for both periods:

	Amazon	S&P500	Boeing
Normal Period	152.928	84.0484	14.2678
Extreme Period	173.463	86.8312	14.4720

Table3: Estimated put option values

From the table we can easily find that Amazon option has the highest price due to its highest stock price ($1847.84 at 2019.12.31), while Boeing is the opposite with the stock price of only $325.76 at the same time. Besides, all three option prices increased from normal period to extreme period due to a growth of their underlying assets. For example, Amazon's stock price reached $2095.97 at 2020.2.21.

Figure5: Stock price changes between two starting days

5.2. Portfolio set-up

We now set up the hypothetical portfolio and calculate its initial price.

- P0: 1000000
- Shares of stock: P0*w(stock)/startprice(stock)
- Shares of option: shares of underlying stock(100% covered)
- Initial price: P0+∑(Shares of option*Option put value)

In order to compare with different hedge results, we set up portfolios with various hedge degree:

Hedge Degree	Option Used for Hedge
No-hedged	No
Single-hedged	Amazon S&P500 Boeing
Double-hedged	Amazon+S&P500 Amazon + Boeing S&P + Boeing
Fully-hedged	Amazon + S&P500 + Boeing

Table4: Portfolios with different hedge degrees

5.3. VaR/ES Estimation of a Hypothetical Portfolio

We estimate VaR/ES using the bootstrap method to resample the rolling returns of each stock at both periods for many times to get the final stock price and calculate the corresponding option final value. Finally, based on the sum of stock values and option values, we can draw the distribution of final portfolio price and use our loss probability - ρ, to estimate VaR/ES of each portfolio.

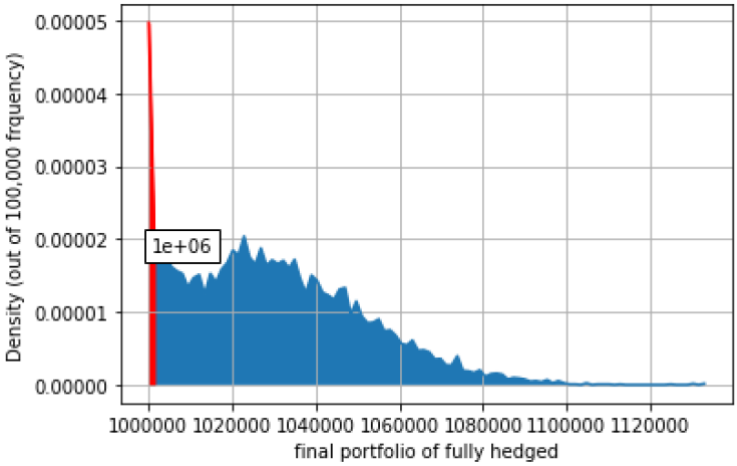

Figure6: Distribution of fully-hedged final portfolio prices

- nboot: 10000
- ρ: 0.01
- T0: the length of rolling return dataset
- Indexset: the index list from 0 to T0-1
- Bindex: for each boot randomly select 35 index number from indexset
- Stock final price: for each stock, find the corresponding stock return from rolling return dataset using bindex and then calculate final price using the following formula:

$$P_{t+h} = P_t e^{\sum_{j=1}^{h} r_{t+j}} = P_t \prod_{j=1}^{h} e^{r_{t+h}}$$

- Option final price: for each option, if the final underlying asset price is less than the strike price, we will take the difference between the strike price and the stock final price as option final price; otherwise, we will not execute the option and the final price will be zero. The formula of option final price is: Max(Strike-Stock final, 0)
- Portfolio final price: ∑(Shares of stock*Stock final price)+∑(Shares of option*Option final price)
- VaR: the opposite number of the difference between the 0.01 quantile for portfolio final price and the portfolio initial price; The formula of VaR estimation is: $-(F_{portfp\text{-}1}(0.01)\text{-portinitp})$
- ES: the difference between the portfolio initial price and the mean of portfolio final price less than the 0.01 quantile for portfolio final price; The formula of ES estimation is: $\text{portinitp-}E(\text{portfp}|\text{portfp} \leq F_{portfp\text{-}1}(0.01))$

5.4 VaR/ES Estimation Result Analysis

We start analyzing our estimation result in the normal period. We estimate our VaR/ES for no-hedged portfolio and single-hedged portfolio to pick up the best single-hedged portfolio.

	VaR	ES
No-hedged	19880.08	28112.97
Amazon	64727.6	66640.7
S&P500	21579.6	27751.5
Boeing	30452.4	38562.4

Table5: Normal period VaR/ES estimation of no-hedged and single-hedged portfolios

From the table above, we can clearly find that the VaR/ES don't decrease with stock hedged as we expect. We guess the reason might be that the expensive option price is larger than the risk we can hedge from the portfolio. On the other word, our hedge method is not very effective to hedge the risk. Also, the difference among increases of portfolios may be caused by the stock weights and different option prices. However, we still get some useful points. For example, all ES values are larger than VaR values, informing that ES can expose more extreme events.

Then we hope to analyze the results of double-hedged and fully-hedged portfolios to see whether higher degree of hedge can give us better result.

	VaR	ES
No-hedged	19880.08	28112.97
Amazon + S&P500	63805.2	65988.6
Amazon + Boeing	71735.5	71735.5
Fully	71735.5	71735.5

Table6: Normal period VaR/ES estimation of no-hedged, double-hedged and fully-hedged portfolios

The table above shows our partial result of double-hedged and fully-hedged portfolios which prove our former views. The VaR/ES are growing larger with higher degree of hedge since our option costs are increasing. Based on the normal situation, we think the extreme condition can only be worse with higher return volatility and we still can't find the actual hedge effects very clearly. We decide to exclude our option costs and estimate VaR/ES again.

