VaR

Yanis HALABI

April 2023

Value at Risk (VaR) is a commonly used risk metric in finance and investment management. It is widely used because it provides a simple and easy-to-understand measure of the potential losses that an investment portfolio or trading position could sustain over a given time horizon. VaR expresses the potential loss in monetary terms, and it provides a clear and concise summary of the risk level of a portfolio or investment strategy.

In this paper, I will discuss the three methods that can be used to calculate this risk metric and use Python to illustrate them. Then, using a random portfolio of equities, we will look into the difference between Marginal VaR (mVaR) and Incremental VaR (iVaR).

Definition of the VaR

The VaR is a statistical measure that quantifies the extent of possible financial losses within a firm, portfolio, or position over a specific timeframe (Investopedia). It is commonly used by investment and commercial banks to determine the extent and probabilities of potential losses.

This metric can be applied to specific positions or to whole portfolios, and it has three components: the amount of potential losses (in monetary amounts or percentage points of the portfolio), the probability of occurrence of these losses, and the timeframe.

Combined with other risk metrics, the VaR allows risk managers to gauge whether they are able to face and cover these potential losses or whether they need to reduce the size of a position in a portfolio, for example.

Methods of Calculation

Historical VaR

This method uses past returns to calculate the Value at Risk. It is simple to calculate but has some drawbacks as it relies on past results, which are not indicative of the future. It also depends on the horizon window chosen, as you might exclude data points that capture periods of crisis.

In the code below, we calculate the historical daily VaR of an equally weighted portfolio composed of 9 stocks and BTC (the data starts on September 2014).

Import Libraries

In [1]:
```
import pandas as pd ; import numpy as np ; from scipy.stats import norm
import datetime as dt ; import yfinance as yf ; import matplotlib.pyplot as plt
from tabulate import tabulate
```

Define Equity Portfolio

In [2]:
```
tickers = ['SOP.PA','MC.PA','RMS.PA','AAPL', 'MSFT','NVDA','BTC-USD',
           'NFLX', 'BMW.DE', 'GLE.PA']

weights = np.full(10, 1/10)
```

Get price Data

n [3]:
```
start_date = dt.date(year = 2013 , month = 1 ,day = 1)
end_date = dt.date.today()

price_data = pd.DataFrame(yf.download(tickers, start = start_date,
                                      end = end_date)['Close']).dropna().round(2)

print(price_data.head(3))
```

```
[*********************100%***********************]  10 of 10 completed
             AAPL  BMW.DE  BTC-USD  GLE.PA   MC.PA   MSFT   NFLX  NVDA  \
Date
2014-09-17  25.40   89.80   457.33   40.97  134.65  46.52  64.93  4.79
2014-09-18  25.45   90.38   424.44   41.54  135.65  46.68  65.57  4.86
2014-09-19  25.24   88.99   394.80   41.39  134.45  47.52  65.36  4.77

            RMS.PA  SOP.PA
Date
2014-09-17  243.70   74.81
2014-09-18  244.55   75.24
2014-09-19  245.60   75.00
```

Calculate returns

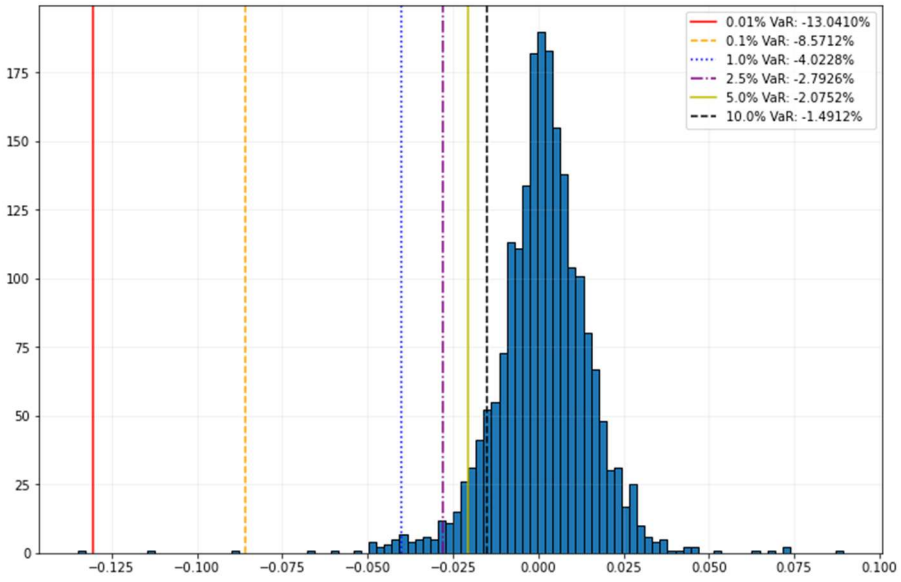
In [4]:
```
returns = price_data.pct_change().dropna().round(4)

print(returns.head(3))
```

```
              AAPL  BMW.DE  BTC-USD  GLE.PA   MC.PA    MSFT    NFLX    NVDA  \
Date
2014-09-18  0.0020  0.0065  -0.0719  0.0139  0.0074  0.0034  0.0099  0.0146
2014-09-19 -0.0083 -0.0154  -0.0698 -0.0036 -0.0088  0.0180 -0.0032 -0.0185
2014-09-22  0.0008 -0.0249   0.0186 -0.0130 -0.0149 -0.0097 -0.0323 -0.0105

            RMS.PA  SOP.PA
Date
2014-09-18  0.0035  0.0057
2014-09-19  0.0043 -0.0032
2014-09-22  0.0035  0.0020
```

