Portfolio Optimisation using Monte-Carlo Simulation

13.11.2021

Satyam Kumar
18HS20030

Overview

We want to invest the investment money in such a way that we can earn the highest amount of return by taking the least amount of possible risk. This is the optimum portfolio.

The goal of this project is to allocate $1000 to companies of our choice. To get the most optimal portfolio, we will run a monte carlo simulation for 10,000 portfolios. The higher the number of portfolios we simulate, the closer we get to the optimal distribution.

We need to find the optimum portfolio. The optimum portfolio is the one that generates the highest return for the lowest risk.

The companies chosen for this simulation are:

`ZM` : Zoom Video Communications, Inc. Class A Common Stock

`UBER` : Uber Technologies, Inc. Common Stock

`SWI` : SolarWinds Corporation Common Stock

`RNG` : RingCentral, Inc. Class A Common Stock

`CRWD` : CrowdStrike Holdings, Inc. Class A Common Stock

`WORK` : Slack Technologies, Inc.

`SYMC` : Symantec Corporation

Access the code (github repo) by clicking here.

Bit of finance

There exists a risk-free rate which is the rate that an investor earns on his/her investment without taking any risk, such as in buying government treasury bills. There is a tradeoff between risk and return. If an investor is expecting to invest in a riskier investment option than the risk-free rate then he/she is expecting to earn more in return. This is to compensate for the risk that the investor is taking. An investor can start his/her investment journey by buying equities of different companies. Let me refer to these transactions as assets from now on. We can obtain the historical stock prices of the assets and calculate their risk and return. As we start to add assets in our portfolio, we start realising that some

of these assets are somewhat correlated with each other. Consequently, the assets can have a relationship with each other hence they can co-move with each other.

A portfolio groups/holds the assets together. We know that when the assets are grouped together, their net risk isn't simply a straight sum of their individual risks. The trick here is to treat the portfolio as an asset and calculate its collective risk and return.

Amongst all the portfolios generate, we will indicate the following three portfolios:

1. **Portfolio with the maximum Sharpe ratio** : This is our optimum portfolio

2. **Portfolio with the minimum risk :** This is the portfolio for investors who are risk-aware and do not want to take high risks.

3. **Portfolio where the investment in each asset is allocated equally**

Hence, there will be 10'001 portfolios in total:

- Portfolio With Assets Equally Allocated
- 10,000 Randomly Generated Portfolios

Metrics Used

$$Return\ For\ Each\ Day\ =\ Log\left(\frac{Today's\ Price}{Yesterday's\ Price}\right)$$

$$Asset\ Expected\ Return\ =\ \frac{Sum(Returns)}{Total\ Number\ Of\ Observations}$$

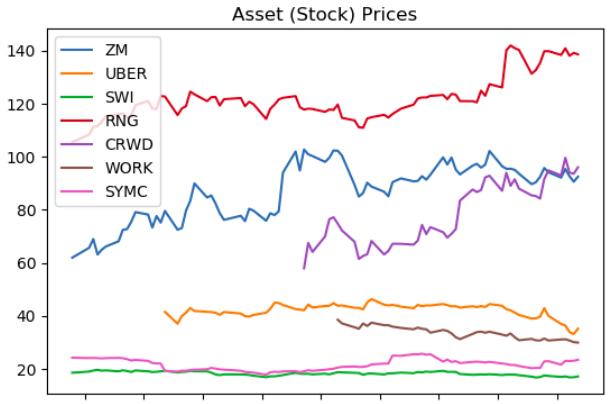
$$Expected\ Portfolio\ Return = Sum\ (Weight\ x\ Asset\ Expected\ Return) Of\ Each\ Asset$$

$$Volatility\ =\ Square\ Root\ (Weights\ Vector\ *\ Covariance\ Matrix * Weights\ Vector\ Transposed)$$

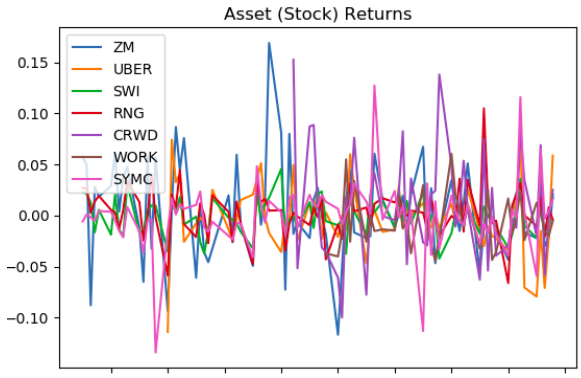
$$Sharpe\ Ratio\ =\ \frac{Portfolio\ Expected\ Return\ -\ Risk\ Free\ Rate}{Portfolio\ Volatility\ (Standard\ Deviation)}$$

