

Portfolio Optimizer

Project Documentation

Author: **Jaroslav Podmajerský**
Ac. year: 2024/2025
Date: January 1, 2026

Brno, January 1, 2026

Contents

1	Introduction	2
2	Mathematical approach for optimization	2
2.1	Risk and return	2
2.2	Multiple dimensions	3
2.3	Optimization	4
2.3.1	Markowitz model	4
2.3.2	General optimization	5
3	Implementation in Python	6
3.1	Overview	6
3.2	Data Fetching and Cleaning	6
3.3	Data Pre-Computing	7
3.4	Portfolio Optimization Framework	7
3.4.1	Constraints	8
3.4.2	Bounds	8
3.5	Output Interpretation	8

1 Introduction

This project serves as a tool for selecting the optimal portfolio according to a preselected metric of choice. We will cover the math behind the Markowitz method and construct such method in python while visualizing the results and computing α as our benchmark against the market.

2 Mathematical approach for optimization

2.1 Risk and return

This section is dedicated to deepen the understanding of the math behind the Markowitz model, but first lets start with a question. What is an optimal portfolio ? The answers can vary on the type of investor but in the most general case we want maximize returns whilst minimizing the risk. Now what do we mean by returns and how do we think about risk from a mathematical perspective ? Example:

Suppose the time series of a stock X which we model as a vector of values (prices) for each day :

$$X = (x_1, x_2, \dots, x_t)$$

Where t is the current day and x_1 the observation 5 years ago. Now what expected return on a given day from a day to day can we expect ? To answer this question we need to compute the % change in the values as such:

$$r_t = \frac{x_t - x_{t-1}}{x_{t-1}}$$

Thus forming a series of returns:

$$R = (r_1, r_2, \dots, r_t)$$

Note that the size of R will be smaller by one compared to the size of the given time series

$$|R| = |X| - 1$$

Now we ask the question what is the expected return on a given day ? To answer this we need to compute the expected value of the return series R

$$E[R] = \frac{1}{T-1} \sum_{i=1}^{T-1} R_i$$

T is the size of the given asset time series. Now we continue to answer the question of risk. Risk is measured by how wrong can our assumption be, or how wide can the values range from the expected value. And this can be accomplish by computing variance of the return series.

$$\text{Var}[R] = E[(r_i - E[R])^2]$$

Now in this equation is a lot to uncover but essentially we are squaring the difference between each element of the series and its expected value and then computing the expected value of the squared differences. As a simple explanation why we end up squaring the differences is to get rid of the negative values which would counter the positive differences and to amplify bigger outliers in the returns.

2.2 Multiple dimensions

Now let us construct a portfolio consisting of multiple assets and observe, calculate the return and risk

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1t} \\ x_{21} & x_{22} & \cdots & x_{2t} \\ \vdots & & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nt} \end{pmatrix}$$

Calculating the returns of such portfolio is quite intuitive, we just compute the expected value of the expected values

$$E[X] = \frac{1}{NT} \sum_{i=1}^N \sum_{j=1}^T x_{ij}$$

But this holds true only under the assumption of equally distributed weights among the portfolio, allowing us to generalize the formula by adding weights to each component, suppose weight vector

$$W = (w_1, w_2, \dots, w_N)$$

s.t.

$$\sum_{i=1}^N w_i = 1$$

then

$$A = X * W$$

Then calculating the expected return of the portfolio becomes

$$E[X] = \frac{1}{NT} \sum_{i=1}^N \sum_{j=1}^T x_{ij} w_{ij}$$

Now we follow the same process for risk (variance), but there is one catch if we would have computed the variance of a variance we would not capture the fact that one asset can move opposite to another asset whilst holding the same variance. Thus the variance of a matrix is captured by a covariance matrix denoted

$$\Sigma = \text{Cov}(A) = \begin{pmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \cdots & \text{Cov}(X_1, X_N) \\ \text{Cov}(X_2, X_1) & \text{Var}(X_2) & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(X_N, X_1) & \text{Cov}(X_N, X_2) & \cdots & \text{Var}(X_N) \end{pmatrix}$$

