GB730 Prescriptive Modeling & Optimization for Business Analytics

Strategic Stock Portfolio Optimization in the S&P 500: Balancing Risk and Return

Annie Chen and Chin-Yi Li

Table of contents

01

Introduction

Our Goal

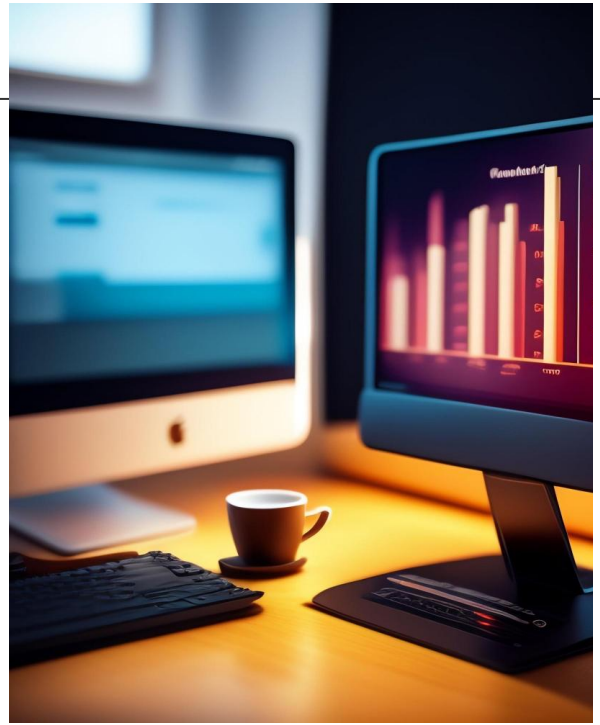
In this optimization model, our objective is to optimize investment strategies within the S&P 500 while maintaining a balance in investment risk.

S&P 500

The S&P 500 has proven itself as a reliable indicator of the U.S. stock market's health and vitality over time. Encompassing the performance of sectors like technology, finance, healthcare, and more, the S&P 500 truly captures the pulse of American commerce.

Input Data

We collected stock historical daily close prices data for S&P 500 companies from Yahoo Finance, and the data collection period spans from December 12, 2018, to December 12, 2023 with a total of 5 years.



02

Optimization Model

Optimization Model

Decision Variables

Allocation: the % of the total value allocated to stocks

Constraints

1. Portfolio standard deviation must be within the specified risk limit.
2. The total allocation must sum up to 100%.

Objective

Maximize the annualized rate of return

Constraints

We utilize the portfolio standard deviation as our measure of risk. The definition and formula for portfolio standard deviation are presented below.

$$\sigma_{port} = \sqrt{\sum_{i=1}^{n} w_i^2 \sigma_i^2 + \sum_{i=1}^{n} \sum_{i=1}^{n} w_i w_j Cov_{ij}}$$

where :

σ_{port} = the standard deviation of the portfolio

W_i = the weights of the individual assets in the portfolio, where weights are determined by the proportion of value in the portfolio

σ_i^2 = the variance of rates of return for asset i

Cov_{ij} = the covariance between the rates of return for assets i and j, where $Cov_{ij} = r_{ij}\sigma_i\sigma_j$

Objective

To compute the annualized rate of return, we initially multiply the average daily rate of return by the allocation weight. The formula is presented below.

$$Annualized\ rate\ of\ return\ = (1 + \sum_{i=1}^{n} Average\ Daily\ Rate\ of\ Return \times Allocation\ Weight)^{252} - 1$$

The exponent of 252 is chosen because the NASDAQ typically has an average of 252 trading days per year.

Optimization Model Tools

Excel

We used top five companies with weighted components from S&P 500 to build a small model in Excel.

- **Solver** – NonLinear
- Formula, Array multiplication

Python

Then we scaled up to 500 companies from all the S&P 500 index to fit in the Pyomo optimization model.