6. Discussion for Changing Assumption

In this part, we try to exclude option costs in initial portfolio prices to see how option actually works for each of our hedging combinations. The first two columns show the normal period VaR/ES for each scenario we mentioned in last part and the last two columns show results from the extreme period:

	Normal period		Extreme period	
	VaR	**ES**	**VaR**	**ES**
No-hedged	19880.08	28112.97	29403.74	38335.94
Amazon	4913.82	6743.60	9724.11	12829.04
S&P500	22310.73	29720.14	25902.09	33242.23
Boeing	19440.30	26003.40	26093.76	34635.90
Amazon + S&P500	4372.97	6821.92	6471.46	8750.80
Amazon + Boeing	0.00	0.00	3296.04	4432.60
Fully	0.00	0.00	0.00	0.00

Table7: VaR/ES estimation of all portfolios without option prices in both periods

Totally speaking, our hedge strategy works very well as the hedge degree increases. VaR/ES gradually decreases among each column. Among three single-hedged portfolios, Amazon option has the best hedging performance since that Amazon stock has the largest weight in the portfolio. However, S&P 500 and Boeing have the same weight but different hedging performance, reflecting a relative higher volatility in Boeing's stock return. Also, their returns are also relative more stable in the normal period, and the hedges don't work very well. S&P500 shows an extremely bad hedging performance in the normal period due to its stable return - S&P500-hedged VaR even larger than no-hedged portfolio, from which we get the point that we don't need to hedge a single asset with low risk. Thus, in the extreme periods, both options start to work with their unstable returns. If we only choose one stock to hedge, Amazon would be the best one.

Then the difference VaR/ES between double-hedged portfolios of Amazon + S&P500 and Amazon + Boeing also tell us hedging riskier assets will get a better result. From the fully-hedged column, it's clearly that without considering option costs, fully-hedged can remove all the risk from the portfolio.

Figure7: Stock prices of Boeing and S&P500

We can also see from the table that in the extreme period, VaR and ES of no-hedged portfolio are larger than those in the normal period, proving a relative larger volatility of stock returns in the extreme period. VaR/ES of the historical data can help us detect the extreme events to some extent and once they include more memories, our forecast would be more precise.

By not considering option prices, we get a relative better hedge performance in both periods. Hedging strategy also works for the extreme period, which would greatly help us to take the risk management process in the extreme events. However, option price is an essential part of actual VaR/ES estimation that we can't omit. We need better hedging strategies instead of our current direct 100% covered method to hedge more risks worth more than our option costs to get a better VaR/ES estimation results of hedging process.

7. Conclusion and suggestion

In this article, we explored the effect of a varieties of option hedging strategies in normal period and coronavirus period separately with 50 days moving average bootstrapping simulation. Though counter-intuitive, the VaR and ES of fully hedged and partially hedged portfolio tend to be higher than no hedged scenario, which could be explained by the high initial cost of in-the-money put option. Besides, since we are hedging with European option that cannot be executed before maturity, the time value of option fades gradually with the proximity of maturity while the inner value of option may rise first due to coronavirus shock but then falls because the US government and Federal Reserve have launched a series of financial stimulus packages to save the economy and stock market from slipping into recession.

There remains a lot of space for us to improve our hedging strategy and further our research. First, instead of simply following a static hedging strategy and using European put option, more dynamic forms of hedging strategies could be developed with the combination of put and call option. For example, we could integrate the indicators like MACD, KDJ and BOLL from technical analysis theory as the signal to adjust our position, so that we are able to reap the profit from stocks/option in advance rather than waiting until maturity. And a couple of hedging strategies like strip hedging could be utilized in the market with tremendous move and high downturn risk.

Besides, we do try to optimize our portfolio under maximized Sharpe ratio paradigm, but other things could also be optimized as well. For instance, the strike price we choose is the same as the beginning price of our stock, which represents an expansive option price behind that. One reason our hedge is not as good as idealistic is due to the high hedging cost of in-the-money option. In the end of our code, we attach delta hedging method programming, while it turns out to be more costly than the other methods. To have a n share stocks we need to have n units of options to fully hedge the risk, but the number for delta hedging needed is n/-delta, which will definitely generate a larger expense if we still choose the same strike price. With that said, the mix of out-of-money and in-the-money options could substitute pure in-the-money ones.

In the end, our hedging strategy and programming could be more flexible and widespread applicable to more cases in reality. We discuss the scenario in normal period and coronavirus shocked period, but a good hedging strategy should be back-tested by similar scenarios in history and run through the whole event period. Unfortunately, on the one hand the coronavirus is not yet ended, the stock market resurges 30% back from the lowest point but still no one knows what could happen next. On the other hand, virus outbreak is a rather rare happened event that gives us less historical data and trend to learn from and the situation is sensitive to the infected situation across the world. In the future we could extend this strategy to a rare-event-triggered hedging strategy that not limitedly reacts to pandemic issues, instead it fits for more generalized forms of shocks.

References

1. Fan, J. , Gu, J. , 2003. Semiparametric estimation of Value at Risk. Econ. Theory 24, 1404–1424 .
2. Ahn D H, Boudoukh J, Richardson M, et al. Optimal risk management using options[J]. The Journal of Finance, 1999, 54(1): 359-375.
3. Wilhelmsson, A. , 2009. Value at risk with time varying variance, skewness and kurtosis-the NIG-ACD Model. Econometrics J. 12, 82–104 .
4. Jánský I., Rippel M. 2011. Value at Risk Forecasting with the ARMA-GARCH Family of Models in Times of Increased Volatility. IES. Working Paper No. 27/2011.
5. An R, Wang D, Huang M, et al. Portfolio Optimization Using Period Value at Risk Based on Historical Simulation Method[C]//2019 Chinese Control And Decision Conference (CCDC). IEEE, 2019: 324-328.
6. Cong J, Tan K S, Weng C. VaR-based optimal partial hedging[J]. ASTIN Bulletin: The Journal of the IAA, 2013, 43(3): 271-299.
7. Chen R, Yu L. A novel nonlinear value-at-risk method for modeling risk of option portfolio with multivariate mixture of normal distributions[J]. Economic Modelling, 2013, 35: 796-804.
8. LE Ghaoui, M Oks, F Oustry. (2003). Worst-Case Value-At-Risk and Robust Portfolio Optimization: A Conic Programming Approach. *Operations research.* https://doi.org/10.1287/opre.51.4.543.16101
9. J Hull, A White. (2017). Optimal delta hedging for options. *Journal of Banking & Finance.* https://www.sciencedirect.com/science/article/pii/S0378426617301085
10. Y El-Khatib, A Hatemi-J. (2017). Option valuation and hedging in markets with a crunch. *Journal of Economic Studies.* https://www.emerald.com/insight/content/doi/10.1108/JES-04-2016-0083/full/html
11. Python for Finance. (2019). Investment portfolio optimization with python. https://www.pythonforfinance.net/2019/07/02/investment-portfolio-optimisation-with-python-revisited/