Where X_i is the i -th row of the matrix portfolio and

$$\text{Cov}(X, Y) = E[(x - E[X])(y - E[Y])]$$

Thus one may observe that

$$\text{Cov}(X, X) = \text{Var}[X]$$

And finally

$$\text{Var}[A] = w^T \Sigma w$$

Or in other words

$$\text{Var}[A] = \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N w_i w_j \text{Cov}(X_i, X_j)$$

2.3 Optimization

2.3.1 Markowitz model

Now we constructed and understood the main metrics of our portfolio, now what we want to achieve from the selected assets is to create a distribution in the weights s.t. we either minimize risk adjusted to wanted return or vice-versa. The simple Markowitz model accounts for the investors desired return or the risk he is willing to take. And our goal is to minimize risk while keep the desired return constant if possible

$$\min_{\mathbf{w}} w^T \Sigma w$$

s.t. we keep the constrains

$$\begin{aligned} w^T \mu &= R_{desired} \\ w^T \bar{1} &= 1 \end{aligned}$$

We can solve such task using Lagrange multipliers to model the constrains

$$\mathcal{L}(w, \lambda, \gamma) = w^T \Sigma w - \lambda(w^T \mu - R_{desired}) - \gamma(w^T \bar{1} - 1)$$

Now we have the constrains parametrized and are free to find the minimum of this function using straight differentiation with respect to the weight vector and set the equation to 0

$$\begin{aligned} \frac{\delta \mathcal{L}}{\delta w} &= 2\Sigma w - \lambda\mu - \gamma \bar{1} = 0 \\ \implies 2\Sigma w &= \lambda\mu + \gamma \bar{1} \\ \implies w &= \frac{1}{2}\Sigma^{-1}(\lambda\mu + \gamma \bar{1}) \end{aligned}$$

Once constructed the weight vector in terms of the multipliers we just need to solve for them using the original constrains

$$\begin{aligned} w^T \mu &= R_{desired} \\ w^T \bar{1} &= 1 \end{aligned}$$

We can plug back in the weight vector and solve for λ and γ

$$\begin{aligned} \frac{1}{2}(\lambda\mu + \gamma \bar{1})^T \Sigma^{-1} &= R_{desired} \\ \frac{1}{2}(\lambda\mu + \gamma \bar{1})^T \bar{1} &= 1 \end{aligned}$$

Let

$$\begin{aligned} A &= \mu^T \Sigma^{-1} \mu \\ B &= \bar{1}^T \Sigma^{-1} \mu \\ C &= \mu^T \Sigma^{-1} \bar{1} \end{aligned}$$

Then the equations after distribution become

$$\begin{aligned} \lambda A + \gamma B &= 2R_{desired} \\ \lambda B + \gamma C &= 2 \end{aligned}$$

After solving this system of linear equations, we get the following.

$$\lambda = \frac{2(R_{desired}A - B)}{CA - B^2}$$

$$\gamma = \frac{2(C - R_{desired}B)}{CA - B^2}$$

Note that we can simplify the algebraic steps because each value is represented by a scalar. Once solved for γ and λ we can compute the optimal weights by plugging the numbers back into the original equation

$$w = \frac{1}{2}\Sigma^{-1}(\lambda\mu + \gamma\bar{1})$$

The steps following the return variant are very similar but instead of minimizing the variance, we minimize the negative expected value of the portfolio.

2.3.2 General optimization

The Markowitz model is depended on the investors grasp on the market and thus a result varies on the user input. Thus a more generalized solution to the problem may be drawn, an investor can choose a metric to which he can minimize the weight vector. One can maximize (minimize the negative) so called sharp ratio which is metric that measures return to given risk

$$S_r = \frac{E[X]}{\sqrt{\text{VAR}[X]}}$$

This metric tells the investor how does much does the return surpass the risk, for example $S_r = 1$ is not adequate and thus shall not be held, $S_r = 2$ and above is considered very good. For reference the sharp ratio of S&P500 is 1.17. The procedure for general optimization is the same as before but with one less constrain, which implies one less Lagrange weight.

3 Implementation in Python

3.1 Overview

The `PortfolioOptimizer` class implements a full pipeline for portfolio construction based on historical market data. It supports:

- Automatic data download using the `yfinance` API
- Data cleaning and preprocessing
- Log-return computation and covariance estimation
- Portfolio optimization under multiple objectives:
 - Maximizing expected return
 - Minimizing volatility
 - Maximizing the Sharpe ratio
- Optional target constraints on return or volatility
- Short-selling support
- Interpretation of optimization results across timeframes

The class provides a high-level interface for classical mean–variance optimization using SciPy’s Sequential Least Squares Programming (SLSQP) solver.

