Computing Value at Risk (VaR)

By: Nickolas Black

January 13, 2023

- **Value at Risk (VaR) is a measure of the risk of loss for investments**
- **Parametric Var (variance-covariance method): First identifies the mean, or expected value, and standard deviation of a portfolio**
- **Conditional VaR (CVaR) is the exact expected lose if the condition is met.**
- **Confidence Interval: A range of values that we expect our true value to lie within with some confidence.**

Key Formulas

(Matrix Multiplication)

Portfolio Expected Portofio return = Portfolio Expected Return * Weights

Portfolio Variance = Weights(Transpose) *(Covariance Matrix)* Weights

Interpretation

- **Daily Value at Risk of 6.5% with a 95% confidence**
  - **There is a 5% confidence that your portfolio will lose 6.5% or more in a day.**
  - **or, we have a 95% confidence that our portfolio will not lose 6.5% in a day.**

In [1]:

```
!pip install yfinance
```

```
Collecting yfinance
  Downloading yfinance-0.2.3-py2.py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 2.0 MB/s eta 0:00:011
Requirement already satisfied: cryptography>=3.3.2 in ./opt/anaconda3/lib/python3.9/site-packages (from yfinance) (3.4.8)
Collecting frozendict>=2.3.4
  Downloading frozendict-2.3.4-cp39-cp39-macosx_10_9_x86_64.whl (33 kB)
Requirement already satisfied: appdirs>=1.4.4 in ./opt/anaconda3/lib/python3.9/site-packages (from yfinance) (1.4.4)
Requirement already satisfied: html5lib>=1.1 in ./opt/anaconda3/lib/python3.9/site-packages (from yfinance) (1.1)
Collecting pytz>=2022.5
  Downloading pytz-2022.7-py2.py3-none-any.whl (499 kB)
     |████████████████████████████████| 499 kB 5.6 MB/s eta 0:00:01
Collecting beautifulsoup4>=4.11.1
  Downloading beautifulsoup4-4.11.1-py3-none-any.whl (128 kB)
     |████████████████████████████████| 128 kB 60.8 MB/s eta 0:00:01
Requirement already satisfied: pandas>=1.3.0 in ./opt/anaconda3/lib/python3.9/site-packages (from yfinance) (1.3.4)
Collecting lxml>=4.9.1
  Downloading lxml-4.9.2-cp39-cp39-macosx_10_15_x86_64.whl (4.8 MB)
     |████████████████████████████████| 4.8 MB 40.9 MB/s eta 0:00:01
Requirement already satisfied: numpy>=1.16.5 in ./opt/anaconda3/lib/python3.9/site-packages (from yfinance) (1.20.3)
Requirement already satisfied: requests>=2.26 in ./opt/anaconda3/lib/python3.9/site-packages (from yfinance) (2.26.0)
Collecting multitasking>=0.0.7
  Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Requirement already satisfied: soupsieve>1.2 in ./opt/anaconda3/lib/python3.9/site-packages (from beautifulsoup4>=4.11.1->yfinance
) (2.2.1)
Requirement already satisfied: cffi>=1.12 in ./opt/anaconda3/lib/python3.9/site-packages (from cryptography>=3.3.2->yfinance) (1.1
4.6)
Requirement already satisfied: pycparser in ./opt/anaconda3/lib/python3.9/site-packages (from cffi>=1.12->cryptography>=3.3.2->yfi
nance) (2.20)
Requirement already satisfied: six>=1.9 in ./opt/anaconda3/lib/python3.9/site-packages (from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in ./opt/anaconda3/lib/python3.9/site-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.7.3 in ./opt/anaconda3/lib/python3.9/site-packages (from pandas>=1.3.0->yfinance
) (2.8.2)
Requirement already satisfied: charset-normalizer~=2.0.0 in ./opt/anaconda3/lib/python3.9/site-packages (from requests>=2.26->yfin
ance) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in ./opt/anaconda3/lib/python3.9/site-packages (from requests>=2.26->yfinance) (3.2)
Requirement already satisfied: certifi>=2017.4.17 in ./opt/anaconda3/lib/python3.9/site-packages (from requests>=2.26->yfinance) (
2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in ./opt/anaconda3/lib/python3.9/site-packages (from requests>=2.26->yfinance
) (1.26.14)
Installing collected packages: pytz, multitasking, lxml, frozendict, beautifulsoup4, yfinance
  Attempting uninstall: pytz
    Found existing installation: pytz 2021.3
    Uninstalling pytz-2021.3:
      Successfully uninstalled pytz-2021.3
  Attempting uninstall: lxml
    Found existing installation: lxml 4.6.3
    Uninstalling lxml-4.6.3:
      Successfully uninstalled lxml-4.6.3
  Attempting uninstall: beautifulsoup4
    Found existing installation: beautifulsoup4 4.10.0
    Uninstalling beautifulsoup4-4.10.0:
      Successfully uninstalled beautifulsoup4-4.10.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the so
urce of the following dependency conflicts.
conda-repo-cli 1.0.4 requires pathlib, which is not installed.
Successfully installed beautifulsoup4-4.11.1 frozendict-2.3.4 lxml-4.9.2 multitasking-0.0.11 pytz-2022.7 yfinance-0.2.3
```