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 24 17:49:55 2020

@author: Group 7
"""
# load packages
import pandas as pd
import numpy  as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import scipy.optimize as optm

# In[]
# load data and create data frame
sp = pd.read_csv("SP500.csv")
sp = sp[['Date','Close']]
sp.index = pd.to_datetime(sp.Date,format="%m/%d/%Y")
del sp['Date']

amzn = pd.read_csv("AMZN.csv")
amzn = amzn[['Date','Close']]
amzn.index = pd.to_datetime(amzn.Date,format="%m/%d/%Y")
del amzn['Date']

#######################################
####### bond here is Boeing stock price
bond = pd.read_csv('BOND.csv')
bond = bond[['Date','Close']]
bond.index = pd.to_datetime(bond.Date,format="%m/%d/%Y")
del bond['Date']

# In[]
#aa = pd.read_csv('AAL.csv')
#aa = aa[['Date','Close']]
#aa.index = pd.to_datetime(aa.Date,format="%m/%d/%Y")
#del aa['Date']

# check for amazon's volatility
amznnew = amzn.copy()
amznnew['lagclose'] = amznnew.Close.shift(1)
amznnew['ret']=np.log(amznnew['Close'])-np.log(amznnew['lagclose'])
amznnew = amznnew[1:]

amznnewretVec = amznnew['ret'].values
amznnewmean = np.mean(amznnewretVec)
amznnewstd = np.std(amznnewretVec)

amznnew['retv']=(amznnew['ret']-amznnewmean)**2.
amznnewrollwindow = amznnew.rolling(window=50,win_type="boxcar")
amznnewrollmean = amznnewrollwindow.mean()
amznnewrollmean['retsd'] = np.sqrt(amznnewrollmean['retv'])

plt.plot(amznnewrollmean['retsd'])
plt.grid()
plt.xlabel("Year")
plt.ylabel('Volatility')

# In[]
spa = sp.copy()
spa = spa[(spa.index >= '1997-05-15') & (spa.index <= '2019-12-31')]

amzna = amzn.copy()
amzna = amzna[(amzna.index >= '1997-05-15') & (amzna.index <= '2019-12-31')]

bonda = bond.copy()
bonda = bonda[(bonda.index >= '1997-05-15') & (bonda.index <= '2019-12-31')]
```

```

#aaa = aa.copy()
#aaa = aaa[(aaa.index >= '2005-01-01') & (aaa.index <= '2019-12-31')]

data = pd.DataFrame({"sp":spa.Close,
                     "amzn":amzna.Close,
                     "bond":bonda.Close})
data = data.dropna()
data.head()

ret = np.log(data.pct_change()+1)
ret = ret.dropna()


# In[]
# test normality of returns
def normality_test(a):
    print('Norm test p-value %14.3f' % stats.normaltest(a)[1])

for i in ["sp","amzn","bond"]:
    print('\nResults for {}'.format(i))
    print('-'*32)
    log_data = np.array(ret[i])
    normality_test(log_data)

# In[]
# calculate weights
pret = []
pvar = []
prs = []
prt = []
for p in range(1000):
    weights = np.random.random(3)
    weights /=np.sum(weights)
    return1=np.sum(ret.mean()*250*weights)
    std1=np.sqrt(np.dot(weights.T, np.dot(ret.cov()*250, weights)))
    Rs=stats.norm.ppf(0.01,loc=return1,scale=std1)
    Rt = -std1*stats.norm.pdf((Rs-return1)/std1)/0.01+return1
    pret.append(return1)
    pvar.append(std1)
    prs.append(Rs)
    prt.append(Rt)

pret = np.array(pret)
pvar = np.array(pvar)
prs = np.array(prs)
prt = np.array(prt)
# plot random 1000 portfolio set: return against volatitliy
rf = 0.03
plt.figure(figsize=(8, 6))
plt.scatter(pvar, pret, c=(pret-rf)/pvar, marker = 'o')
plt.grid(True)
plt.xlabel('Expected Volatility')
plt.ylabel('Expected Return')
plt.colorbar(label = 'Sharpe Ratio')


# In[]
# weights calculation
def statsRecord(weights):
    weights = np.array(weights)
    pret = np.sum(ret.mean()*weights)*250
    pvar = np.sqrt(np.dot(weights.T, np.dot(ret.cov()*250,weights)))
    sharpe = (pret-rf)/pvar
    return  sharpe

# minimum sharpe
def max_sharpe(weights):
    return -statsRecord(weights)

```

```
# Initialize
x0 = 3*[1./3]

#权重（某股票持仓比例）限制在0和1之间。
bnds1 = tuple((0,1) for x in range(3))
cons = ({'type':'eq', 'fun':lambda x: np.sum(x)-1})

optv = optm.minimize(max_sharpe,
                     x0,
                     method = 'SLSQP',
                     bounds = bnds1,
                     constraints = cons)
w = optv['x']
# In[]
# now start for Normal period: 2020-01-02 to 2020-02-21
# get rolling return

spb = sp.copy()
spb = spb[(spb.index >= '1997-05-15') & (spb.index <= '2020-02-21')]

amznb = amzn.copy()
amznb = amznb[(amznb.index >= '1997-05-15') & (amznb.index <= '2020-02-21')]

#googb = goog.copy()
#googb = googb[(googb.index >= '2005-01-01') & (googb.index <= '2020-02-21')]

bondb = bond.copy()
bondb = bondb[(bondb.index >= '1997-05-15') & (bondb.index <= '2020-02-21')]

#aab = aa.copy()
#aab = aab[(aab.index >= '2010-01-01') & (aab.index <= '2020-02-21')]

datanormal = pd.DataFrame({"sp":spb.Close,"amzn":amznb.Close,"bond":bondb.Close})
#datanormal = pd.DataFrame({"sp":spb.Close,"amzn":amznb.Close,"bond":bondb.Close})
datanormal = datanormal.dropna()
datanormal.head()

datanormal['splag'] = datanormal.sp.shift(1)
datanormal['spret']=np.log(datanormal['sp'])-np.log(datanormal['splag'])
datanormal['amznlag'] = datanormal.amzn.shift(1)
datanormal['amznret']=np.log(datanormal['amzn'])-np.log(datanormal['amznlag'])
datanormal['bondlag'] = datanormal.bond.shift(1)
datanormal['bondret']=np.log(datanormal['bond'])-np.log(datanormal['bondlag'])
#datanormal['aalag'] = datanormal.aa.shift(1)
#datanormal['aaret']=np.log(datanormal['aa'])-np.log(datanormal['aalag'])
datanormal = datanormal[1:]

rollwindow = datanormal.rolling(window=50,win_type="boxcar").mean()
rollwindow = rollwindow[50:]