Historical VaR

In [5]:
```
#calculate portfolio returns
portfolio_returns = returns.dot(weights)
portfolio_returns = portfolio_returns.sort_values(ascending = True)

#define the confidence levels
confidence_levels=[0.01,0.1,1.0,2.5,5.0,10.0]

#empty list to append the different VaR values
rows = []

#loop through the different confidence levels and append the results to the list rows
for i in range(len(confidence_levels)):

    VaR = np.percentile(portfolio_returns, confidence_levels[i])
    rows.append([str(confidence_levels[i])+'%', '{:.3f}%'.format(VaR*100)])

print(tabulate(rows, headers=['Confidence Level', 'Value at Risk'],
               tablefmt='fancy_grid', stralign='center', numalign='center'))

#plot VaR function
def plot_VaR(portfolio_returns, confidence_levels):
    plt.rcParams["figure.figsize"] = (12, 8)
    plt.hist(portfolio_returns, bins=100, edgecolor='black')

    colors = ['r','orange','b', 'purple','y', 'black']
    linestyle_str = ['-', '--',':','-.','-','--']

    for i in range(len(confidence_levels)):
        VaR = np.percentile(portfolio_returns, confidence_levels[i])
        label = '{}% VaR: {:.4f}%'.format(confidence_levels[i], VaR*100)
        plt.axvline(x=VaR, color=colors[i], linestyle=linestyle_str[i], label=label)

    plt.legend()
    xticks = np.arange(-0.125, 0.125, 0.025)
    plt.grid(True,alpha=0.2)
    plt.xticks(xticks)
    plt.show()

plot_VaR(portfolio_returns,confidence_levels)
```

Confidence Level	Value at Risk
0.01%	-13.041%
0.1%	-8.571%
1.0%	-4.023%
2.5%	-2.793%
5.0%	-2.075%
10.0%	-1.491%



Based on **historical data**, we can ensure with 99.9% confidence that losses won't exceed 8.571% in a single day.

Variance-Covariance, Parametric VaR

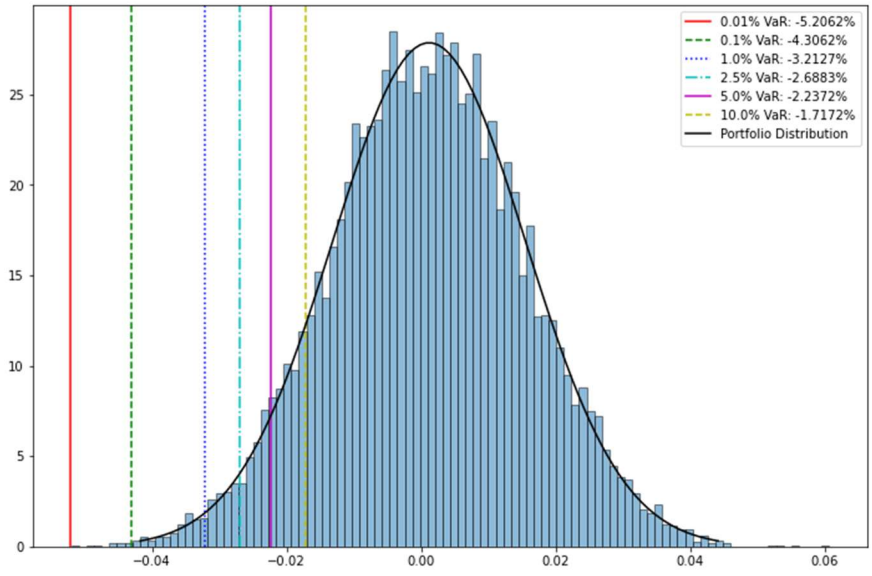
Rather than assuming that the past will inform the future, the parametric VaR is built under the assumption that the returns are normally distributed.