- **For loop** – repetition for 500 stocks
- **Packages** – work efficiently (**Pyomo**)

03

Application:
Hypothetical Client

Hypothetical Clients:

1. Ourselves - we will invest in stock after our analysis
2. People who is new to stock investment
3. People who want to invest without actively investigating or managing individual stocks, indices, or other specific investments

Excel Demo

Input Variables: Daily Stock close prices in past 5 years

Date	AAPL	MSFT	AMZN	NVDA	GOOGL
12/12/18	42.275002	109.080002	83.177002	37.224998	53.686501
12/13/18	42.737499	109.449997	82.918999	37.2225	53.676998
12/14/18	41.369999	106.029999	79.595497	36.612499	52.585499
12/17/18	40.985001	102.889999	76.045502	35.895	51.282501
12/18/18	41.517502	103.970001	77.573997	36.735001	52.170502
12/19/18	40.2225	103.690002	74.753998	34.627499	51.772999
12/20/18	39.2075	101.510002	73.041496	33.775002	51.179001
12/21/18	37.682499	98.230003	68.872498	32.392502	49.5625

CoVariance Matrix

	AAPL	MSFT	AMZN	NVDA	GOOGL
AAPL	0.000420024	0.0003033	0.0002895	0.0004424	0.0002808
MSFT	0.000303282	0.0003766	0.0003006	0.0004484	0.0002978
AMZN	0.000289465	0.0003006	0.0005063	0.0004448	0.0003012
NVDA	0.000442404	0.0004484	0.0004448	0.0010694	0.0004155
GOOGL	0.000280773	0.0002978	0.0003012	0.0004155	0.0004058
ROR AVG	0.001419607	0.0011633	0.0007	0.0025441	0.0009268

Input Variables:

- NASDAQ stock exchange stocks data
- Time frame

Here we use the top 5 stocks and 5 years historical data to demo the model.

Calculations:

- Rate on Return (RoR)
- Covariance Matrix

The model will generate RoR average and covariance matrix for further calculations.

Excel Demo

Decision Variable: Allocation Weight						**Objective: Maximize**
Risk	**AAPL**	**MSFT**	**AMZN**	**NVDA**	**GOOGL**	Year of Return (YOR)
0.15	0.25724268	0.322692	0.1316	0	0.288461	0.3192444
0.2	0.25724437	0.322686	0.13161	0	0.288463	0.319244
0.25	0.25724469	0.322686	0.13161	0	0.288464	0.319244

=MMULT(R14:V14,MMULT(R4:V8,TRANSPOSE(R14:V14)))

	Constraint 1:			**Constraint 2:**		
Variance of ROR	Standard Deviation of ROR			Allocation Weight = 100%		
0.000325002	0.28618248	<=	0.15	100%	=	100%
0.000325002	0.28618248	<=	0.2	100%	=	100%
0.000325002	0.28618248	<=	0.25	100%	=	100%

- **Input Variables:** Risk (max risk that can be tolerance by investors)
- **Decision Variables:** Allocation Weights among stocks
- **Constraint 1**: Portfolio Standard Deviation <= Risk
- **Constraint 2**: Total Allocation Weight = 100%
- **Objective**: Maximize Year of Return (YoR)

We use "**Solver**" (nonlinear engines) to solve our optimization model.

Python Demo - Pyomo

Optimization Model

Decision Variables
model.x

Objective

Calculating standard deviation of RoR

Constraints: within risk and allocation limits

Solve the model

```
[ ] def stock_model(risk, ror_average_list, covariance_matrix_list):

      #initialize a "Concrete Model"
      model = ConcreteModel()

      stock_num = len(ror_average_list)

      #initialize DVs
      model.x = Var(range(stock_num), domain = NonNegativeReals, initialize=0.2)  #model.x[i] #Initialize with non-zero values

      #define the objective - Maximum Year of Return (YoR)
      model.obj = Objective(expr = sum(ror_average_list[i] * model.x[i] for i in range(stock_num)), sense = maximize)


      #Constraint1: Standard Deviation of RoR
      def portfolio_variance(model):
            var_expr = sum(sum(model.x[i] * covariance_matrix_list[i][j] * model.x[j] for i in range(stock_num)) for j in range(stock_num))
            return var_expr + 1e-6  # Ensure positive value to avoid sqrt of negative number

      # Use Pyomo's sqrt function to avoid issues
      model.min_risk = Constraint(expr = sqrt(portfolio_variance(model)) * sqrt(252) <= risk)

      #Constraint2: Allocation Weight = 100%
      model.allocation = Constraint(expr = sum(model.x[i] for i in range(stock_num)) == 1)

      #solve model
      opt = SolverFactory('ipopt')
      opt.options['halt_on_ampl_error'] = 'yes'
      results = opt.solve(model, tee=True)

      x_star = [model.x[i].value for i in range(stock_num)]
      obj_star = value(model.obj)

      return {"x*": x_star, "obj*": obj_star}
```