T0 = len(rollwindow)

sprollingret = rollwindow['spret'].values
amznrollingret = rollwindow['amznret'].values
bondrollingret = rollwindow['bondret'].values
#aarollingret = rollwindow['aaret'].values


# In[]
# now start for extreme period: 2020-02-23 to 2020-04-13
# get rolling return


spc = sp.copy()
spc = spc[(spc.index >= '1997-05-15') & (spc.index <= '2020-04-13')]

amznc = amzn.copy()
amznc = amznc[(amznc.index >= '1997-05-15') & (amznc.index <= '2020-04-13')]

#googc = goog.copy()
```

```
#googc = googc[(googc.index >= '2005-01-01') & (googc.index <= '2020-02-21')]

bondc = bond.copy()
bondc = bondc[(bondc.index >= '1997-05-15') & (bondc.index <= '2020-04-13')]

#aab = aa.copy()
#aab = aab[(aab.index >= '2010-01-01') & (aab.index <= '2020-02-21')]

dataExtreme = pd.DataFrame({"sp":spc.Close,"amzn":amznc.Close,"bond":bondc.Close})
#datanormal = pd.DataFrame({"sp":spb.Close,"amzn":amznb.Close,"bond":bondb.Close})
dataExtreme = dataExtreme.dropna()
dataExtreme.head()

dataExtreme['splag'] = dataExtreme.sp.shift(1)
dataExtreme['spret']=np.log(dataExtreme['sp'])-np.log(dataExtreme['splag'])
dataExtreme['amznlag'] = dataExtreme.amzn.shift(1)
dataExtreme['amznret']=np.log(dataExtreme['amzn'])-np.log(dataExtreme['amznlag'])
dataExtreme['bondlag'] = dataExtreme.bond.shift(1)
dataExtreme['bondret']=np.log(dataExtreme['bond'])-np.log(dataExtreme['bondlag'])
#dataExtreme['aalag'] = dataExtreme.aa.shift(1)
#dataExtreme['aaret']=np.log(dataExtreme['aa'])-np.log(dataExtreme['aalag'])
dataExtreme = dataExtreme[1:]

rollwindow2 = dataExtreme.rolling(window=50,win_type="boxcar").mean()
rollwindow2 = rollwindow2[50:]
T02 = len(rollwindow2)

sprollingret2 = rollwindow2['spret'].values
amznrollingret2 = rollwindow2['amznret'].values
bondrollingret2 = rollwindow2['bondret'].values
#aarollingret = rollwindow['aaret'].values

# In[]
# estimate option price

# assets mean and std
spreturn = ret['sp'].values
amznreturn = ret['amzn'].values
bondreturn = ret['bond'].values
# mean
spmean = np.mean(spreturn)
spstd = np.std(spreturn)
amznmean = np.mean(amznreturn)
# std
amznstd = np.std(amznreturn)
bondmean = np.mean(bondreturn)
bondstd = np.std(bondreturn)

# year volatility
spyear = np.sqrt(250.)*spstd
amznyear = np.sqrt(250.)*amznstd
bondyear = np.sqrt(250.)*bondstd

# BSM Model
def callput(price,strike,vol,rf,tmat):
    d1 = (np.log(price/strike)+(rf+vol*vol/2.)*tmat)/(vol*np.sqrt(tmat))
    d2 = d1-vol*np.sqrt(tmat)
    nd1 = stats.norm.cdf(d1)
    nnd1 = stats.norm.cdf(-d1) #n(-d1)
    nd2 = stats.norm.cdf(d2)
    nnd2 = stats.norm.cdf(-d2) #n(-d2)

    cval = price*nd1 - strike*np.exp(-rf*tmat)*nd2
    pval = strike*np.exp(-rf*tmat)*nnd2-price*nnd1
    delta = nd1-1         #put option should be nd1-1
    return cval, pval, delta

# theoretical option value at time t
# Normal time
# amazon
```

```
amznprice = amzna.copy()
amznstartprice = amznprice.iloc[-1]
amznstrike = amznstartprice
h = 35.
T = h/250.
amzncval, amznpval, amzndelta = callput(amznstartprice,amznstrike,amznyear,rf,T)

# sp
spprice = spa.copy()
spstartprice = spprice.iloc[-1]
spstrike = spstartprice
h = 35.
T = h/250.
spcval, sppval, spdelta = callput(spstartprice,spstrike,spyear,rf,T)

# bond
bondprice = bonda.copy()
bondstartprice = bondprice.iloc[-1]
bondstrike = bondstartprice
h = 35.
T = h/250.
bondcval, bondpval, bonddelta = callput(bondstartprice,bondstrike,bondyear,rf,T)

# Extreme time
# amazon
amznprice2 = amznb.copy()
amznstartprice2 = amznprice2.iloc[-1]
amznstrike2 = amznstartprice2
h = 35.
T = h/250.
amzncval2, amznpval2, amzndelta2 = callput(amznstartprice2,amznstrike2,amznyear,rf,T)

# sp
spprice2 = spb.copy()
spstartprice2 = spprice2.iloc[-1]
spstrike2 = spstartprice2
h = 35.
T = h/250.
spcval2, sppval2, spdelta2 = callput(spstartprice2,spstrike2,spyear,rf,T)

# bond
bondprice2 = bondb.copy()
bondstartprice2 = bondprice2.iloc[-1]
bondstrike2 = bondstartprice2
h = 35.
T = h/250.
bondcval2, bondpval2, bonddelta2 = callput(bondstartprice2,bondstrike2,bondyear,rf,T)

# In[]
# NO HEDGE
# estimate VaR

# set up portfolio for no hedge
# normal

P0 = 1000000.
sharessp = P0*w[0]/spstartprice
sharesamzn = P0*w[1]/amznstartprice
sharesbond = P0*w[2]/bondstartprice