3.2 Data Fetching and Cleaning

Market data is downloaded using the Python module `yfinance`. Each ticker is fetched individually using the provided period and interval:

```
df = yf.download(  
    ticker,  
    period=self._period,  
    interval=self._interval,  
    auto_adjust=False,  
    progress=False  
)
```

Downloaded data is passed through a cleaning routine that:

- Flattens MultiIndex columns (typical when downloading OHLCV)
- Removes invalid or empty datasets
- Stores the cleaned result internally

Only successfully downloaded tickers are preserved.

3.3 Data Pre-Computing

To prepare the data for optimization, daily log-returns are computed:

$$r_t = \ln \left(1 + \frac{P_t - P_{t-1}}{P_{t-1}} \right)$$

For each asset the following columns are created:

- Raw price change (**RawChange**)
- Percentage change (**PctChange**)
- Log-return (**LogChange**)

All log-returns are assembled into the matrix:

$$R = \begin{bmatrix} r_{1,1} & \cdots & r_{1,n} \\ \vdots & \ddots & \vdots \\ r_{T,1} & \cdots & r_{T,n} \end{bmatrix}$$

From this matrix the following are derived:

$$\mu = E[R] \quad (\text{mean returns}) \quad (1)$$

$$\Sigma = \text{Cov}(R) \quad (\text{covariance matrix}) \quad (2)$$

These serve as the fundamental statistical inputs to all subsequent optimization steps.

3.4 Portfolio Optimization Framework

Let:

- w denote the vector of portfolio weights,
- μ the vector of mean returns,
- Σ the covariance matrix.

The optimizer computes:

$$\begin{aligned} \text{Return}(w) &= w^\top \mu \\ \text{Volatility}(w) &= \sqrt{w^\top \Sigma w} \\ \text{Sharpe}(w) &= \frac{\text{Return}(w)}{\text{Volatility}(w)} \end{aligned}$$

Depending on the selected method, the objective function for the SLSQP optimizer is:

- **Maximize Sharpe ratio:**

$$\min_w - \frac{w^\top \mu}{\sqrt{w^\top \Sigma w}}$$

- **Minimize volatility:**

$$\min_w \sqrt{w^\top \Sigma w}$$

- **Maximize expected return:**

$$\min_w -(w^\top \mu)$$

3.4.1 Constraints

Every optimization problem contains:

$$\sum_i w_i = 1$$

Additionally, users may specify a target return or target volatility. These constraints are normalized using the provided timeframe mapping from TIMEFRAME to amount of trading days in that TIMEFRAME:

$$\text{TIMEFRAME_NORM} = \begin{cases} 1 & \text{daily} \\ 21 & \text{monthly} \\ 63 & \text{quarterly} \\ 252 & \text{annually} \end{cases}$$

If a target return is specified for timeframe T :

$$w^\top \mu = \frac{R_T}{\text{TIMEFRAME_NORM}(T)}$$

If a target volatility is specified:

$$w^\top \Sigma w = \frac{\sigma_T^2}{\text{TIMEFRAME_NORM}(T)}$$

3.4.2 Bounds

By default:

$$0 \leq w_i \leq 1$$

Short selling is enabled via:

$$-1 \leq w_i \leq 1$$

3.5 Output Interpretation

The raw optimization results are scaled to the desired timeframe. Given daily return μ_d and daily volatility σ_d :

$$\begin{aligned} \mu_T &= \mu_d \cdot \text{TIMEFRAME_NORM}(T) \\ \sigma_T &= \sigma_d \cdot \sqrt{\text{TIMEFRAME_NORM}(T)} \\ \text{Sharpe}_T &= \text{Sharpe}_d \cdot \sqrt{\text{TIMEFRAME_NORM}(T)} \end{aligned}$$

The optimizer prints a readable summary including:

- Expected return for the chosen timeframe
- Volatility
- Sharpe ratio
- Non-negligible asset weights (allocations)

References

[1] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[1]