The downside of this method is the assumption of normality, which may not be accurate, as we will see in the following code.

Parametric VaR

```
# Define the confidence levels
confidence_levels = [0.0001, 0.001, 0.01,0.025, 0.05, 0.1]

# Calculate the different VaRs using the percent point function of a normal distribution
VaRs = [norm.ppf(p, portfolio_mean, portfolio_std) for p in confidence_levels]

# Print the VaRs
rows2 = [[str(p*100)+'%', '{:.3f}%'.format(v*100)] for p, v in zip(confidence_levels, Va
print(tabulate(rows2, headers=['Confidence Level', 'Value at Risk'], tablefmt='fancy_gri
               stralign='center', numalign='center'))

# Plot the histogram of the returns
returns = np.random.normal(portfolio_mean, portfolio_std, 10000)
plt.hist(returns, bins=100, density=True, alpha=0.5, edgecolor='black')

# Generate data to plot the normal distributions
x = np.linspace(portfolio_mean - 3*portfolio_std, portfolio_mean + 3*portfolio_std, 100)
ys = [norm.pdf(x, portfolio_mean, portfolio_std),] + [norm.pdf(x, portfolio_mean, v) for

# Plot the normal distributions
colors = ['r', 'g', 'b', 'c', 'm', 'y']
linestyle_str = ['-', '--',':','-.','-','--']

for i in range(len(confidence_levels)):
    VaR = norm.ppf(confidence_levels[i], portfolio_mean, portfolio_std)
    label = '{}% VaR: {:.4f}%'.format(confidence_levels[i]*100, VaR*100)
    plt.axvline(x=VaR, color=colors[i], linestyle=linestyle_str[i], label=label)  # Add

# Add the normal distribution curve for the portfolio
plt.plot(x, ys[0], 'k', label='Portfolio Distribution')

plt.legend()
plt.show()
```

Confidence Level	Value at Risk
0.01%	-5.206%
0.1%	-4.307%
1.0%	-3.213%
2.5%	-2.689%
5.0%	-2.238%
10.0%	-1.718%

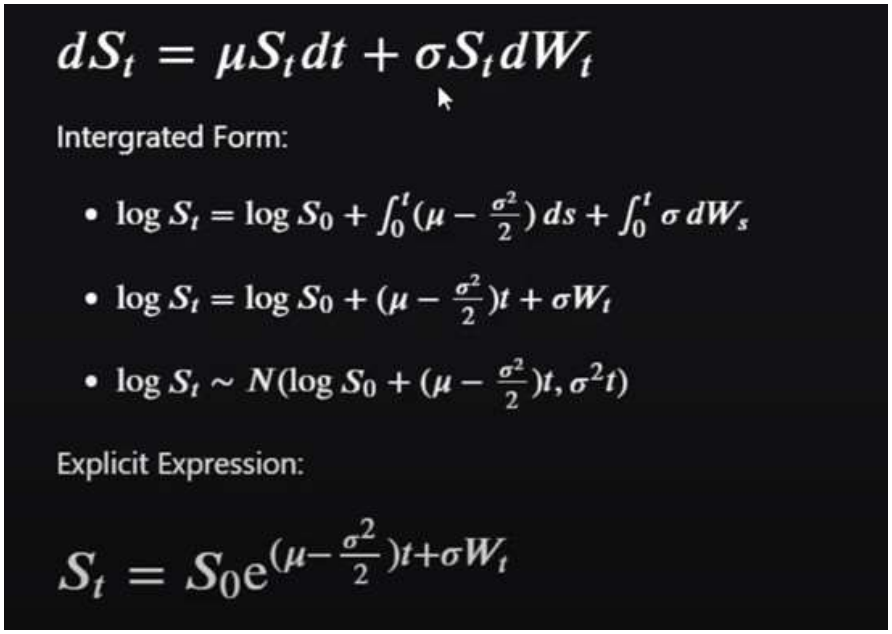


As you can see, with the parametric method and its assumption of normality, we can ensure at 99.9% that losses will not exceed 4.307% in a single day, which is not representative of the real distribution of the portfolio return. In this case, the historical VaR gives us a better picture of the potential losses that could occur on our portfolio, but again, past performances are not indicative of future performances.