# extreme
sharessp2 = P0*w[0]/spstartprice2
sharesamzn2 = P0*w[1]/amznstartprice2
sharesbond2 = P0*w[2]/bondstartprice2

initportv = P0

# In[]
###################################
# Normal period
```

```
indexset = range(T0)
nboot = 10000
portval = np.zeros(nboot)
p = 0.01

for i in range(nboot):
    bindex = np.random.choice(indexset,size=35)
    spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice[0]
    amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice[0]
    bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice[0]
    portval[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal

VaR_noHedge_n = -(np.percentile(portval,100.*p)-initportv)
esP_noHedge_n = initportv-np.mean(portval[portval<=np.percentile(portval,100.*p)])


###################################
# Extreme period
indexset2 = range(T02)
portval2 = np.zeros(nboot)

for i in range(nboot):
    bindex = np.random.choice(indexset2,size=35)
    spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2[0]
    amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2[0]
    bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2[0]
    portval2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal

VaR_noHedge_e = -(np.percentile(portval2,100.*p)-initportv)
esP_noHedge_e = initportv-np.mean(portval2[portval2<=np.percentile(portval2,100.*p)])

# In[]
# FULLY HEDGED for three stocks
# estimate VaR

# calculate VaR
initportvF = P0+amznpval*sharesamzn+sppval*sharessp+bondpval*sharesbond
initportvF2 = P0+amznpval2*sharesamzn2+sppval2*sharessp2+bondpval2*sharesbond2

###################################
# normal period
indexset = range(T0)
nboot = 10000
portvalF = np.zeros(nboot)

p = 0.01

for i in range(nboot):
    bindex = np.random.choice(indexset,size=35)
    spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice
    amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
    bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
    amznoptval = np.maximum(amznstrike-amznfinal,0.)
    spoptval = np.maximum(spstrike-spfinal,0.)
    bondoptval = np.maximum(bondstrike-bondfinal,0.)
    portvalF[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal+sharesamzn*

VaR_Fully_n = -(np.percentile(portvalF,100.*p)-initportvF)
esP_fully_n = initportvF-np.mean(portvalF[portvalF<=np.percentile(portvalF,100.*p)])


###################################
# Extreme period
indexset2 = range(T02)
portvalF2 = np.zeros(nboot)

for i in range(nboot):
    bindex = np.random.choice(indexset2,size=35)
    spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2
    amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
```

```
    bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
    amznoptval = np.maximum(amznstrike2-amznfinal,0.)
    spoptval = np.maximum(spstrike2-spfinal,0.)
    bondoptval = np.maximum(bondstrike2-bondfinal,0.)
    portvalF2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal+sharesa


VaR_Fully_e = -(np.percentile(portvalF2,100.*p)-initportvF2)
esP_fully_e = initportvF2-np.mean(portvalF2[portvalF2<=np.percentile(portvalF2,100.*p)])

# In[]
# PARTIAL HEDGED - only sp500
# estimate VaR

# calculate VaR
initportvSP = P0+sppval*sharessp
initportvSP2 = P0+sppval2*sharessp2

###################################
# normal period
indexset = range(T0)
nboot = 10000
portvalSP = np.zeros(nboot)
p = 0.01

for i in range(nboot):
    bindex = np.random.choice(indexset,size=35)
    spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice
    amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
    bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
    spoptval = np.maximum(spstrike-spfinal,0.)
    portvalSP[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal+sharessp*s

VaR_sp_n = -(np.percentile(portvalSP,100.*p)-initportvSP)
esP_sp_n = initportvSP-np.mean(portvalSP[portvalSP<=np.percentile(portvalSP,100.*p)])

###################################
# Extreme period
indexset2 = range(T02)
portvalSP2 = np.zeros(nboot)

for i in range(nboot):
    bindex = np.random.choice(indexset2,size=35)
    spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2
    amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
    bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
    spoptval = np.maximum(spstrike2-spfinal,0.)
    portvalSP2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal+shares

VaR_sp_e = -(np.percentile(portvalSP2,100.*p)-initportvSP2)
esP_sp_e = initportvSP2-np.mean(portvalSP2[portvalSP2<=np.percentile(portvalSP2,100.*p)])

# In[]
# PARTIAL HEDGED - only amzn
# estimate VaR

# calculate VaR
initportvAMZN = P0+amznpval*sharesamzn
initportvAMZN2 = P0+amznpval2*sharesamzn2

###################################
# normal period
indexset = range(T0)
nboot = 10000
portvalAMZN = np.zeros(nboot)

p = 0.01

for i in range(nboot):
    bindex = np.random.choice(indexset,size=35)
```

```
    spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice
    amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
    bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
    amznoptval = np.maximum(amznstrike-amznfinal,0.)
    portvalAMZN[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal+sharesam

VaR_amzn_n = -(np.percentile(portvalAMZN,100.*p)-initportvAMZN)
esP_amzn_n = initportvAMZN-
np.mean(portvalAMZN[portvalAMZN<=np.percentile(portvalAMZN,100.*p)])

###################################
# Extreme period
indexset2 = range(T02)
portvalAMZN2 = np.zeros(nboot)

for i in range(nboot):
    bindex = np.random.choice(indexset2,size=35)
    spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2
    amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
    bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
    amznoptval = np.maximum(amznstrike2-amznfinal,0.)
    portvalAMZN2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal+shar

VaR_amzn_e = -(np.percentile(portvalAMZN2,100.*p)-initportvAMZN2)
esP_amzn_e = initportvAMZN2-
np.mean(portvalAMZN2[portvalAMZN2<=np.percentile(portvalAMZN2,100.*p)])

# In[]
# PARTIAL HEDGED - only bond
# estimate VaR

# calculate VaR
initportvBOND = P0+bondpval*sharesbond
initportvBOND2 = P0+bondpval2*sharesbond2

###################################
# normal period
indexset = range(T0)
nboot = 10000
portvalBOND = np.zeros(nboot)
p = 0.01

for i in range(nboot):
    bindex = np.random.choice(indexset,size=35)
    spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice
    amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
    bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
    bondoptval = np.maximum(bondstrike-bondfinal,0.)
    portvalBOND[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal+sharesbc

