Using Linear Algebra to Improve Classical Portfolio Theory

Andy Gusty

December 11, 2023

Abstract

Mean-variance portfolio optimization, first proposed by Markowitz, is a revered theoretical method for constructing optimal portfolios of assets. However, in many practical applications, mean-variance portfolio optimization generates sub-optimal portfolios due to uncertainty in the covariance matrix. In this paper, I explore a mathematical method to improve Markowitz's model, which works by normalizing the eigienvalues of the inverse covariance matrix to alleviate overdispersion caused by uncertainty in the covariance predictions. I then compare this method via a simulation to the classical mean-variance portfolio optimization method. I found that portfolios generated with the normalized covariance matrix performed significantly better than those made with the classical model. I conclude that normalization of the covariance matrix is useful in practical applications of portfolio optimization to reduce the effects of noise generated by uncertainty in predictions.

1 Introduction

One of the most important questions that all investors face is that of which assets should be included in their portfolio. With so many different choices available, one could create a nearly infinite number of portfolios, but how can one decide which is optimal? An answer to this question came in 1952, when Harry Markowitz proposed a method called mean-variance portfolio optimization [1], which became the new standard for portfolio theory and earned Markowitz the Nobel Prize for Economics in 1990. What mean-variance portfolio optimization aims to do is maximize the rate of return for a given level of risk. Practically, this allows an investor to safely generate the maximum return possible while minimizing the risk of losing more than they can safely tolerate in the case that market conditions change for the worse.

Theoretically, mean-variance portfolio optimization is sound, but in practical implementations, asset weights are oftentimes noisy and unstable over time, leading to sub-optimal portfolios. These issues ultimately arise from two areas of uncertainty: the matrix of mean returns and the matrix of covariances. In the real world, the future returns and volatility of an asset is unknown, so they must be estimated based on past data, which is not always representative of future trends. To get around this, investors have devised a number of methods to improve Markowitz's model.

Firstly, it has been shown that uncertainty in the mean return of an asset has a greater effect on the final portfolio weight than uncertainty in the covariance [2]. As a result, most recent research throws out mean returns all together and focuses solely on creating a minimum-variance portfolio (MVP). After this is done, the only uncertainty left is that of the covariance matrix. One way that the uncertainty manifests is by creating very large positive or negative weights for some assets, which generally lead to unstable results. To correct this, most researchers require that all asset weights be positive, i.e., no short-selling (borrowing against an asset).

The final improvement, which is the focus of this paper, aims to decrease the effect of uncertainty in the covariance matrix by shrinking its inverse matrix (which is more directly used in computations, as will be seen later). In 2020, Fangquan Shi and colleagues proposed this can be done by regularizing the eigenvalues of the matrix, and then rebuilding it with the new eigenvalues using spectral decomposition [3]. All of these concepts will be explored thoroughly in subsequent sections.

The goal of this paper is to test whether Shi's modified method for portfolio optimization is an improvement upon the classical model for mean-variance portfolio optimization by comparing the two in a simulation. The simulation will have 100 rounds. Each round will work by randomly selecting 10 stocks from the S&P 500 and applying both models to create two portfolios. A third portfolio will also be made with equal weighting of all 10 stocks, which will act as a control. I will then employ a "rolling-window" procedure to compare portfolio performance over a period of 3 months. After the simulation is complete, we will compare portfolio performance based on the resulting mean returns, variance, and Sharpe ratio (a measure of the risk-reward ratio). A complete description of the logistics of the simulation can be found in section (4).

2 The Classical Model

Before delving into the math of this proposed improvement, an understanding of Markowitz's classical model is necessary. For this purpose, I will be relying heavily on *Portfolio Theory with Matrix Algebra*, published by the University of Washington [5]. For simplicity, let us create portfolios from three risky assets.