Steps of the process

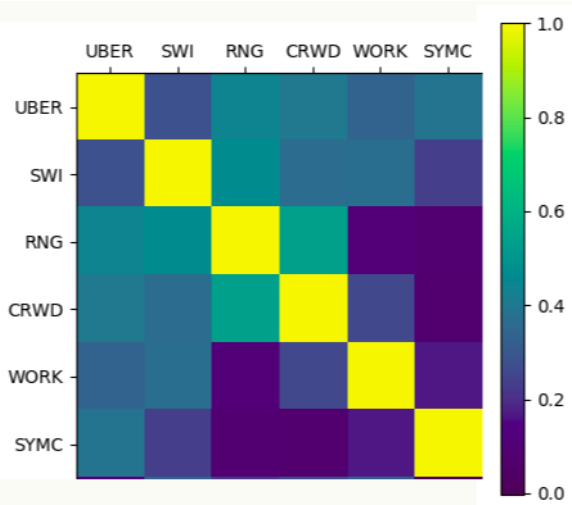
1. Fetch the Stock Prices



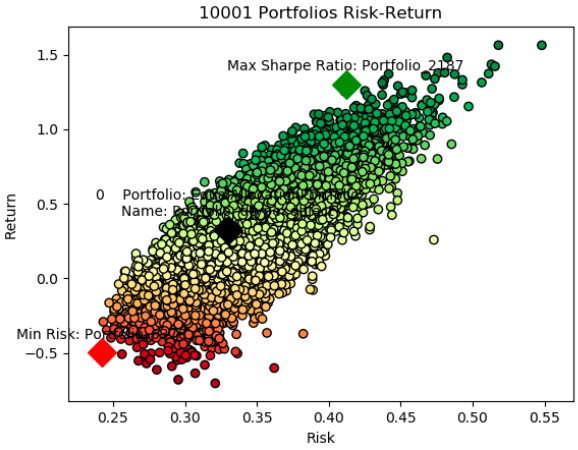
2. Generate the Returns

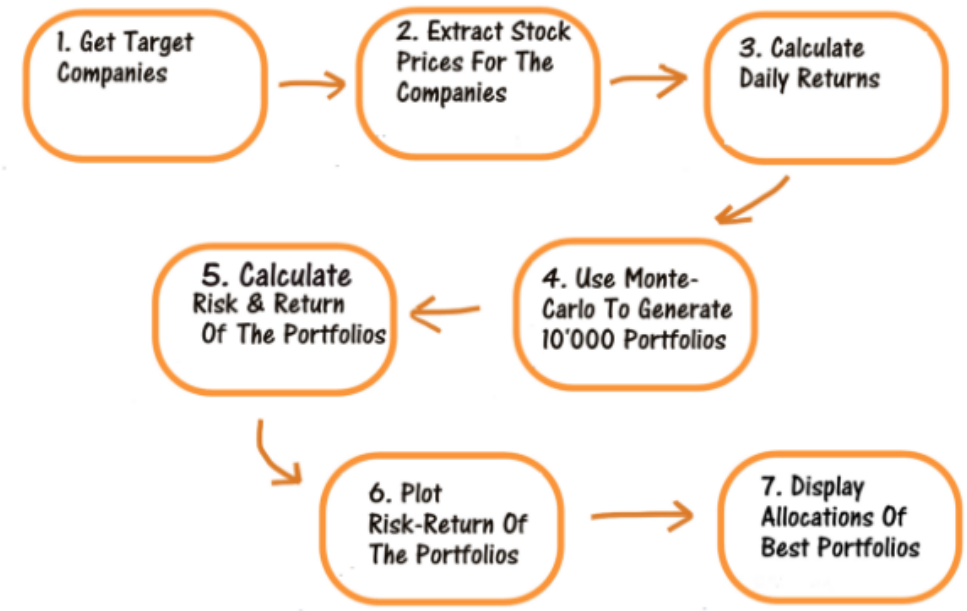


3. Generate the covariance matrix



4. Simulate all portfolios





Code Walkthrough

`settings.py :` code where the configurable settings are stored. change the tickers in *MyCompanies* list to change companies. It will output the data in an excel spreadsheet which will be stored in *PortfolioOptimisationPath = './PortfolioOptimisation.xlsx'*

`object_factory.py :` instantiate the required objects for us as the application is heavy on object-oriented design

`companies_extractor.py :` returns the static list of companies which are configured in the settings.py file.

