QFO_Blog

October 25, 2021

1 From Quantatitive to Quantum Finance

If you are a follower of Quantum Computing news you may have noticed a strange new player in the world of Quantum Research: the world's largest financial institutions. J.P. Morgan, Wells Fargo, Barclays, Citigroup, Goldman Sachs, and others are all currently pursuing quantum capabilities. According to the Boston Consulting Group, Quantum Computing could add "up to almost $70 billion in additional operating income for banks and other financial-services companies as the technology matures over the next several decades." But what exactly do these banks want with a Quantum Computer - even as the technology is just barely emerging from academic labs? To answer these questions, we are first going to begin with a pivotal moment in wealth management, the advent of Quantitative Finance.

Although stock and options trading began as early as the 17th century, the story of Quantitative Finance waits until the 1900s to emerge. Mathematician Louis de Bachelier (1900) is credited with being the first to introduce the idea of using Brownian Motion to model asset pricing, an idea that would go on to form the foundations of modern Quantative Finance, albeit 70 years after its largely ignored invention. Despite the burial of Bachelier's work, other Mathematicians, Statisticians, Physicists, and Economists continued to revolutionize the field of finance. Our next stop is in 1953: Economist and Nobel-laureate Harry Markowitz's development of Modern Portfolio Theory (MPT). Markowitz was unique in his approach of using statistical measures in order to maximize portfolio diversification. MPT was so revolutionary in the field of finance that renowned economist Milton Friedman notoriously argued that MPT was not economic theory at all. Nevertheless, Markowitz Portfolio Optimization represented a new era for finance.

Even better, MPO is not terribly complicated and can be programmed in just a few lines of classical code:

```
A key component of the MPT theory is diversification. Most investments are either high risk and

Using MPT we can either set an acceptable risk level snf optimize portfolio returns, or set our
```

1.1 Markowitz Model

1.1.1 Efficient Frontier Portfolio Optimisation in Python

Next we will test the Datareader by importing data Netflix's stock data.

[2]:

```
                 High         Low        Open       Close    Volume  \
Date
2016-01-04  110.000000  105.209999  109.000000  109.959999  20794800
2016-01-05  110.580002  105.849998  110.449997  107.660004  17664600
2016-01-06  117.910004  104.959999  105.290001  117.680000  33045700
2016-01-07  122.180000  112.290001  116.360001  114.559998  33636700
2016-01-08  117.720001  111.099998  116.330002  111.389999  18067100

             Adj Close
Date
2016-01-04  109.959999
2016-01-05  107.660004
2016-01-06  117.680000
2016-01-07  114.559998
2016-01-08  111.389999
```

Now we want the actual stock portfolio that we are going to optimize, since let's face it, we need more than one investment to optimize! We are going to use the same procedure as above to load in the stock data for Apple, Amazon, Google, and Facebook because if you are going to have a fake portfolio you should make it a rich one.

You can once again explore the portfolio dataframe below.

Finally, we plot both the change in stock price over time as well as the daily change or 'Volatility' of each stock price. *Notice how the prices of the stocks fluctate over time in unpredicatble ways* - this is the essence of the financial optimization problem: using pseudo-random variables to your advantage.

CHALLENGE: How would you attempt to make sense of 4 data series that seem like a complete roll of the dice?

The answer in the case of Markowitz Portfolio Optimization is to compare how each data series changes TOGETHER. For those with a background in statistics, we use the Covariance of our portfolio matrix

---

1.1.2 Acceptable Risk

MPT assumes that investors are risk-averse, meaning they prefer a less risky portfolio to a riskier one for a given level of return; risk aversion implies that most people should invest in multiple asset classes.

The expected return of the portfolio is calculated as a weighted sum of the returns of the individual assets. If a portfolio contained four equally weighted assets with expected returns of 2%, 7%, 10%, and 12%, the portfolio's expected return would be:

(2% x 25%) + (7% x 25%) + (10% x 25%) + (12% x 25%) = 7.75%

This is NOT the case for the risk of a portfolio.