2.1 Portfolio Characteristics

Let $R_i (i = A, B, C)$ denote the return on asset i and assume that R_i follows the constant expected return (CER) model. Then, we have that

$$\begin{aligned} R_i &\sim iid N(\mu_i, \sigma_i^2) \\ cov(R_i, R_j) &= \sigma_{ij} \end{aligned} \tag{2.1}$$

Let us also define portfolio x to be,

$$\begin{aligned} x_i &= \text{share of wealth in asset } i \\ x_A + x_B + x_C &= 1 \end{aligned} \tag{2.2}$$

The return of the portfolio is the random variable,

$$R_{p,x} = x_A R_A + x_B R_B + x_C R_C \tag{2.3}$$

In matrix form, we have,

$$\boldsymbol{R} = \begin{pmatrix} R_A \\ R_B \\ R_C \end{pmatrix}, \quad \boldsymbol{x} = \begin{pmatrix} x_A \\ x_B \\ x_C \end{pmatrix}, \quad R_{p,x} = \boldsymbol{x^T R} = (x_A, x_B, x_C) \cdot \begin{pmatrix} R_A \\ R_B \\ R_C \end{pmatrix} \tag{2.4}$$

Then, the expected return of the stocks in the portfolio is

$$E[\boldsymbol{R}] = E\left[\begin{pmatrix} R_A \\ R_B \\ R_C \end{pmatrix}\right] = \begin{pmatrix} \mu_A \\ \mu_B \\ \mu_C \end{pmatrix} = \boldsymbol{\mu} \tag{2.5}$$

With the expected return of the portfolio being

$$\mu_{p,x} = E[\boldsymbol{x^T R}] = \boldsymbol{x^T} E[\boldsymbol{R}] = \boldsymbol{x^T \mu} = (x_A, x_B, x_C) \cdot \begin{pmatrix} \mu_A \\ \mu_B \\ \mu_C \end{pmatrix} \tag{2.6}$$

and the 3×3 covariance matrix of returns is

$$\begin{aligned} \text{var}(\boldsymbol{R}) &= \begin{pmatrix} \text{var}(R_A) & \text{cov}(R_A, R_B) & \text{cov}(R_A, R_C) \\ \text{cov}(R_B, R_A) & \text{var}(R_B) & \text{cov}(R_B, R_C) \\ \text{cov}(R_C, R_A) & \text{cov}(R_C, R_B) & \text{var}(R_C) \end{pmatrix} \\ &= \begin{pmatrix} \sigma_A^2 & \sigma_{AB} & \sigma_{AC} \\ \sigma_{AB} & \sigma_B^2 & \sigma_{BC} \\ \sigma_{AC} & \sigma_{BC} & \sigma_C^2 \end{pmatrix} = \boldsymbol{\Sigma} \end{aligned} \tag{2.7}$$

Notice that the covariance matrix is symmetric, meaning $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^T$. This will be important later, as this property is necessary for spectral decomposition. This property arises from the fact that $\text{cov}(a, b) = \text{cov}(b, a)$. Using this covariance matrix, we can describe the variance of the entire portfolio as

$$\sigma_{p,x}^2 = \text{var}(\boldsymbol{x^T R}) = \boldsymbol{x^T \Sigma x} = (x_A, x_B, x_C) \cdot \begin{pmatrix} \sigma_A^2 & \sigma_{AB} & \sigma_{AC} \\ \sigma_{AB} & \sigma_B^2 & \sigma_{BC} \\ \sigma_{AC} & \sigma_{BC} & \sigma_C^2 \end{pmatrix} \cdot \begin{pmatrix} x_A \\ x_B \\ x_C \end{pmatrix} \tag{2.8}$$