Monte Carlo using Geometric Brownian Motion to simulate portfolio returns.

When using Geometric Brownian Motion to simulate our portfolio returns, we find much higher VaR values. They give us a more realistic picture of the potential losses that could occur.

Geometric Brownian Motion Formula



Define the variables

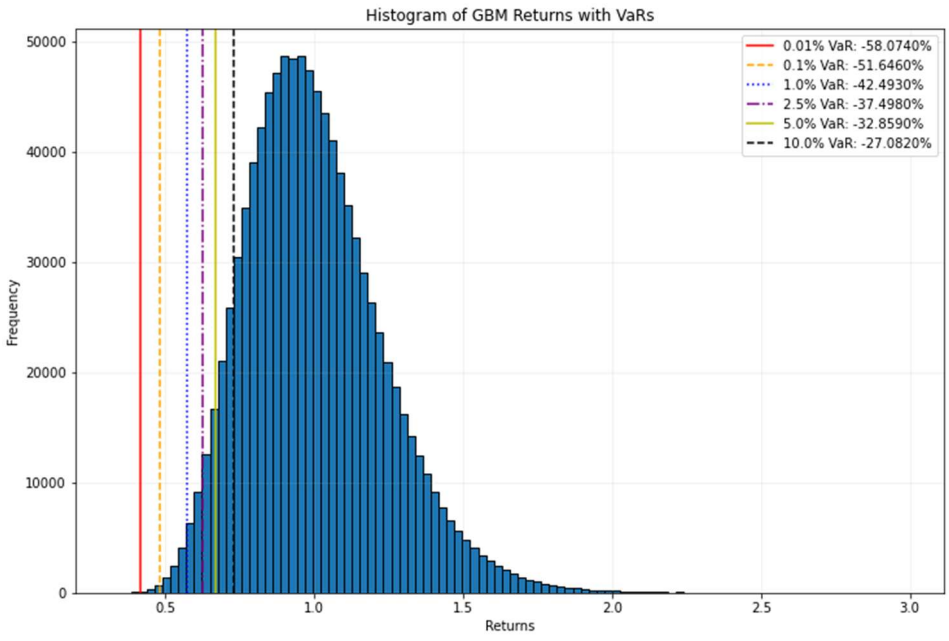
```
#Define the variables of the Geometric Brownian motion

T = 252 #corresponds to 1 year
M = 1000000 #number of simulations
n = 252 #number of steps - 252 trading days
dt = T/n #time step

mu = portfolio_mean
print(mu)

sigma = np.sqrt(252)*portfolio_std # Portfolio Volatility, 252-day sigma
print(sigma)
```

```
0.0011725
0.2272
```

GBM returns

```
# Calculate the GBM returns
GBM_returns = np.exp(((mu - sigma ** 2 / 2 ) * dt) + sigma *
                     np.random.normal(0 , np.sqrt(dt) , size=(M,n)).T)

confidence_levels = [0.01, 0.1, 1.0, 2.50, 5.0, 10.0]

var = scs.scoreatpercentile(GBM_returns-1, confidence_levels)

VaRs = [[str(cl)+'%', '{:.3f}%'.format(var*100)]
        for cl, var in zip(confidence_levels, var)]

print(tabulate(VaRs, headers=['Confidence Level', 'Value at Risk'], tablefmt='fancy_grid
                    stralign='center', numalign='center'))

# Plot the results

plt.rcParams["figure.figsize"] = (12, 8)
plt.hist(GBM_returns[1], bins= 100,edgecolor='black')

colors = ['r','orange','b', 'purple','y', 'black']
linestyle_str = ['-', '--',':','-.','-','--']


for i in range(len(confidence_levels)):
    cl = confidence_levels[i]
    var = VaRs[i][1].strip('%')
    var = float(var)
    var = (var/100) +1

    label = '{}% VaR: {:.4f}%'.format(cl, (var-1)*100)
    plt.axvline(x=float(var), color=colors[i], linestyle=linestyle_str[i], label=label)


plt.legend()
plt.grid(True, alpha=0.2)

plt.xlabel('Returns')
plt.ylabel('Frequency')
plt.title('Histogram of GBM Returns with VaRs')

plt.show()
```

