Python Workshop: Portfolio Optimisation

Ahmed Pirzada[1]

[1]University of Bristol

School of Economics
November 2025

Portfolio with only two stocks

Consider two stocks - **Stock A** and **Stock B** - with following properties:

- **Stock A** offers an annual return of 30% ($\mu_A = 30\%$), whereas **Stock B** offers an annual return of 20% ($\mu_B = 20\%$).

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix} = \begin{bmatrix} 30\% \\ 20\% \end{bmatrix}$$

- **Stock A** has a variance of 0.1 ($\sigma_A^2 = 0.1$), whereas **Stock B** has a variance of 0.04 ($\sigma_B^2 = 0.04$). The two stocks also have a covariance of 0.02 ($\sigma_{AB} = 0.02$).

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_A^2 & \sigma_{AB} \\ \sigma_{AB} & \sigma_B^2 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.02 \\ 0.02 & 0.04 \end{bmatrix}$$

Portfolio optimisation problem

Your problem: How to split your investment between **Stock A** and **Stock B**?

$$\boldsymbol{w} = \begin{bmatrix} w_A \\ w_B \end{bmatrix}$$

where w_A and w_B are the weights you assign to each stock in your portfolio. However you decide to choose $\boldsymbol{w}$, you will have:

Portfolio Return: $R_p(\boldsymbol{w}) = w_A \cdot \mu_A + w_B \cdot \mu_B = \underbrace{\boldsymbol{w}^\top \boldsymbol{\mu}}_{\text{matrix representation}}$

Portfolio Risk: $\sigma_p^2(\boldsymbol{w}) = w_A^2 \cdot \sigma_A^2 + w_B^2 \cdot \sigma_B^2 + 2 w_A w_B \sigma_{AB} = \underbrace{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}_{\text{matrix representation}}$

Two Special Cases

1 If you only invest in **Stock A** (i.e. $w_A = 1$ and $w_B = 0$), you get the maximum return but also face high risk.

$$R_p(\boldsymbol{w}) = 30\%$$
$$\sigma_p^2(\boldsymbol{w}) = 0.1$$

2 If you invest in **Stock B** (i.e. $w_A = 0$ and $w_B = 1$), you get the lowest return but also face low risk.

$$R_p(\boldsymbol{w}) = 20\%$$
$$\sigma_p^2(\boldsymbol{w}) = 0.04$$

How Do You Choose?

Choose **w** which maximises **Sharpe Ratio:**

$$S_p(\boldsymbol{w}) = \frac{R_p(\boldsymbol{w})}{\sqrt{\sigma_p^2(\boldsymbol{w})}} = \frac{w_A \cdot \mu_A + w_B \cdot \mu_B}{\sqrt{w_A^2 \cdot \sigma_A^2 + w_B^2 \cdot \sigma_B^2 + 2 w_A w_B \sigma_{AB}}} = \frac{\boldsymbol{w}^\top \boldsymbol{\mu}}{\sqrt{\boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}}}$$

Mathematically,

$$\max_{\boldsymbol{w}} \; S_p(\boldsymbol{w})$$

subject to,

$$\textbf{weights sum to 1:} \; w_A + w_B = 1;$$
$$\textbf{no short-selling:} \; 0 \leq w_{A,B} \leq 1$$

Note: To solve for $\boldsymbol{w}^*$, we need to know both $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. These come from data.

Python implementation

1. We use yfinance library to get data on stocks and calculate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

2. We do not have to calculate the optimal value for $\boldsymbol{w}^*$. Instead, the minimize() function from the scipy library does it for us.

   However, we do need to set up the objective function, the constraints, and provide an initial guess for $\boldsymbol{w}$.

   Since we are using the minimize() function, we specify the objective function as the negative of the Sharpe ratio:

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

   We use the dot() function from the numpy library to calculate the value for this function for any $\boldsymbol{w}$.