If there are two portfolios, $\boldsymbol{x}$ and $\boldsymbol{y}$, then the covariance between the return on portfolio $\boldsymbol{x}$ and the return on portfolio $\boldsymbol{y}$ is

$$\begin{aligned} \sigma_{xy} &= \text{cov}(R_{p,x}, R_{p,y}) = \text{cov}(\boldsymbol{x^T R}, \boldsymbol{y^T R}) \\ &= \boldsymbol{x^T \Sigma y} = (x_A, x_B, x_C) \cdot \begin{pmatrix} \sigma_A^2 & \sigma_{AB} & \sigma_{AC} \\ \sigma_{AB} & \sigma_B^2 & \sigma_{BC} \\ \sigma_{AC} & \sigma_{BC} & \sigma_C^2 \end{pmatrix} \cdot \begin{pmatrix} y_A \\ y_B \\ y_C \end{pmatrix} \end{aligned} \tag{2.9}$$

Finally, we can express the condition that the portfolio weights sum to one as

$$\boldsymbol{x^T 1} = (x_A, x_B, x_C) \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 1 \tag{2.10}$$

2.2 Global Minimum Variance Portfolio

The next step in Markowitz's process is to find the portfolio weighting that leads to minimum overall variance. That is, a portfolio, $\boldsymbol{m} = (m_A, m_B, m_C)^T$, which solves the constrained minimization problem

$$\min_{m_A, m_B, m_C} \quad \sigma^2_{p,m} = \boldsymbol{m^T \Sigma m} \text{ s.t. } \boldsymbol{m^T 1} = 1 \tag{2.11}$$

The Lagrangian to solve this optimization problem is

$$\begin{aligned} L(m_A, m_B, m_c, \lambda) &= m_A^2 \sigma_A^2 + m_B^2 \sigma_B^2 + m_C^2 \sigma_C^2 \\ &\quad + 2 m_A m_B \sigma_{AB} + 2 m_A m_C \sigma_{AC} + 2 m_B m_C \sigma_{BC} \\ &\quad + \lambda (m_A + m_B + m_C - 1) \end{aligned} \tag{2.12}$$

And the first order conditions (FOCs) for a minimum are

$$0 = \frac{\partial L}{\partial m_A} = 2 m_A \sigma_A^2 + 2 m_B \sigma_{AB} + 2 m_C \sigma_{AB} + \lambda, \tag{2.13}$$

$$0 = \frac{\partial L}{\partial m_B} = 2 m_B \sigma_B^2 + 2 m_A \sigma_{AB} + 2 m_C \sigma_{BC} + \lambda, \tag{2.14}$$

$$0 = \frac{\partial L}{\partial m_C} = 2 m_C \sigma_C^2 + 2 m_A \sigma_{AC} + 2 m_B \sigma_{BC} + \lambda, \tag{2.15}$$

$$0 = \frac{\partial L}{\partial m_C} = m_A + m_B + m_C - 1 \tag{2.16}$$

We can express the FOCs in matrix notation as

$$\boldsymbol{0} = \frac{\partial L(\boldsymbol{m}, \lambda)}{\partial \boldsymbol{m}} = 2\boldsymbol{\Sigma m} + \lambda \boldsymbol{1}, \tag{2.17}$$

$$0 = \frac{\partial L(\boldsymbol{m}, \lambda)}{\partial \lambda} = \boldsymbol{m^T \cdot 1} - 1. \tag{2.18}$$

The first equation can be solved for $\boldsymbol{m}$:

$$\boldsymbol{m} = -(1/2)\lambda \boldsymbol{\Sigma^{-1} \cdot 1}. \tag{2.19}$$

Next, multiply both sides by $\boldsymbol{1^T}$ and use the second FOC to solve for λ:

$$\begin{aligned} 1 &= \boldsymbol{1^T m} = -(1/2)\lambda \boldsymbol{1^T \cdot \Sigma^{-1} \cdot 1} \\ \lambda &= -2 \frac{1}{\boldsymbol{1^T \cdot \Sigma^{-1} \cdot 1}} \end{aligned} \tag{2.20}$$