Confidence Level	Value at Risk
0.01%	-58.074%
0.1%	-51.646%
1.0%	-42.493%
2.5%	-37.498%
5.0%	-32.859%
10.0%	-27.082%



Marginal VaR (mVaR)

Marginal VaR corresponds to the amount of additional risk that is added by a new investment in the portfolio. (Sub-portfolio = NFLX + BMW + SOGE)

{mVaR] = [VaR of the existing portfolio] – [VaR of the portfolio without the sub-portfolio]

A positive number indicates that the position is adding risk to the portfolio, while a negative one indicates that the position is reducing risk.

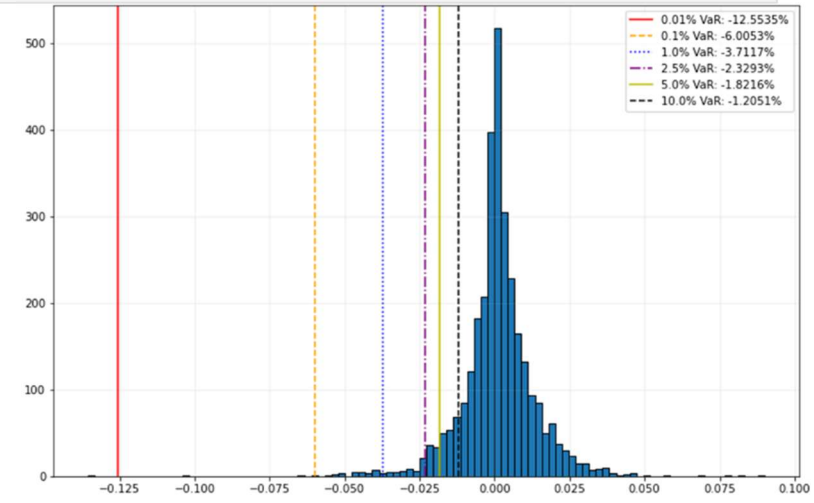
Note that an individual investment may have a high VaR individually, but if it's negatively correlated to the portfolio, it may contribute a much lower amount of VaR to the portfolio than its individual VaR.

What matters here is the asset's contribution to the portfolio risk, not the individual VaR which measures the risk in isolation.

Marginal Historical VaR : mVaR

```
# suppose that we have a sub-portfolio without : Netflix($NFLX), BMW($BMW.DE) and Société
sub_portfolio_tickers = ['SOP.PA','MC.PA','RMS.PA','AAPL', 'MSFT','NVDA','BTC-USD']
sub_portfolio_weights = np.full(7, 1/7)

sub_portfolio_price_data = pd.DataFrame(yf.download(sub_portfolio_tickers, start= start_
                                                    end = end_date)['Close']).round(2)
sub_portfolio_stocks_returns = sub_portfolio_price_data.pct_change().dropna().round(4)

#sub_portfolio returns

sub_portfolio_returns = sub_portfolio_stocks_returns.dot(sub_portfolio_weights)

confidence_levels=[0.01,0.1,1.0,2.5,5.0,10.0]
rowy = []
#loop through the different confidence levels and append the results to the list rowy
for i in range(len(confidence_levels)):

    sub_portfolio_VaR = np.percentile(sub_portfolio_returns, confidence_levels[i])
    VaR = np.percentile(portfolio_returns, confidence_levels[i])
    rowy.append([str(confidence_levels[i])+'%', '{:.3f}%'.format((VaR -
                                                        sub_portfolio_VaR)*100

print(tabulate(rowy, headers=['Confidence Level', 'Marginal VaR'],
               tablefmt='fancy_grid', stralign='center', numalign='center'))

plot_VaR(sub_portfolio_returns,confidence_levels)
```

Confidence Level	Marginal VaR
0.01%	-0.487%
0.1%	-2.560%
1.0%	-0.309%
2.5%	-0.461%
5.0%	-0.253%
10.0%	-0.286%



