Portfolio Optimization Using Monte Carlo Simulation

Portfolio & Risk Management

Feb 08, 2018

6 min read

By Mandeep Kaur

In the previous blog of this series, we saw how to compute the mean and the risk (or standard deviation) of a portfolio containing '*n*' number of stocks, each stock '*i*' having a weight of 'w_i'.

In this blog, we will see how to do portfolio optimization by changing these weights. By portfolio optimization, we mean getting a portfolio that meets any of the three conditions based on the investor's requirements.

Conditions of Portfolio Optimization

Annual Returns and Standard Deviation

To simplify our analysis in this blog, we will deal with daily returns and standard deviation and will consider only 1 month of stock data (Dec 2017). However, in practice, we work with annual returns and standard deviation. The formulae for converting daily returns and standard deviation to an annual basis are as shown (assuming 252 trading days in a year):

Annual Return = Daily Return * 252 Annual Standard Deviation = Daily Standard Deviation * 252

Let us consider a portfolio consisting of four stocks in banking/financial services sector, namely: Bank of America (BAC), Goldman Sachs (GS), JP Morgan Chase & Co (JPM) and Morgan Stanley (MS).

To start with, we'll assign random weights to all four stocks, keeping the sum of the weights to be 1. We will compute the return and standard deviation of the portfolio as we did in the previous blog and simulation, we will assign random weights to the [illegible]s. One important point to keep in mind is that the sum o[illegible]eights should always sum up to 1. At every particular combination of these weights, we will compute the return and standard deviation of the portfolio and save it. We'll then change the weights and assign some random values and repeat the above procedure.

The number of iterations depends on the error that the trader is willing to accept. Higher the number of iterations, higher will be the accuracy of the optimization but at the cost of computation and time. For the purpose of this blog, we will restrict ourselves to 10000 such iterations. Out of these 10000 results for returns and corresponding standard deviation, we can then achieve portfolio optimization by identifying a portfolio that satisfies on any of the 3 conditions discussed above.

Portfolio Optimization Process in Python

Let's start by importing relevant libraries and fetching the data for the stocks for Dec 2017.

```
#Arrange the data in ascending order
data=data.iloc[::-1]
print (data.round(2))
```

The data looks as shown:

```
              BAC      GS     JPM     MS
Date
2017-11-30  28.17  247.64  103.98  51.61
2017-12-01  28.10  248.95  104.25  51.95
2017-12-04  29.06  250.65  106.40  52.68
2017-12-05  28.93  248.33  105.17  52.01
2017-12-06  28.64  245.95  104.39  51.69
2017-12-07  28.78  248.56  104.08  52.35
2017-12-08  29.05  250.35  105.38  52.89
2017-12-11  28.94  250.13  105.07  52.77
2017-12-12  29.32  257.68  106.30  53.85
2017-12-13  28.84  255.56  104.96  53.18
2017-12-14  28.73  255.48  104.12  52.64
2017-12-15  29.04  257.17  105.59  53.10
2017-12-18  29.48  260.02  106.41  53.24
2017-12-19  29.45  256.48  105.96  52.93
2017-12-20  29.48  255.18  105.59  52.51
2017-12-21  29.82  261.01  107.27  52.88
2017-12-22  29.88  258.97  106.89  52.72
2017-12-26  29.78  257.72  106.47  52.47
2017-12-27  29.73  255.95  106.66  52.57
2017-12-28  29.80  256.50  107.23  52.65
2017-12-29  29.52  254.76  106.39  52.47
```

```
Date
2017-11-30   NaN   NaN   NaN   NaN
2017-12-01 -0.25  0.53  0.26  0.66
2017-12-04  3.42  0.68  2.06  1.41
2017-12-05 -0.45 -0.93 -1.15 -1.27
2017-12-06 -1.00 -0.96 -0.75 -0.62
2017-12-07  0.49  1.06 -0.30  1.28
2017-12-08  0.94  0.72  1.25  1.03
2017-12-11 -0.38 -0.09 -0.29 -0.23
2017-12-12  1.31  3.02  1.16  2.05
2017-12-13 -1.64 -0.82 -1.25 -1.24
2017-12-14 -0.38 -0.03 -0.81 -1.02
2017-12-15  1.08  0.66  1.41  0.87
2017-12-18  1.52  1.11  0.77  0.26
2017-12-19 -0.10 -1.36 -0.42 -0.58
2017-12-20  0.10 -0.51 -0.35 -0.79
2017-12-21  1.15  2.28  1.59  0.70
2017-12-22  0.20 -0.78 -0.35 -0.30
2017-12-26 -0.33 -0.48 -0.40 -0.47
2017-12-27 -0.17 -0.69  0.19  0.19
2017-12-28  0.24  0.21  0.53  0.15
2017-12-29 -0.94 -0.68 -0.79 -0.34
```