Finally, substitute the value for λ back into the first FOC to solve for $\boldsymbol{m}$

$$\boldsymbol{m} = -(1/2)\left(-2\frac{1}{\mathbf{1^T} \cdot \mathbf{\Sigma^{-1}} \cdot \mathbf{1}}\right)\mathbf{\Sigma^{-1}} \cdot \mathbf{1} \tag{2.21}$$

So,

$$\boldsymbol{m} = \frac{\mathbf{\Sigma^{-1}} \cdot \mathbf{1}}{\mathbf{1^T} \cdot \mathbf{\Sigma^{-1}} \cdot \mathbf{1}} \tag{2.22}$$

The first three elements of $\boldsymbol{m}$ are the portfolio weights $\boldsymbol{m} = (m_A, m_B, m_C)^T$ for the global minimum variance portfolio with the expected return $\mu_{p,m} = \boldsymbol{m^T\mu}$ and variance $\sigma^2_{p,m} = \boldsymbol{m^T\Sigma m}$.

2.3 Finding Efficient Portfolios

At this point, if we follow Markowitz and assume that investors wish to find portfolios with the best expected risk-return trade-off, then we should seek to find portfolios that maximize expected return for a given level of risk as measured by portfolio variance. Let $\sigma^2_{p,0}$ denote a target level of risk. Then we seek to solve the constrained maximization problem

$$\begin{aligned} &\max_{\boldsymbol{x}} \mu_p = \boldsymbol{x^T\mu} \\ &\text{s.t.} \quad \sigma^2_p = \boldsymbol{x^T\Sigma x} = \sigma^2_{p,0}, \qquad \text{and } \boldsymbol{x^T}\mathbf{1} = 1. \end{aligned} \tag{2.23}$$

Markowitz was able to show that this problem is the same as minimizing expected risk for a given level of return. If we let $\mu_{p,0}$ denote a target expected return level, then we have the constrained minimization problem

$$\begin{aligned} &\min_{\boldsymbol{x}} \sigma^2_{p,x} = \boldsymbol{x^T\Sigma x} \\ &\text{s.t.} \quad \mu_p = \boldsymbol{x^T\mu} = \mu_{p,0}, \qquad \text{and } \boldsymbol{x^T}\mathbf{1} = 1. \end{aligned} \tag{2.24}$$

Either problem can be chosen to be solved, but typically the latter is chosen because it offer some computational conveniences and because many investors want to target returns rather than risk. The *efficient portfolio frontier* is a graph of μ_p against σ_p values for the set of efficient portfolios, which are generated by solving equation (2.24) for all possible target expected return levels $\mu_{p,0}$ above the expected return of the global minimum variance portfolio, $\boldsymbol{m}$.
To solve this problem, first form the Lagrangian function

$$L(x, \lambda_1, \lambda_2) = \boldsymbol{x^T\Sigma x} + \lambda_1(\boldsymbol{x^T\mu} - \mu_{p,0}) + \lambda_2(\boldsymbol{x^T}\mathbf{1} - 1). \tag{2.25}$$

Notice there are two Lagrange multipliers λ_1 and λ_2 since there are two constraints ($\boldsymbol{x^T\mu} = \mu_{p,0}$ and $\boldsymbol{x^T}\mathbf{1} = 1$). The FOCs for a minimum are the linear equations

$$\frac{\partial L(\boldsymbol{x}, \lambda_1, \lambda_2)}{\partial \boldsymbol{x}} = 2\mathbf{\Sigma}\boldsymbol{x} + \lambda_1\boldsymbol{mu} + \lambda_2\mathbf{1} = 0, \tag{2.26}$$

$$\frac{\partial L(\boldsymbol{x}, \lambda_1, \lambda_2)}{\partial \lambda_1} = \boldsymbol{x^T\mu} - \mu_{p,0} = 0, \tag{2.27}$$

$$\frac{\partial L(\boldsymbol{x}, \lambda_1, \lambda_2)}{\partial \lambda_2} = \boldsymbol{x^T}\mathbf{1} - 1 = 0. \tag{2.28}$$