Python Demo - Input & Output

Get Data from Yahoo Finance

```
[ ] def get_stock_data(symbol, start_date, end_date):
        try:
            stock_data = yf.download(symbol, start=start_date, end=end_date)
            return stock_data['Close']
        except Exception as e:
            print(f"Error fetching data for {symbol}: {e}")
            return pd.Series(name=symbol)
```

```
[ ] def get_multiple_stocks_data(symbols, start_date, end_date):
        close_prices = pd.DataFrame()

        for symbol in symbols:
            stock_data = get_stock_data(symbol, start_date, end_date)
            close_prices[symbol] = stock_data

        return close_prices
```

```
[ ] # Get the list of S&P 500 companies
    sp500_symbols = pd.read_html('https://en.wikipedia.org/wiki/List_of_S%26P_500_companies')[0]['Symbol'].tolist()
```

```
[ ] # Set the date range
    start_date = datetime.datetime.now() - datetime.timedelta(days=365 * 5)  # 5 years ago from today
    end_date = datetime.datetime.now()
```

```
[ ] # Download historical stock data for S&P 500 companies
    sp500_close_prices = get_multiple_stocks_data(sp500_symbols, start_date, end_date)
```

ics

Input:

We use functions to define the data we want and input the historical stock data from yahoo finance.

```
risk = 0.15
stock_model(risk, ror_average_list, covariance_matrix_list)
```

```
'obj*': 0.0013297549266884031}
```

```
risk = 0.5
stock_model(risk, ror_average_list, covariance_matrix_list)
```

```
'obj*': 0.0031507685616699375}
```

```
risk = 0.99
stock_model(risk, ror_average_list, covariance_matrix_list)
```

```
'obj*': 0.003683845866498079}
```

Output:

Users can define the risk they can tolerance and run the pyomo optimization model to get:

- allocation weights among each stocks
- the objective of the maximum YoR.

Benefits & Limitations

Benefits

In the Excel sections, users can easily comprehend the calculation logic and set the risk according to their requirements. In the Python sections, users don't need to search for stock information themselves, as we have already provided all the relevant code for that purpose.

Limitations

In the Excel sections, users are required to manually input stock information if they wish to explore various companies. In the Python sections, although we have included all companies in the S&P 500, users need to apply filters if they want to view specific combinations of companies.

Further Analysis & Extensions

Further Analysis

Currently, our constraints are established based on risk considerations. To enhance the depth of our analysis, we may incorporate a constraint regarding whether to buy or not, preventing the allocation from becoming too fragmented.

Extensions

In our existing model, we integrate S&P 500 stocks. To enrich and expand our model for users, we can incorporate stocks from various countries or focus on specific industries, thereby augmenting the value and usability of our model.

04

Supporting Files

Portfolio Standard Deviation

As neither of us had a financial background initially, we dedicated over fifteen hours to grasp the concepts of stocks and portfolio standard deviation. Resources we utilized included, but were not limited to, the following two methods.

Documents

- Investopedia
- Corporate Finance Institute
- Business Insider

Videos

- Finance Kid
- Teach Me Finance
- Stockholm Business School

Excel Configuration and Coding

In the sections covering Excel configuration and coding techniques, we not only rely on class materials but also refer to additional resources as outlined below.

Excel

- The Excel Hub
- Initial Return
- David Johnk

Python

- PyPI: Yahoo Finance
- GeeksforGeeks
- Pyomo Documentation

Thank you!