VaR_bond_n = -(np.percentile(portvalBOND,100.*p)-initportvBOND)
esP_bond_n = initportvBOND-
np.mean(portvalBOND[portvalBOND<=np.percentile(portvalBOND,100.*p)])

###################################
# Extreme period
indexset2 = range(T02)
portvalBOND2 = np.zeros(nboot)

for i in range(nboot):
    bindex = np.random.choice(indexset2,size=35)
    spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2
    amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
    bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
    bondoptval = np.maximum(bondstrike2-bondfinal,0.)
    portvalBOND2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal+shar

VaR_bond_e = -(np.percentile(portvalBOND2,100.*p)-initportvBOND2)
esP_bond_e = initportvBOND2-
np.mean(portvalBOND2[portvalBOND2<=np.percentile(portvalBOND2,100.*p)])
```

```
555.
556.  # In[]
557.  # PARTIAL HEDGED - sp500+amzn
558.  # estimate VaR
559.
560.  # calculate VaR
561.  initportvSPAMZN = P0+sppval*sharessp+amznpval*sharesamzn
562.  initportvSPAMZN2 = P0+sppval2*sharessp2+amznpval2*sharesamzn2
563.
564.  ###################################
565.  # normal period
566.  indexset = range(T0)
567.  nboot = 10000
568.  portvalSPAMZN = np.zeros(nboot)
569.  p = 0.01
570.
571.  for i in range(nboot):
572.      bindex = np.random.choice(indexset,size=35)
573.      spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice
574.      amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
575.      bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
576.      spoptval = np.maximum(spstrike-spfinal,0.)
577.      amznoptval = np.maximum(amznstrike-amznfinal,0.)
578.      portvalSPAMZN[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal+shares
579.  VaR_spamzn_n = -(np.percentile(portvalSPAMZN,100.*p)-initportvSPAMZN)
580.  esP_spamzn_n = initportvSPAMZN-
      np.mean(portvalSPAMZN[portvalSPAMZN<=np.percentile(portvalSPAMZN,100.*p)])
581.
582.  ###################################
583.  # Extreme period
584.  indexset2 = range(T02)
585.  portvalSPAMZN2 = np.zeros(nboot)
586.  for i in range(nboot):
587.      bindex = np.random.choice(indexset2,size=35)
588.      spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2
589.      amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
590.      bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
591.      spoptval = np.maximum(spstrike2-spfinal,0.)
592.      amznoptval = np.maximum(amznstrike2-amznfinal,0.)
593.      portvalSPAMZN2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal+sh
594.  VaR_spamzn_e = -(np.percentile(portvalSPAMZN2,100.*p)-initportvSPAMZN2)
595.  esP_spamzn_e = initportvSPAMZN2-
      np.mean(portvalSPAMZN2[portvalSPAMZN2<=np.percentile(portvalSPAMZN2,100.*p)])
596.
597.
598.
599.  # In[]
600.  # PARTIAL HEDGED - amzn+bond
601.  # estimate VaR
602.
603.  # calculate VaR
604.  initportvAMZNBOND = P0+amznpval*sharesamzn+bondpval*sharesbond
605.  initportvAMZNBOND2 = P0+bondpval2*sharesbond2+amznpval2*sharesamzn2
606.
607.  ###################################
608.  # normal period
609.  indexset = range(T0)
610.  nboot = 10000
611.  portvalAMZNBOND = np.zeros(nboot)
612.  p = 0.01
613.
614.  for i in range(nboot):
615.      bindex = np.random.choice(indexset,size=35)
616.      spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice
617.      amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
618.      bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
619.      bondoptval = np.maximum(bondstrike-bondfinal,0.)
620.      amznoptval = np.maximum(amznstrike-amznfinal,0.)
621.      portvalAMZNBOND[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal+shar
622.  VaR_amznbond_n = -(np.percentile(portvalAMZNBOND,100.*p)-initportvAMZNBOND)
```

```
esP_amznbond_n = initportvAMZNBOND-
np.mean(portvalAMZNBOND[portvalAMZNBOND<=np.percentile(portvalAMZNBOND,100.*p)])

###################################
# Extreme period
indexset2 = range(T02)
portvalAMZNBOND2 = np.zeros(nboot)
for i in range(nboot):
    bindex = np.random.choice(indexset2,size=35)
    spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2
    amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
    bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
    bondoptval = np.maximum(bondstrike2-bondfinal,0.)
    amznoptval = np.maximum(amznstrike2-amznfinal,0.)
    portvalAMZNBOND2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal+
VaR_amznbond_e = -(np.percentile(portvalAMZNBOND2,100.*p)-initportvAMZNBOND2)
esP_amznbond_e = initportvAMZNBOND2-
np.mean(portvalAMZNBOND2[portvalAMZNBOND2<=np.percentile(portvalAMZNBOND2,100.*p)])



# In[]
# exclude option cost
# FULLY HEDGED for three stocks
# estimate VaR

# calculate VaR
initportvF = P0
initportvF2 = P0
###################################
# normal period
indexset = range(T0)
nboot = 10000
portvalF = np.zeros(nboot)

p = 0.01

for i in range(nboot):
    bindex = np.random.choice(indexset,size=35)
    spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice
    amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
    bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
    amznoptval = np.maximum(amznstrike-amznfinal,0.)
    spoptval = np.maximum(spstrike-spfinal,0.)
    bondoptval = np.maximum(bondstrike-bondfinal,0.)
    portvalF[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal+sharesamzn*

VaR_Fully_n = -(np.percentile(portvalF,100.*p)-initportvF)
esP_fully_n = initportvF-np.mean(portvalF[portvalF<=np.percentile(portvalF,100.*p)])


###################################
###################################
# Extreme period
indexset2 = range(T02)
portvalF2 = np.zeros(nboot)

for i in range(nboot):
    bindex = np.random.choice(indexset2,size=35)
    spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2
    amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
    bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
    amznoptval = np.maximum(amznstrike2-amznfinal,0.)
    spoptval = np.maximum(spstrike2-spfinal,0.)
    bondoptval = np.maximum(bondstrike2-bondfinal,0.)
    portvalF2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal+sharesa