In [4]:

```
import yfinance as yf
import numpy as np
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt
from scipy.stats import norm
```

In [9]:

```
# list the ticket prices for companies in the portfolio
tickers = ['AZTA', 'BX', 'TAN', 'LSCC', 'F', 'RTX', 'TSM', 'AMD', 'LRCX']

# List the weights for each company in the portfolio. Each company has 10% of the portfolio
weights = np.array([0.09, 0.09, 0.09, 0.18, 0.18, 0.09, 0.09, 0.09, 0.09])
```

In [10]:

```
# Enter the start date and end date for the data call
start = dt.datetime(2019,1,1) # Data from January 1, 2019
end = dt.datetime.now() # .now() specfies to the present moment

# Enter the ticker, start, and end time. Returns data frame of listed companies with just adjusted close
data = yf.download(tickers, start, end)['Adj Close']
data # display data
```

```
[*********************100%***********************]  9 of 9 completed
```

Out[10]:

Date	AMD	AZTA	BX	F	LRCX	LSCC	RTX	TAN	TSM
2019-01-02	18.830000	25.487028	25.613279	7.018951	130.142838	6.960000	61.813934	18.716314	32.730354
2019-01-03	17.049999	23.807423	24.897778	6.912335	123.866264	6.810000	59.062840	18.517097	30.794495
2019-01-04	19.000000	25.155014	25.749561	7.178876	129.917038	6.850000	61.083340	19.373726	31.341194
2019-01-07	20.570000	25.692097	26.516174	7.365456	131.074448	6.960000	61.054806	19.732313	31.574211
2019-01-08	20.750000	25.604214	26.746157	7.436534	128.853638	7.010000	62.156387	19.742273	31.314301
...	...	...	...	...	...	...	...	...	...
2023-01-06	63.959999	61.180000	79.220001	12.580000	445.269989	66.529999	102.459999	73.870003	78.070000
2023-01-09	67.239998	60.630001	80.580002	12.690000	452.429993	68.650002	99.599998	74.709999	80.309998
2023-01-10	68.050003	61.950001	80.529999	12.840000	458.609985	70.139999	99.919998	77.040001	81.269997
2023-01-11	69.059998	61.840000	83.070000	13.220000	464.299988	72.080002	99.589996	80.209999	81.779999
2023-01-12	70.800003	60.320000	85.059998	13.430000	470.040009	72.080002	100.680000	81.680000	87.000000

1016 rows × 9 columns

In [12]:

```
# Returns data frame of the percentage change of returns for all the companies
returns = data.pct_change()
returns
```

Out[12]:

Date	AMD	AZTA	BX	F	LRCX	LSCC	RTX	TAN	TSM
2019-01-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2019-01-03	-0.094530	-0.065900	-0.027935	-0.015190	-0.048228	-0.021552	-0.044506	-0.010644	-0.059146
2019-01-04	0.114370	0.056604	0.034211	0.038560	0.048849	0.005874	0.034209	0.046261	0.017753
2019-01-07	0.082632	0.021351	0.029772	0.025990	0.008909	0.016058	-0.000467	0.018509	0.007435
2019-01-08	0.008751	-0.003421	0.008673	0.009650	-0.016943	0.007184	0.018043	0.000505	-0.008232
...	...	...	...	...	...	...	...	...	...
2023-01-06	0.026151	0.017124	0.034339	0.026939	0.067640	0.030036	0.015662	0.041743	0.030899
2023-01-09	0.051282	-0.008990	0.017167	0.008744	0.016080	0.031865	-0.027913	0.011371	0.028692
2023-01-10	0.012046	0.021771	-0.000621	0.011820	0.013660	0.021704	0.003213	0.031187	0.011954
2023-01-11	0.014842	-0.001776	0.031541	0.029595	0.012407	0.027659	-0.003303	0.041147	0.006275
2023-01-12	0.025196	-0.024580	0.023956	0.015885	0.012363	0.000000	0.010945	0.018327	0.063830

1016 rows × 9 columns

Parametric VaR

First, we compute the risk of the portfolio or the portfolio's covariance.