We can represent this system of linear equations in matrix form as

$$\begin{pmatrix} 2\boldsymbol{\Sigma} & \boldsymbol{\mu} & \mathbf{1} \\ \boldsymbol{\mu^T} & 0 & 0 \\ \mathbf{1^T} & 0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mu_{p,0} \\ 1 \end{pmatrix} \tag{2.29}$$

or

$$\boldsymbol{A}\boldsymbol{z}_x = \boldsymbol{b}_0, \tag{2.30}$$

with solution

$$\boldsymbol{z}_x = \boldsymbol{A}^{-1}\boldsymbol{b}_0. \tag{2.31}$$

The first three elements of $\boldsymbol{z}_x$ are the portfolio weights $\boldsymbol{x} = (x_A, x_B, x_C)^T$ for the minimum variance portfolio with expected return $\mu_{p,x} = \mu_{p,0}$. If $\mu_{p,0}$ is greater than or equal to the expected return on the global minimum variance portfolio, then $\boldsymbol{x}$ is an efficient portfolio.

2.4 Computing the Efficient Frontier

According to Markowitz, it can be shown that any minimum variance portfolio can be created as a convex combination of any two minimum variance portfolios with different target expected returns. I will not be showing the proof of this statement as it is out of the context of this paper. However, this property is quite remarkable in that it allows us to compute the entire portfolio frontier in (μ_p, σ_p) space using only two efficient portfolios. The following describes this process for three risky assets using matrices.

Let $\boldsymbol{x} = (x_A, x_B, x_C)^T$ and $\boldsymbol{y} = (y_A, y_B, y_C)^T$ be any two minimum variance portfolios with different target expected returns $\boldsymbol{x^T\mu} = \mu_{p,0} \neq \boldsymbol{y^T\mu} = \mu_{p,1}$. Let α be any constant and define the portfolio $\boldsymbol{z}$ as a linear combination of portfolios $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\begin{aligned} \boldsymbol{z} &= \alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y} \\ &= \begin{pmatrix} \alpha x_A + (1-\alpha)y_A \\ \alpha x_B + (1-\alpha)y_B \\ \alpha x_C + (1-\alpha)y_C \end{pmatrix} \end{aligned} \tag{2.32}$$

Then $\boldsymbol{z}$ is a minimum variance portfolio with expected return and variance given by

$$\mu_{p,z} = \boldsymbol{z^T\mu} = \alpha\mu_{p,x} + (1-\alpha)\mu_{p,y}, \tag{2.33}$$

$$\sigma^2_{p,z} = \boldsymbol{z^T\Sigma z} = \alpha^2\sigma^2_{p,x} + (1-\alpha)^2\sigma^2_{p,y} + 2\alpha(1-\alpha)\sigma_{x,y}, \tag{2.34}$$

where

$$\sigma^2_{p,x} = \boldsymbol{x^T\Sigma x}, \sigma^2_{p,y} = \boldsymbol{y^T\Sigma y}, \sigma_{x,y} = \boldsymbol{x^T\Sigma y}. \tag{2.35}$$

If $\mu_{p,z} \geq \mu_{p,m}$, where $\mu_{p,m}$ is the expected return on the global minimum variance portfolio, then portfolio $\boldsymbol{z}$ is an efficient portfolio.

To recap, the process for the classical mean-variance portfolio optimization model is as follows:

1. Compute the global minimum variance portfolio, $\boldsymbol{m}$, by solving equation (2.22). The expected rate of return of this portfolio is $\mu_{p,m}$
2. Select a target rate of return $\mu_{p,0} \geq \mu_{p,m}$ and compute the minimum variance portfolio, $\boldsymbol{x}$, for that target rate of return by solving equation (2.29).
3. Repeat Step 2 with a new rate of return, $\mu_{p,1} \geq \mu_{p,m}$.
4. Now, we can compute any efficient portfolio, $\boldsymbol{z}$, for any target rate of return desired by using equation (2.32).