VaR_Fully_e = -(np.percentile(portvalF2,100.*p)-initportvF2)
esP_fully_e = initportvF2-np.mean(portvalF2[portvalF2<=np.percentile(portvalF2,100.*p)])

```

```
691.
692.  # In[]
693.  # PARTIAL HEDGED - only sp500
694.  # estimate VaR
695.
696.  # calculate VaR
697.  initportvSP = P0
698.  initportvSP2 = P0
699.  ###################################
700.  # normal period
701.  indexset = range(T0)
702.  nboot = 10000
703.  portvalSP = np.zeros(nboot)
704.
705.  p = 0.01
706.
707.  for i in range(nboot):
708.      bindex = np.random.choice(indexset,size=35)
709.      spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice
710.      amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
711.      bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
712.      spoptval = np.maximum(spstrike-spfinal,0.)
713.      portvalSP[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal+sharessp*s
714.
715.  VaR_sp_n = -(np.percentile(portvalSP,100.*p)-initportvSP)
716.  esP_sp_n = initportvSP-np.mean(portvalSP[portvalSP<=np.percentile(portvalSP,100.*p)])
717.
718.  ###################################
719.  # Extreme period
720.  indexset2 = range(T02)
721.  portvalSP2 = np.zeros(nboot)
722.
723.  for i in range(nboot):
724.      bindex = np.random.choice(indexset2,size=35)
725.      spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2
726.      amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
727.      bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
728.      spoptval = np.maximum(spstrike2-spfinal,0.)
729.      portvalSP2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal+shares
730.
731.  VaR_sp_e = -(np.percentile(portvalSP2,100.*p)-initportvSP2)
732.  esP_sp_e = initportvSP2-np.mean(portvalSP2[portvalSP2<=np.percentile(portvalSP2,100.*p)])
733.
734.  # In[]
735.  # PARTIAL HEDGED - only amzn
736.  # estimate VaR
737.
738.  # calculate VaR
739.  initportvAMZN = P0
740.  initportvAMZN2 = P0
741.  ###################################
742.  # normal period
743.  indexset = range(T0)
744.  nboot = 10000
745.  portvalAMZN = np.zeros(nboot)
746.
747.  p = 0.01
748.
749.  for i in range(nboot):
750.      bindex = np.random.choice(indexset,size=35)
751.      spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice
752.      amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
753.      bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
754.      amznoptval = np.maximum(amznstrike-amznfinal,0.)
755.      portvalAMZN[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal+sharesam
756.
757.  VaR_amzn_n = -(np.percentile(portvalAMZN,100.*p)-initportvAMZN)
758.  esP_amzn_n = initportvAMZN-
      np.mean(portvalAMZN[portvalAMZN<=np.percentile(portvalAMZN,100.*p)])
759.
```

```
###################################
# Extreme period
indexset2 = range(T02)
portvalAMZN2 = np.zeros(nboot)

for i in range(nboot):
    bindex = np.random.choice(indexset2,size=35)
    spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2
    amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
    bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
    amznoptval = np.maximum(amznstrike2-amznfinal,0.)
    portvalAMZN2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal+shar

VaR_amzn_e = -(np.percentile(portvalAMZN2,100.*p)-initportvAMZN2)
esP_amzn_e = initportvAMZN2-
np.mean(portvalAMZN2[portvalAMZN2<=np.percentile(portvalAMZN2,100.*p)])

# In[]
# PARTIAL HEDGED - only bond
# estimate VaR

# calculate VaR
initportvBOND = P0
initportvBOND2 = P0
###################################
# normal period
indexset = range(T0)
nboot = 10000
portvalBOND = np.zeros(nboot)
p = 0.01

for i in range(nboot):
    bindex = np.random.choice(indexset,size=35)
    spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice
    amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
    bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
    bondoptval = np.maximum(bondstrike-bondfinal,0.)
    portvalBOND[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal+sharesbc

VaR_bond_n = -(np.percentile(portvalBOND,100.*p)-initportvBOND)
esP_bond_n = initportvBOND-
np.mean(portvalBOND[portvalBOND<=np.percentile(portvalBOND,100.*p)])

###################################
# Extreme period
indexset2 = range(T02)
portvalBOND2 = np.zeros(nboot)

for i in range(nboot):
    bindex = np.random.choice(indexset2,size=35)
    spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2
    amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
    bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
    bondoptval = np.maximum(bondstrike2-bondfinal,0.)
    portvalBOND2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal+shar

VaR_bond_e = -(np.percentile(portvalBOND2,100.*p)-initportvBOND2)
esP_bond_e = initportvBOND2-
np.mean(portvalBOND2[portvalBOND2<=np.percentile(portvalBOND2,100.*p)])


# In[]
# PARTIAL HEDGED - sp500+amzn
# estimate VaR

# calculate VaR
initportvSPAMZN = P0
initportvSPAMZN2 = P0
###################################
# normal period
```

```
indexset = range(T0)
nboot = 10000
portvalSPAMZN = np.zeros(nboot)
p = 0.01

for i in range(nboot):
    bindex = np.random.choice(indexset,size=35)
    spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice
    amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
    bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
    spoptval = np.maximum(spstrike-spfinal,0.)
    amznoptval = np.maximum(amznstrike-amznfinal,0.)
    portvalSPAMZN[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal+shares
VaR_spamzn_n = -(np.percentile(portvalSPAMZN,100.*p)-initportvSPAMZN)
esP_spamzn_n = initportvSPAMZN-
np.mean(portvalSPAMZN[portvalSPAMZN<=np.percentile(portvalSPAMZN,100.*p)])

###################################
# Extreme period
indexset2 = range(T02)
portvalSPAMZN2 = np.zeros(nboot)
for i in range(nboot):
    bindex = np.random.choice(indexset2,size=35)
    spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2
    amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
    bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
    spoptval = np.maximum(spstrike2-spfinal,0.)
    amznoptval = np.maximum(amznstrike2-amznfinal,0.)
    portvalSPAMZN2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal+sh
VaR_spamzn_e = -(np.percentile(portvalSPAMZN2,100.*p)-initportvSPAMZN2)
esP_spamzn_e = initportvSPAMZN2-
np.mean(portvalSPAMZN2[portvalSPAMZN2<=np.percentile(portvalSPAMZN2,100.*p)])