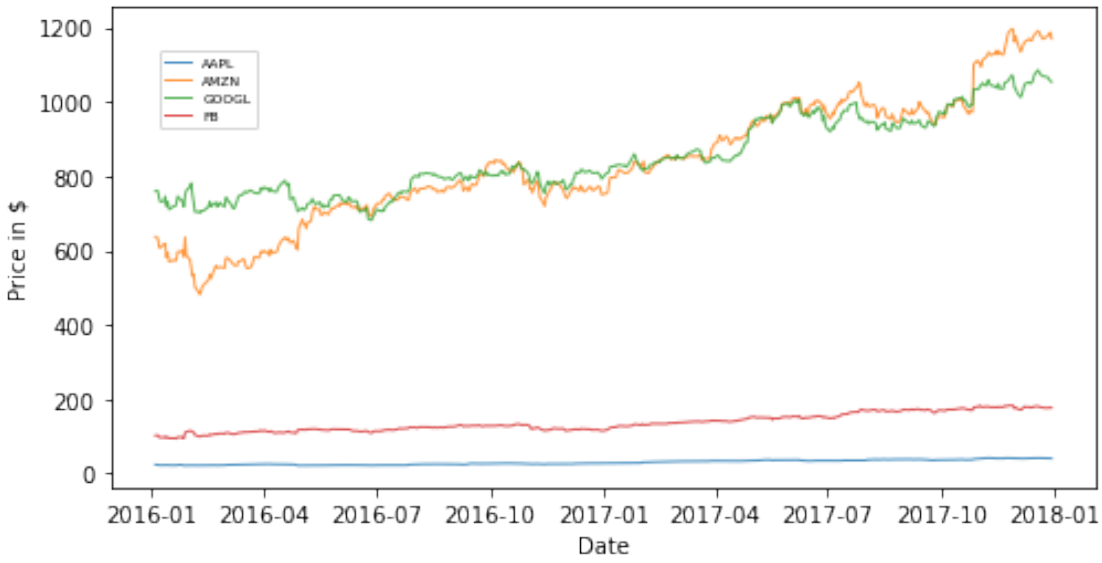
The portfolio's risk is a function of the **variances** of each asset and the **correlations** of each pair of assets. To calculate the risk of a four-asset portfolio, we need each of the four assets' variances and six correlation values, since there are six possible two-asset combinations with four assets. Because of the asset correlations, the total portfolio risk, or standard deviation, is lower than what would be calculated by a weighted sum.

(While we don't have time to cover the intricacies of the statistics involved if you are interested I encourage you to read this article from Stanford)

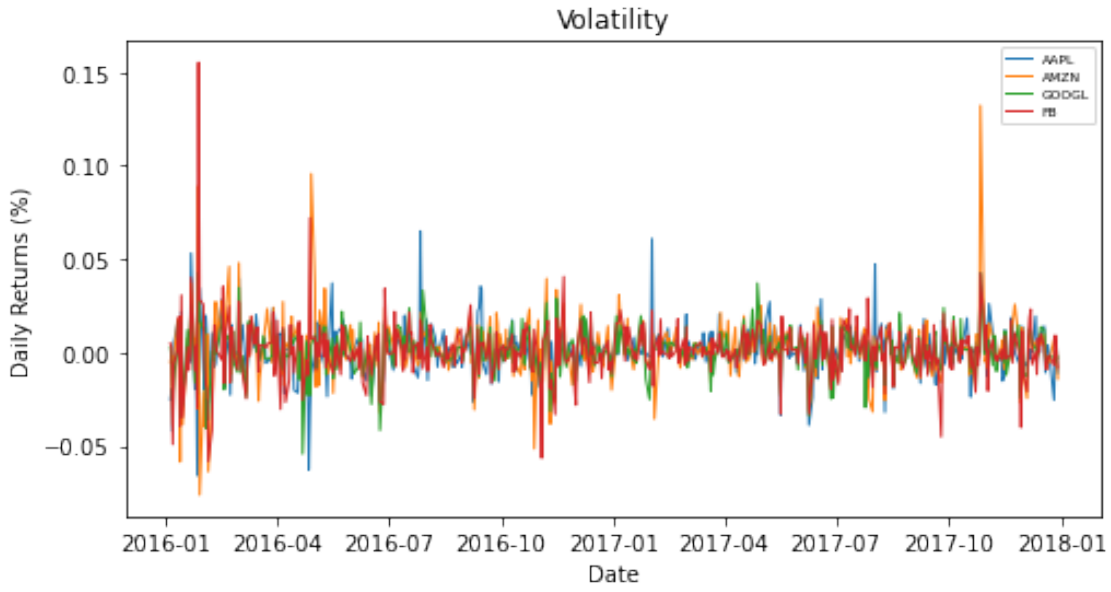
[4]:

Date	AAPL	AMZN	GOOGL	FB
2016-01-04	24.286827	636.989990	759.440002	102.220001
2016-01-05	23.678219	633.789978	761.530029	102.730003
2016-01-06	23.214838	632.650024	759.330017	102.970001
2016-01-07	22.235077	607.940002	741.000000	97.919998
2016-01-08	22.352644	607.049988	730.909973	97.330002

[6]:
```
Text(0.5, 0, 'Date')
```



[7]:
```
Text(0.5, 0, 'Date')
```



As discussed above, if we want to pin down the volatility above, we are going to need to analyze not just how each stock price changes, but how they all change together. You can think of this as reducing the degrees of freedom of our data. In order to convert from daily returns and covariance (which comes easily from the data above) we multiple by 250 days since there are approximately 250 open trading days a year.

Let's use this information to find all POSSIBLE stock portfolios combinations that contain our four chosen stocks. We plot these data points as 'Expected Return' vs 'Expected Volatility (Standard Deviation)' below.

We will do this by randomly generating 50000 portfolio weight/stock combinations using our 4 selected prices. Following the minutia of the code below is unimportant (most of it is staistics and data manipulation) what is important are the takeways from this exercise.

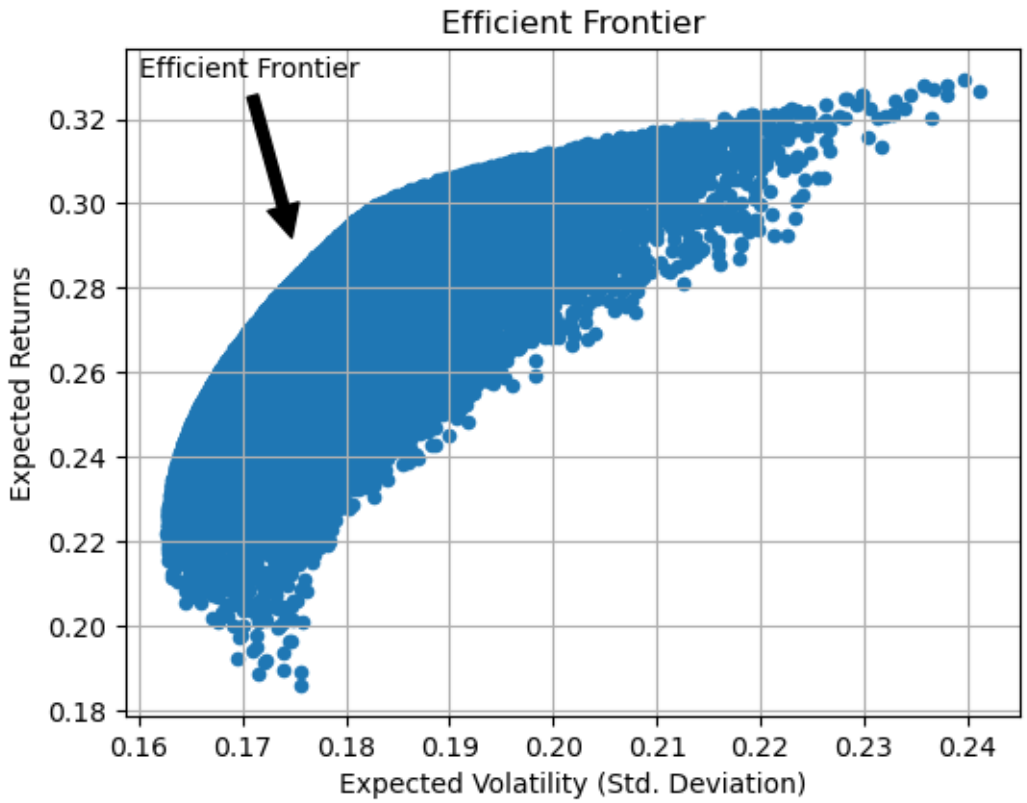
[9]:

	Returns	Volatility	AAPL weight	AMZN weight	GOOGL weight	FB weight
0	0.292096	0.202286	0.023875	0.514125	0.202750	0.259250
1	0.264452	0.176105	0.666280	0.073598	0.182612	0.077510
2	0.267344	0.174573	0.341212	0.336459	0.297326	0.025003
3	0.277317	0.176002	0.498807	0.288638	0.170514	0.042041
4	0.271846	0.173572	0.565873	0.081712	0.141994	0.210420

Let's take a look at the graph: Remember, we want to *maximize* Expected Returns while *minimizing* Risk, in this case volatiliity. There is a clearly a subset of all possible portfolios that represent the upper limit of portfolio performance - we call this boundary the **Efficient Frontier**.

```
The Ffficient Frontier is the line that indicates the efficient set of portfolios that will prov

When a portfolio falls to the right of the efficient frontier, it possesses greater risk relativ
```



Now that we understand the Efficient Frontier, it is finally time to find our maximized portfolios according to MPT. We have done all of the hard-work above by calculating the Covariance Matrix, Expected Returns, and Volatility above, therefore, we simply run our calculations, save the results, and use Panda's latent ability to locate extrema within a datafrane.

The next step then is to define the parameter Panda's must optimize. It turns out that this metric is the **Sharpe Ratio** which describes the excess return you are receiving for the extra volatility you accept by holding a riskier asset.

Sharpe Ratio$ = $R_p - R_{f\overline{\sigma_p\$}}$ $R_p = $ Expected Portfolio Return, $R_f = $ Risk Free Rate, $\sigma_p = $ $Portfolio Standard Deviation$

The code below translates our calculations above into functions to carry out on all possible portfolio combinations and subsequently finds the portfolios that 1. Maximize the Sharpe Ratio 2. Minimize Volatitlity

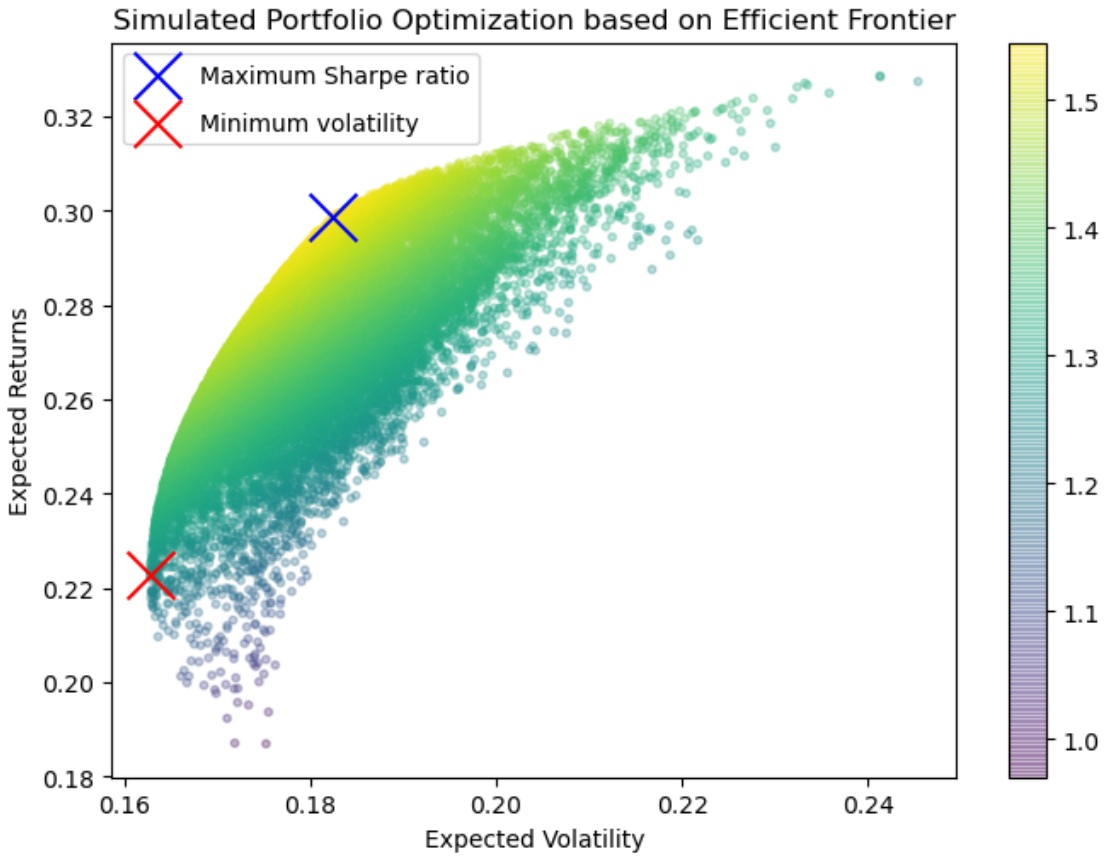
```
--------------------------------------------------------------------------------
Maximum Sharpe Ratio Portfolio Allocation

Annualised Return: 0.3
Annualised Volatility: 0.18
```

```
              AAPL    AMZN  GOOGL     FB
allocation  44.03   28.93   0.19  26.85
--------------------------------------------------------------------------------
Minimum Volatility Portfolio Allocation

Annualised Return: 0.22
Annualised Volatility: 0.16


              AAPL  AMZN  GOOGL    FB
allocation  34.93  0.34  56.95  7.78
```