The covariance matrix shows the covariance or risk of each company with another company. One particular company's variance is the intercept of the company on the Matrix.

In [14]:

```
# compute the covariance matrix
cov_matrix = returns.cov()
cov_matrix
```

Out[14]:

	AMD	AZTA	BX	F	LRCX	LSCC	RTX	TAN	TSM
AMD	0.001208	0.000650	0.000470	0.000361	0.000707	0.000689	0.000222	0.000518	0.000493
AZTA	0.000650	0.001169	0.000427	0.000393	0.000745	0.000721	0.000342	0.000524	0.000468
BX	0.000470	0.000427	0.000704	0.000398	0.000492	0.000481	0.000298	0.000392	0.000293

	AMD	AZTA	BX	F	LRCX	LSCC	RTX	TAN	TSM
F	0.000361	0.000393	0.000398	0.000800	0.000429	0.000398	0.000353	0.000347	0.000270
LRCX	0.000707	0.000745	0.000492	0.000429	0.001003	0.000741	0.000344	0.000521	0.000522
LSCC	0.000689	0.000721	0.000481	0.000398	0.000741	0.001197	0.000290	0.000553	0.000490
RTX	0.000222	0.000342	0.000298	0.000353	0.000344	0.000290	0.000487	0.000267	0.000186
TAN	0.000518	0.000524	0.000392	0.000347	0.000521	0.000553	0.000267	0.000815	0.000362
TSM	0.000493	0.000468	0.000293	0.000270	0.000522	0.000490	0.000186	0.000362	0.000538

Next we compute the expected returns, which is the average of the returns.

In [16]:

```
# compute expected returns
avg_returns = returns.mean()
avg_returns
```

Out[16]:

```
AMD     0.001904
AZTA    0.001429
BX      0.001536
F       0.001036
LRCX    0.001767
LSCC    0.002893
RTX     0.000724
TAN     0.001861
TSM     0.001232
dtype: float64
```

In [18]:

```
count = returns.count() # Confirm that there are the same number of observations in the data
count
```

Out[18]:

```
AMD     1015
AZTA    1015
BX      1015
F       1015
LRCX    1015
LSCC    1015
RTX     1015
TAN     1015
TSM     1015
dtype: int64
```

Construct a Normal Distribution curve

Compute the mean and standard deviation of the portforlio to set to the x scale of the normal distribution chart. The x axis will be a scale with evenly spaced numbers based on our observations. The y scale is the frequency of occurances. New instances will fall within the normal distribution.

To compute the scale. Use the weights of each stock in the portfolio and the other parameters.

In [19]:

```
port_mean = avg_returns @ weights
port_std = np.sqrt(weights.T @ cov_matrix @ weights)
```

In [20]:

```
port_mean # Expected daily return of the portfolio
```

Out[20]:

```
0.0015465729164030372
```

In [21]:

```
port_std # Portfolio Covariance
```

Out[21]:

```
0.02230284202172032
```

In [28]:

```
#np.arange returns evenly spaced values with an given interval
x = np.arange(-0.05,0.055,0.001)

# probabilty density function from scipy stats. Its parameters are the scale, mean, and std
norm_dist = norm.pdf(x, port_mean, port_std)

# display distribution.
norm_dist
```