# In[]
# PARTIAL HEDGED - amzn+bond
# estimate VaR

# calculate VaR
initportvAMZNBOND = P0
initportvAMZNBOND2 = P0
###################################
# normal period
indexset = range(T0)
nboot = 10000
portvalAMZNBOND = np.zeros(nboot)
p = 0.01

for i in range(nboot):
    bindex = np.random.choice(indexset,size=35)
    spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice
    amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
    bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
    bondoptval = np.maximum(bondstrike-bondfinal,0.)
    amznoptval = np.maximum(amznstrike-amznfinal,0.)
    portvalAMZNBOND[i] = sharessp*spfinal + sharesamzn*amznfinal + sharesbond*bondfinal+shar
VaR_amznbond_n = -(np.percentile(portvalAMZNBOND,100.*p)-initportvAMZNBOND)
esP_amznbond_n = initportvAMZNBOND-
np.mean(portvalAMZNBOND[portvalAMZNBOND<=np.percentile(portvalAMZNBOND,100.*p)])

###################################
# Extreme period
indexset2 = range(T02)
portvalAMZNBOND2 = np.zeros(nboot)
for i in range(nboot):
    bindex = np.random.choice(indexset2,size=35)
    spfinal = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2
    amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
    bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
    bondoptval = np.maximum(bondstrike2-bondfinal,0.)
    amznoptval = np.maximum(amznstrike2-amznfinal,0.)
```

```
    portvalAMZNBOND2[i] = sharessp2*spfinal + sharesamzn2*amznfinal + sharesbond2*bondfinal+
VaR_amznbond_e = -(np.percentile(portvalAMZNBOND2,100.*p)-initportvAMZNBOND2)
esP_amznbond_e = initportvAMZNBOND2-
np.mean(portvalAMZNBOND2[portvalAMZNBOND2<=np.percentile(portvalAMZNBOND2,100.*p)])
```

```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 30 13:05:38 2020

@author: Group7
"""

def deltaput(price,strike,vol,rf,tmat):
    d1 = (np.log(price/strike)+(rf+vol*vol/2.)*tmat)/(vol*np.sqrt(tmat))
    nd1 = stats.norm.cdf(d1)
    delta = nd1-1         #put option should be nd1-1
    return delta

def gammaput(price,strike,vol,rf,tmat):
    d1 = (np.log(price/strike)+(rf+vol*vol/2.)*tmat)/(vol*np.sqrt(tmat))
    gamma = stats.norm.pdf(d1)/(price * vol * np.sqrt(tmat))       #put option should be nd1
1
    return gamma


#delta hedge SP500
#normal
inidelta=deltaput(spstartprice,spstrike,spyear,rf,T)
inidelta2=deltaput(spstartprice2,spstrike2,spyear,rf,T)


# estimate VaR

# calculate VaR
initoptionD=sharessp/-inidelta
initportvSPD = P0+sppval*initoptionD

####################################
# normal period
indexset = range(T0)
nboot = 10000
portvalSPD = np.zeros(nboot)
p = 0.01

for i in range(nboot):
    bindex = np.random.choice(indexset,size=35)
    spfinal = np.prod(np.exp(sprollingret[bindex]))*spstartprice.values
    splistD=spstartprice.values*np.exp(np.cumsum(sprollingret[bindex]))
    splistD=pd.DataFrame(np.insert(splistD,0,spstartprice))
    timelistD=np.arange(35,-1,-1)/250
    deltalistD=deltaput(pd.DataFrame(splistD).values,pd.DataFrame(np.tile(spstrike,36)).valu
    spppositionD=pd.DataFrame(initoptionD.values*-deltalistD)
    spposlagD=spppositionD.shift(1)
    pos=spppositionD-spposlagD
    pos=pos[1:]
    spcostD=np.sum(pos*splistD[1:])
    amznfinal = np.prod(np.exp(amznrollingret[bindex]))*amznstartprice
    bondfinal = np.prod(np.exp(bondrollingret[bindex]))*bondstartprice
    spoptvalD = np.maximum(spstrike-spfinal,0.)
    portvalSPD[i] = (sharessp+np.sum(pos).values)*spfinal-
spcostD.values + sharesamzn*amznfinal + sharesbond*bondfinal+initoptionD*spoptvalD

VaR_sp_nD = round(-(np.percentile(portvalSPD,100.*p)-initportvSPD),4)
esP_sp_nD = round(initportvSPD-
np.mean(portvalSPD[portvalSPD<=np.percentile(portvalSPD,100.*p)]),4)



####################################
# Extreme period
initoptionD2=sharessp2/-inidelta2
initportvSPD2 = P0+sppval2*initoptionD2
indexset2 = range(T02)
```

```
portvalSPD2 = np.zeros(nboot)

for i in range(nboot):
    bindex = np.random.choice(indexset,size=35)
    spfinal2 = np.prod(np.exp(sprollingret2[bindex]))*spstartprice2.values
    splistD2=spstartprice2.values*np.exp(np.cumsum(sprollingret2[bindex]))
    splistD2=pd.DataFrame(np.insert(splistD2,0,spstartprice2))
    timelistD2=np.arange(35,-1,-1)/250
    deltalistD2=deltaput(pd.DataFrame(splistD2).values,pd.DataFrame(np.tile(spstrike2,36)).v
    spppositionD2=pd.DataFrame(initoptionD2.values*-deltalistD2)
    spposlagD2=spppositionD2.shift(1)
    pos2=spppositionD2-spposlagD2
    pos2=pos2[1:]
    spcostD2=np.sum(pos2*splistD2[1:])
    amznfinal = np.prod(np.exp(amznrollingret2[bindex]))*amznstartprice2
    bondfinal = np.prod(np.exp(bondrollingret2[bindex]))*bondstartprice2
    spoptvalD2 = np.maximum(spstrike2-spfinal,0.)
    portvalSPD2[i] = (sharessp+np.sum(pos2).values)*spfinal-
spcostD2.values + sharesamzn2*amznfinal + sharesbond2*bondfinal+initoptionD2*spoptvalD2

VaR_sp_nD2 = round(-(np.percentile(portvalSPD2,100.*p)-initportvSPD2),4)
esP_sp_nD2 = round(initportvSPD2-
np.mean(portvalSPD2[portvalSPD2<=np.percentile(portvalSPD2,100.*p)]),4)
```