`calculator.py :` metrics calculator to calculate risk, returns and sharpe-ratio for all the portfolios generated

`monte_carlo_simulator.py :` main class that generates 10,001 portfolios for us

`chart_plotter.py :` this file prepares the required charts for us

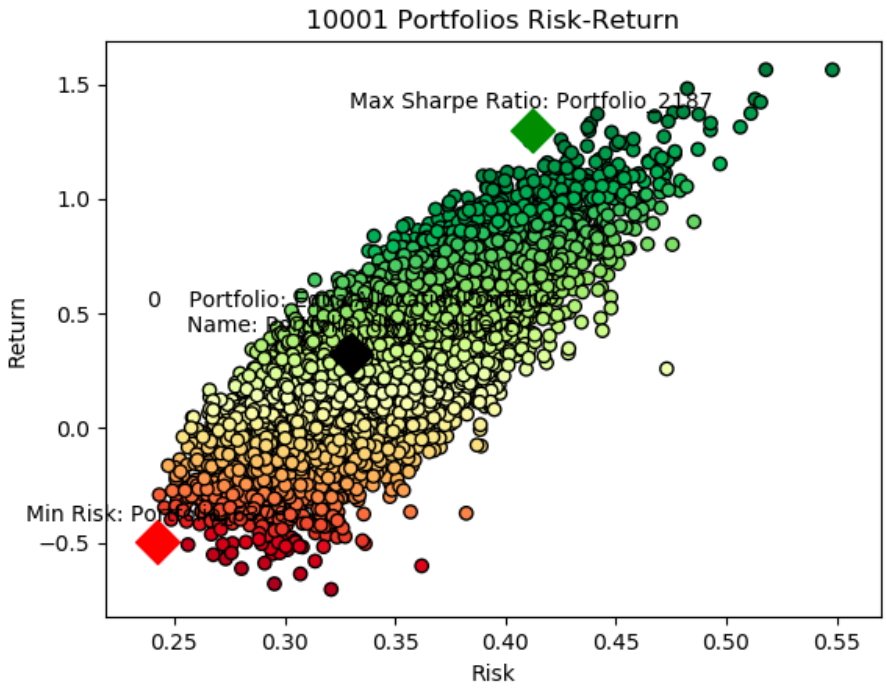
`file_repository.py :` the intermediate data along with the final portfolios are stored in an excel file, this saving work is performed by this file

`mappers.py :` maps the data which is the output of the Monte Carlo simulator in the required input format for the Chart Plotter

`main.py :` glues everything together using *generate_optimum_portfolio()* function

1. It first instantiates the required objects
2. Gets companies extractor to get the required companies.
3. Gets the prices extractor to retrieve the asset prices
4. Calculates daily returns of the assets along with the covariance
5. Calls Monte Carlo simulator to generate the portfolios along with their return, risk and Sharpe ratio.
6. The chart plotter then plots the required data and the file repository saves the data for analysis.

Monte-Carlo Simulation Result



The code plots the portfolios on a Risk-Return chart, where the y-axis is the return of the portfolio and x-axis is the risk of the portfolio.

We can see that the three portfolios are marked. The **green diamond** is our optimum portfolio.

Green - Diamond Portfolio: This is our chosen optimum portfolio. For each portfolio, we generated the Sharpe Ratio. This is the portfolio with the maximum Sharpe ratio. It gave us the highest return for the lowest possible risk. This portfolio is named Portfolio 2187.

Black - Diamond Portfolio: This is our equally allocated portfolio. We can see that the portfolio with an equal allocation is somewhere in the middle. It is not as good as the

portfolio 2187 which is marked as the Green diamond. We can also see there are many portfolios that generated higher returns for the same risk we would take if we invested our investments equally. Hence, it means the application generated superior portfolios for us. Matter of fact, all portfolios that are above the black diamond on y-axis with the same x-axis value, are better portfolios than the equally allocated portfolio because for the same risk, they generate higher returns.

Red - Diamond Portfolio: This is the portfolio that generated the lowest risk.

Portfolio_2187

This is our optimum portfolio.

Symbol	Portfolio_2187
ZM	11%
UBER	3%
SWI	4%
RNG	47%
CRWD	30%
WORK	0%
SYMC	6%
Return	**130%**
Risk	**41%**
Sharpe-Ratio	**316%**

It is suggesting we should not invest in WORK.It is also highlighting that we should invest most of our money in RNG and CRWD.

The portfolio return of the portfolio is 130% by taking the risk of 41%. It's Sharpe Ratio is approximately 316% (≈ 130/41).