In absolute values at 99.9%: mVaR = 2.560% = 8.5712% - 6.0053%

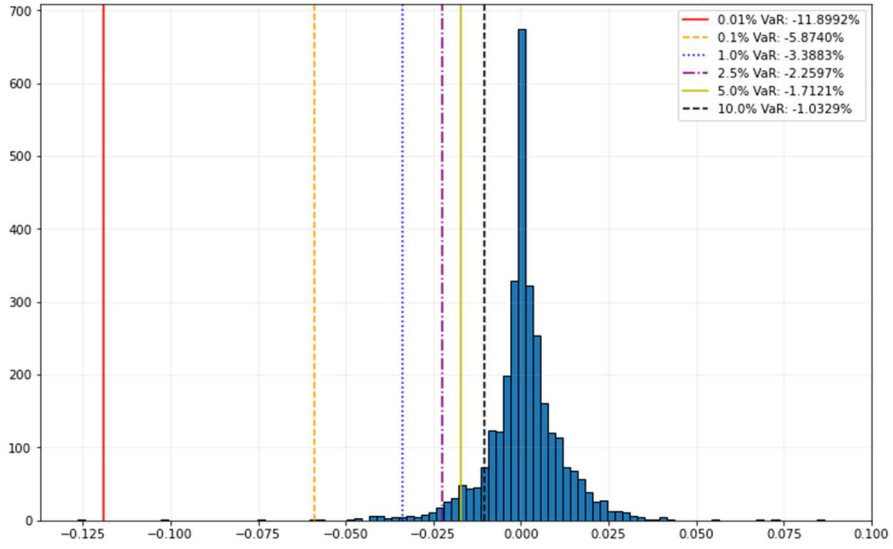
A positive number indicates that the position is adding risk to the portfolio.

Incremental VaR (iVaR)

Incremental Historical VaR

```
#We will now increase the size of our BMW holdings to 20% to see the impact on the VaR a

tickers = ['SOP.PA','MC.PA','RMS.PA','AAPL', 'MSFT','NVDA','BTC-USD',
           'NFLX', 'BMW.DE', 'GLE.PA']
new_portfolio_weights = np.full(10,0.888/10)
new_portfolio_weights[-2] = 0.2



new_portfolio_price_data = pd.DataFrame(yf.download(tickers, start= start_date, end = en
new_portfolio_stocks_returns = new_portfolio_price_data.pct_change().dropna().round(4)


#new_portfolio return

new_portfolio_returns = new_portfolio_stocks_returns.dot(new_portfolio_weights)

confidence_levels=[0.01,0.1,1.0,2.5,5.0,10.0]

rowy2 = []

for i in range(len(confidence_levels)):

    new_portfolio_VaR = np.percentile(new_portfolio_returns, confidence_levels[i])
    VaR = np.percentile(portfolio_returns, confidence_levels[i])
    rowy2.append([str(confidence_levels[i])+'%', '{:.3f}%'.format(( VaR - new_portfolio

print(tabulate(rowy2, headers=['Confidence Level', 'Incremental VaR'],
               tablefmt='fancy_grid', stralign='center', numalign='center'))


plot_VaR(new_portfolio_returns,confidence_levels)
```

Confidence Level	Incremental VaR
0.01%	-1.141%
0.1%	-2.691%
1.0%	-0.633%
2.5%	-0.530%
5.0%	-0.362%
10.0%	-0.458%



$$d(VaR) = \frac{dw_i}{w_i} iVaR_i$$

The equation above can be interpreted as follows, if we change by (*dwi/wi) %* the position of a stock in our portfolio, the VaR of the portfolio will change by the iVaR of that position, multiplied by (*dwi/wi) %.*

In our case, at a 5% level of confidence, dVar = -0.362% = 10% * iVaR ; iVaR = - 3.62%

Here the iVaR of BMW.DE is negative, meaning that if we increase the size of the position in the portfolio, it will decrease the VaR of the portfolio (based on historical values).

Whereas mVaR tells us how much removing the entire position would change the overall VaR of the portfolio, incremental VaR indicates the impact of a small change in a position on the overall VaR of the portfolio.

Sources:

Northstar Risk. (n.d.). Northstar Risk: Research. [online] Available at: https://www.northstarrisk.com/research

Northstar Risk. (n.d.). Northstar Risk: 003.Historical VaR. [online] Available at: https://www.northstarrisk.com/003historical-var-white-paper.

Investopedia. (n.d.). Marginal VaR Definition. [online] Available at: https://www.investopedia.com/terms/m/marginal-var.asp.

Corporate Finance Institute. (n.d.). Value at Risk (VaR). [online] Available at: https://corporatefinanceinstitute.com/resources/risk-management/value-at-risk-var/.

Kenton, W. (2022). Value at Risk (VaR) Explained. [online] Investopedia. Available at: https://www.investopedia.com/terms/v/var.asp.