Since its introduction, MPT has been surpassed by many theories and models that correct some its flawed assumptions, account for more variables, and posses greater mathematical sophistication. However, it is difficult to argue against Markowitz's work as a landmark moment in Quantitative Finance. His work on MPT, although simple, opened the flood gates for what today has become a prominent player in the world of finance.

We will very quickly look at one such Quantitative Model that leverages advances in mathematics:

the Black-Scholes-Merton Model for options pricing.

1.2 Black-Scholes-Merton Model

1.2.1 Options Pricing

Previously we worked with stock information, however, not all securities operate in the same manner. **Options**, contracts giving the buyer the right to buy (call) or sell (put) the underlying asset at a specific price on or before a certain date.

Options are a method for investors to add flexibility to their portfolio. Options can create leverage, can be used as a hedge bet, and can even generate recurring income. The downside is that given the time-dependent nature of options trading, they can carry increased risk to the investor - thus prompting the creation of a mathematical model to more consistently predict options pricing.

The BSM formula estimates the prices of call and put options, and was the first widely adopted mathematical formula to do so. Previously, options traders didn't possess a consistent mathematical way to value options, and empirical evidence has shown that price estimates produced by this formula are close to observed prices.

The BSM model does this by solving the Partial-Differential-Equation below, which is derived from Stochastic Calculus and the modeling of Brownian Motion - in other words, it is an equation that seeks to model pseudo-random behavior.

Myron Scholes and Robert Merton won the 1997 Nobel Prize in Economists. Fischer Black was named a contributor, since his passing rendered him ineligible to win the prize.

1.2.2 BSM Formula:

$$\partial C \over \partial t + \frac{1}{2}\sigma^{2} S^{2} \frac{\partial^2 C}{\partial C^2} + rS\frac{\partial C}{\partial S} = rC$$

$$C(S,t) = N(d_1)S_t - N(d_2)Ke^{-rt}$$

- C = Call option price
- S_t = Current stock price
- K = Strike price of the option
- r = Risk-free interest rate (a number between 0 and 1)
- σ = Volatility of the stocks return (a number between 0 and 1)
- t = Time to option maturity (in years)
- N = normal cumulative distribution function

$$d_1 = \frac{1}{\sigma\sqrt{t}}\left[\ln\left(\frac{S}{K}\right) + t\left(r + \frac{\sigma^2}{2}\right)\right]$$

$$d_2 = \frac{1}{\sigma\sqrt{t}}\left[\ln\left(\frac{S}{K}\right) + t\left(r - \frac{\sigma^2}{2}\right)\right]$$

$$N(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{1}{2}z^2} dz$$

```
We will not be spending time to understand this model in mathematical detail but, it will be ill
```

In the next two cells we define d_1, d_2, and the put/call functions of the BSM Model

We are going to again use Datareader to build a dataframe to price Netflix stock options for an option that expires on December 18th, 2022.

```
The Option Price is:  305.1146194579531
```