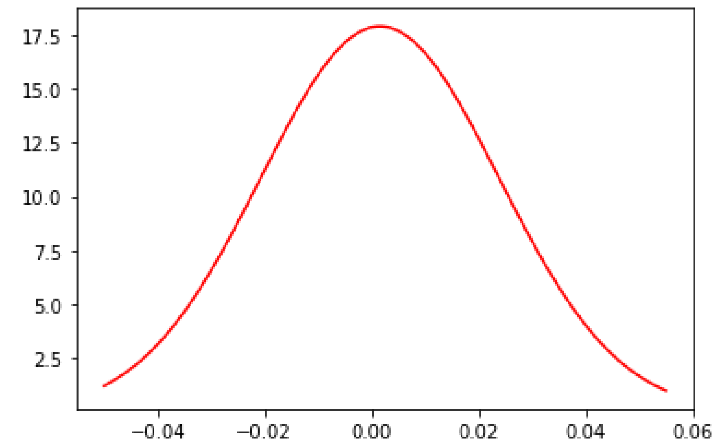
Out[28]:

```
array([ 1.2376997 ,  1.37146283,  1.51663018,  1.67379493,  1.84353633,
        2.02641343,  2.22295827,  2.43366874,  2.65900111,  2.89936221,
        3.15510145,  3.42650274,  3.71377629,  4.01705053,  4.3363642 ,
        4.67165869,  5.02277081,  5.38942606,  5.77123255,  6.16767571,
        6.57811388,  7.00177491,  7.43775386,  7.88501202,  8.34237709,
        8.80854495,  9.28208268,  9.76143329, 10.24492183, 10.73076311,
       11.21707094, 11.70186882, 12.18310211, 12.65865149, 13.12634767,
       13.58398716, 14.02934911, 14.46021274, 14.87437555, 15.26967181,
       15.64399122, 15.99529758, 16.32164716, 16.62120654, 16.89226973,
       17.13327434, 17.34281657, 17.5196648 , 17.66277166, 17.77128439,
       17.84455324, 17.88213799, 17.88381237, 17.84956628, 17.77960593,
       17.67435176, 17.53443425, 17.36068758, 17.15414139, 16.91601064,
```

```
        16.64768363, 16.35070859, 16.02677881, 15.67771654, 15.30545599,
        14.91202552, 14.49952929, 14.07012868, 13.62602355, 13.16943373,
        12.70258083, 12.22767068, 11.74687643, 11.2623227 , 10.77607074,
        10.29010487,  9.80632016,  9.32651162,  8.85236482,  8.38544802,
         7.92720591,  7.47895478,  7.04187929,  6.61703062,  6.20532616,
         5.80755033,  5.42435684,  5.05627191,  4.70369865,  4.36692226,
         4.046116  ,  3.74134795,  3.45258811,  3.17971606,  2.92252888,
         2.68074918,  2.45403331,  2.24197944,  2.04413557,  1.86000738,
         1.68906569,  1.53075368,  1.3844937 ,  1.24969363,  1.12575278,
         1.01206732])
```

In [29]:

```
# Plot the normal distribution
plt.plot(x, norm_dist, color='r')
plt.show()
```



In [40]:

```
confidence_level = 0.05 #set the confidence level

# PPF represents the percent point function
VaR = norm.ppf(confidence_level, port_mean, port_std)

#formated print of VaR
print("I have 95% confidence that my portfolio will not lose more than {:.2f}% in one day or there is 5% confidence that it will l
ose more than {:.2f}% in one day".format(VaR*100, VaR*100))
```

```
I have 95% confidence that my portfolio will not lose more than -3.51% in one day or there is 5% confidence that it will lose more
than -3.51% in one day
```

In [43]:

```
# VaR over 5 days
num_days = 5

# compute 5 day Var
five_day_VaR = VaR * np.sqrt(num_days) # Var time square root of number of days

# Formatted print
print(" There is 95% confidence my protfolio will not lose more that {:.2f}% in the next {} days".format(five_day_VaR*100, num_day
s))
```

```
 There is 95% confidence my protfolio will not lose more that -7.86% in the next 5 days
```

Confidence Interval

The value of z at 95.5% confidence inderval is equal to 2. Sigma represents the standard diviation and n represents the count.

In [47]:

```
# Compute the lower interval
lower = port_mean - 2 * port_std / np.sqrt(count)

#compute the higher interval
higher = port_mean + 2 * port_std / np.sqrt(count)
```

In [48]:

```
lower
```

Out[48]:

```
AMD     0.000146
AZTA    0.000146
BX      0.000146
F       0.000146
LRCX    0.000146
LSCC    0.000146
RTX     0.000146
TAN     0.000146
TSM     0.000146
dtype: float64
```

In [49]:

```
higher
```

Out[49]:

```
AMD     0.002947
AZTA    0.002947
BX      0.002947
F       0.002947
LRCX    0.002947
LSCC    0.002947
```

```
ESCC     0.002947
RTX      0.002947
TAN      0.002947
TSM      0.002947
dtype: float64
```

With 95.5% degree of confidence we can expect daily returns between the lower and higher confidence interval.

In [ ]:

```

```