We will now calculate the mean return of all stocks and the covariance matrix.

```
#Calculate mean returns and covariances of all four the stoc
mean_returns = stock_ret.mean()
cov_matrix = stock_ret.cov()
print (mean_returns)
print (cov_matrix)
```

containing all zeroes and will save the simulatio[...]s in this array. The number of columns in the array is 7 to [...] portfolio return, standard deviation, Sharpe Ratio and the weights of all stocks. The number of columns will change with the number of stocks in the portfolio as we have to store the weights for all the stocks. That's why we use the 'len function' while defining the array. The number of rows in the array is equal to the number of iterations.

```
#Set the number of iterations to 10000 and define an array t
num_iterations = 10000
simulation_res = np.zeros((4+len(stock)-1,num_iterations))
```

Let’s now move on to the iterations.

```
for i in range(num_iterations):
#Select random weights and normalize to set the sum to 1
        weights = np.array(np.random.random(4))
        weights /= np.sum(weights)
```

```
#Calculate Sharpe ratio and store it in t      ay
        simulation_res[2,i] = simulation_re  ,i] / simulati
```

```
#Save the weights in the array
        for j in range(len(weights)):
                simulation_res[j+3,i] = weights[j]
```

We then save the output in a '*pandas data frame*' for easy analysis and plotting of data.

```
sim_frame = pd.DataFrame(simulation_res.T,columns=['ret','st
print (sim_frame.head())
print (sim_frame.tail())
```

```
#Spot the position of the portfolio with      t Sharpe Rati
max_sharpe = sim_frame.iloc[sim_frame['sh      .idxmax()]
```

```
#Spot the position of the portfolio with minimum Standard De
min_std = sim_frame.iloc[sim_frame['stdev'].idxmin()]
print ("The portfolio for max Sharpe Ratio:\n", max_sharpe)
print ("The portfolio for min risk:\n", min_std)
```

```
The portfolio for max Sharpe Ratio:
ret       0.002224
stdev     0.010542
sharpe    0.210910
BAC       0.822040
GS        0.147847
JPM       0.009119
MS        0.020994
Name: 9294, dtype: float64
The portfolio for min risk:
ret       0.001010
stdev     0.009090
sharpe    0.111160
BAC       0.008306
GS        0.019566
JPM       0.356755
MS        0.615373
Name: 2260, dtype: float64
```

```
#Plot a red star to highlight position of the portfolio with
plt.scatter(max_sharpe[1],max_sharpe[0],marker=(5,1,0),color
```

```
#Plot a blue star to highlight position of the portfolio wit
plt.scatter(min_std[1],min_std[0],marker=(5,1,0),color='b',s
plt.show()
```

In the output, the red star shows the portfolio with the maximum Sharpe ratio and the blue star depicts the point with the minimum standard deviation.

above. We can select the portfolio with maximur n for a given risk or a portfolio with minimum risk for a given retu or we can simply select the portfolio with maximum Sharpe ratio.

Summary

Just to summarize our complete analysis, we first downloaded the stock price data for one month and computed the mean return of all the stocks and the covariance matrix (which is used in computing the standard deviation of the portfolio). We then ran a Monte Carlo Simulation to compute the risk and return of the portfolio by randomly selecting the weights of the portfolio. We then identify the optimal portfolio based on the Sharpe ratio or other conditions.

Next Step

Our next blog covers various aspects of the portfolio performance evaluation and portfolio performance measurement. It starts with

ion and measurement are necessary. Then explains how

Disclaimer: *All investments and trading in the s arket involve risk. Any decisions to place trades in the financial markets, including trading in stock or options or other financial instruments is a personal decision that should only be made after thorough research, including a personal risk and financial assessment and the engagement of professional assistance to the extent you believe necessary. The trading strategies or related information mentioned in this article is for informational purposes only.*

Download Python Code

- Portfolio Optimization Using Monte Carlo Simulation - Python Code

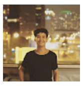
Login to Download

Jan 29, 2018

Portfolio Analysis: Calculating Risk and Returns, Strategies and More

Apr 27, 2018

Value at Risk (VaR) Calculation in Excel and Python



Tanapat Kamsaiin • 去年

Cannot download the python code.

QuantInsti • 去年

Hi! Thank you for bringing it to our notice. We have fixed the same.

Home Courses Trading Platform Contact Us

Login/Sign Up

Career Growth Tutorials

Options Trading

Portfolio & Risk Management

Python For Trading

Sentiment Trading

Technical Indicators

About

QuantInsti
Quantra

Our cookie policy