The cell below builds BSM model functions to return the volatility of an option and applies it to the Netflix test stock

```
Implied Volatility: 33.800000000000026 %
```

Now we can apply to model to our 4 stocks from earlier

```
Call Price:
{'AAPL': 0.015943519764232694, 'AMZN': 2972.496676275921, 'GOOGL':
2388.276705572797, 'FB': 29.008906419005072}


--------------------------------------------------------------------------------


Implied Volatility:
{'AAPL': '26.60000000000002 %', 'AMZN': '0.1 %', 'GOOGL': '0.1 %', 'FB':
'31.300000000000022 %'}
```

Now that we have seen the power of having a computer by our side when making investments, we understand how impactful an operating-scale quantum computer would be to any financial competitor.

1.3 Quantum Financial Optimization

Quantum Computers are a world of high returns and high risks, which happen to be the specialty of investors and financial services. Ultimately, trading stocks and options is an arena defined by fierce competition where success is decided by a microsecond of difference. Above we used classical computing to wrangle 4 stocks with a total of 2,000 original stock price data points, and we saw how difficult it is to predict the volatility of such a portfolio. Now imagine instead of 4 stocks we are manging 400 and instead of analyzing stock prices by the day, we want to analyze them by the *second* and it is easy to picture the benefits of having qubits at your disposal.

A Quantum Computer is well suited to this task because above all Quantum Computing excels at parallel computations - the exact tool needed to process millions of pseudo-random variable with billions of dollars on the line. In other words, to neglect the emergence of Quantum Finance would be to play basketball in Chuck Taylors while everyone else is playing in Air Jordans.

Below we are going to run through a Quantum Portfolio Optimization (QPO) algorithm you can run today using the Qiskit libraries. We are also going to compare this to a classical method to see if we are indeed generating useful results.

To begin we will load in our needed packages

1.4 load in the data from the CSV –> Remove before publishing

[52]:

Date	AAPL	AMZN	GOOGL	FB
2016-01-04	24.286827	636.989990	759.440002	102.220001
2016-01-05	23.678221	633.789978	761.530029	102.730003
2016-01-06	23.214842	632.650024	759.330017	102.970001
2016-01-07	22.235075	607.940002	741.000000	97.919998
2016-01-08	22.352644	607.049988	730.909973	97.330002

1.5 Begin QPO

We want to solve the following mean-variance portfolio optimization problem for n assets:

$$\min_{x \in \{0,1\}^n} q x^T \Sigma x - \mu^T x$$

subject to: $1^T x = B$

- $x \in \{0,1\}^n$: Binary vector denoting chosing assets,
- $\mu \in \mathbb{R}^n$: Asset Expected Returns,
- $\Sigma \in \mathbb{R}^{n \times n}$: Asset Covariance,
- $q > 0$: Risk tolerance of the investor
- B: Budget, i.e. the number of assets to be selected out of n.

Such That: - the vector of portfolio weights is normalized (=1), - the full budget B has to be spent: we must choose exactly B assets.

We are going to use the Qiskit Variational Quantum Eigensolver (VQE) to optimize the same portfolio as before in the classical examples. In a nutshell, VQE uses the variational principle of quantum mechanics to find the minimum eigenvalue of a given operator.

In the case of a state described by a Hamiltonian H, this approach gives the ground state of the specified system.

We can express the principle mathematically as: $\lambda_{min} = \langle H \rangle_\psi = \langle \psi | H | \psi \rangle = \sum_{i=1}^{N} \lambda_i | \langle \psi_i | \psi \rangle |^2$

Our operator will be built using the Qiskit_Finance package and will convert the above mean-variance problem into an operator for which we can find the minimum eigenvalue and subsequently eigenvector which will be our optimum portfolio! It does this by modeling the problem as an Ising Hamiltonian. Although that subject is deserving of its own blog, just think of it as creating a Hamiltonian defined in binary terms (on/off, 0/1).

After importing our required packages we next need to calculate the mean and covariance of our stock data. Thankfully, Pandas Dataframes have a built-in function we can call to calculate the mean price of each stock as well as the covariance matrix

```
MU:
AAPL      30.096189
AMZN     833.578032
GOOGL    851.317635
FB       136.766720
```

```
dtype: float64


SIGMA:
              AAPL           AMZN          GOOGL           FB
AAPL     40.843961     995.007772     656.919581   147.323755
AMZN    995.007772   28000.693276   16831.350250  3825.909816
GOOGL   656.919581   16831.350250   11337.098379  2443.373784
FB      147.323755    3825.909816    2443.373784   581.653846
```

Now we set the Risk Factor (q) of our investor, as well as the number of stocks (assets) we will optimize over. The number of assets is also the number of qubits initiated in the computation.

Finally, we use the portfolio module of Qiskit_Finance to generate our mean-variance operator, here represented by the variable Op.

This cell is simply to organize the data for optimizations as well as print the final results in a way we can easily understand.

Let's first run the classical NumPy minimum eigensolver function from Qiskit on our Op operator. This algorithm will run a classical search through viable eigenfunctions to minimize the eigenvalue.

```
Optimal: selection [1 0 0 0], value -5.6742

----------------- Full result ---------------------
selection       value           probability
-------------------------------------------------
 [1 0 0 0]      -5.6742         1.0000
 [1 1 1 1]      43044.2711              0.0000
 [0 1 1 1]      41242.6942              0.0000
 [1 0 1 1]      8213.2347               0.0000
 [0 0 1 1]      7414.6655               0.0000
 [1 1 0 1]      18283.3959              0.0000
 [0 1 0 1]      17146.7386              0.0000
 [1 0 0 1]      291.7097                0.0000
 [0 0 0 1]      158.0602                0.0000
 [1 1 1 0]      36461.6036              0.0000
 [0 1 1 0]      34815.3504              0.0000
 [1 0 1 0]      5464.4769               0.0000
 [0 0 1 0]      4821.2316               0.0000
 [1 1 0 0]      14152.1022              0.0000
 [0 1 0 0]      13170.7686              0.0000
 [0 0 0 0]      16.0000         0.0000
```

Let's now compare that to the results we get from using Qiskit's Variational Quantum Eigensolver.

1.6 VQE

Since we are running a quantum algorithm, we must utilize a Qiskit backend in order to simulate the effects of a quantum computer. There are different backends you can use other than Aer.

VQE utilizes a classical optimizer to search through the possible eigenfunction trials states of the Hamiltonian. We will use the Constrained Optimization By Linear Approximation optimizer, or COBYLA, which is already built into Qiskit_Finance.

Next we set a random seed (in order to push the VQE towards the global minimum), initialize our qubits using the TwoLocal command (printed below), and use the VQE function to create our quantum instance for quantum simulation to run.

Finally, we run our VQE instance, optimize our portfolio on the Qiskit backend, and print our results!

```
Optimal: selection [1. 0. 0. 0.], value -5.6742

----------------- Full result ---------------------
selection       value           probability
-------------------------------------------------
 [1 0 0 0]      -5.6742         0.1442
 [0 1 0 0]      13170.7686              0.1256
 [0 1 1 1]      41242.6942              0.1239
 [0 1 1 0]      34815.3504              0.0884
 [1 1 0 1]      18283.3959              0.0778
 [1 1 0 0]      14152.1022              0.0657
 [0 0 1 1]      7414.6655               0.0540
 [0 0 0 0]      16.0000         0.0528
 [1 0 1 1]      8213.2347               0.0441
 [0 0 0 1]      158.0602                0.0426
 [1 0 1 0]      5464.4769               0.0400
 [1 1 1 1]      43044.2711              0.0389
 [1 0 0 1]      291.7097                0.0373
 [0 1 0 1]      17146.7386              0.0301
 [0 0 1 0]      4821.2316               0.0279
 [1 1 1 0]      36461.6036              0.0068
```

As you can see we return the same result as the classical example! While this may seem underwhelming, that is due to the relative simplicity of our example. Outside this blog on the trading floor, variables to be calculated increase exponentially and likewise need to be met with exponential computing power - precisely what Quantum Computing promises to deliver.

Often, while mired in the complex problems of the present, it is easy to lose sight of Quantum Computing's future. However, Quantum Finance is one area where real, market-altering change is occurring overnight. Just as banks are preparing themselves to play a new, distinctly Quantum game, everyone invested in Quantum Computing should prepare for the day qubits dominate Wall Street.