3 Improving the Classical Model

The improvement to the classical model proposed by Shi and colleagues revolves around the idea of shrinking the inverse covariance matrix via its eigenvalues. Both the covariance matrix and its inverse could be chosen to be shrunk, but since the inverse is used to solve equation (2.22) and (2.29), it makes the most logistical sense to shrink the inverse covariance matrix. To shrink the eigenvalues, the Schatten q-norm is used, which is the q-norm of the vector of singular values of a matrix. For simplicity, let $\boldsymbol{\Omega} = \underset{\sim}{\Sigma}^{-1}$. Then, the Schatten q-norm is defined as

$$||\boldsymbol{\Omega}||_q = \left(\sum_{i=1}^{p} m_i^q \right)^{1/q} \tag{3.1}$$

Where m_i is the ith largest eigenvalue of the matrix $\boldsymbol{\Omega}$. Now, we face the difficulty of shrinking these eigenvalues while still providing a reliable estimate of $\boldsymbol{\Omega}$. We do this by minimizing a loss-function, which is essentially computes the distance between a desired output and the output received. In this model, Shi minimizes the negative log-likelihood loss function to make sure we are not losing information from the original matrix as we normalize its eigenvalues. We do this with

$$\begin{aligned} \min_{\Omega} \quad & \operatorname{tr}(\boldsymbol{\Omega S}) - \log|\boldsymbol{\Omega}| \\ \text{s.t.} \quad & \left(\sum_{i=1}^{p} m_i^q \right)^{1/q} \leq \delta. \end{aligned} \tag{3.2}$$

Where $\boldsymbol{S}$ is the sample covariance matrix (which is used to estimate the true value of $\boldsymbol{\Sigma}$, since it is unknown) and δ is an arbitrary threshold. The top line of this equation is the negative log-likelihood loss function. The bottom line simply shows that we are trying to minimize Schatten q-norm. Note that when $q = 1$, the Schatten norm restrict the sum of the eigenvalues of $\boldsymbol{\Omega}$ to be less than a threshold, δ. When $q = 2$, the sum of the squared eigenvalues is constrained, and when $q = +\infty$, the largest eigenvalue of the inverse covariance matrix is constrained. Because the Schatten norm is easiest to work with when $q = 1$, I will be focusing on this for the remainder of this paper.

From equation (3.2), the equivalent Lagrangian form is

$$\min_{\boldsymbol{\Omega}} \ \operatorname{tr}(\boldsymbol{\Omega S}) - \log|\boldsymbol{\Omega}| + \rho \sum_{i=1}^{p} m_i^q, \tag{3.3}$$

Where $\rho \geq 0$ is a regularization parameter. Note that the sum of a matrix's eigenvalues is equal to the sum of that matrix's trace. Thus, $\sum_{i=1}^{p} m_i^q = \operatorname{tr}(\boldsymbol{\Omega}^q)$ and we can rewrite the previous equations as

$$\operatorname{tr}(\boldsymbol{\Omega S}) - \log|\boldsymbol{\Omega}| + \rho \operatorname{tr}(\boldsymbol{\Omega}^q) \tag{3.4}$$

Note that the negative log-likelihood function is a convex function of $\boldsymbol{\Omega}$. Additionally, $\operatorname{tr}(\boldsymbol{\Omega})$ is also a convex function. This means that the objective function in equation (3.4) is strictly convex. With this property, it has been shown by Witten and Tibshiarani that this makes $\boldsymbol{\Omega}$ positive definite [4]. With this, differentiate with respect to Ω and set the expression equal to $\mathbf{0}$ to find the minimum

$$\boldsymbol{S} - \boldsymbol{\Omega}^{-1} + \rho q \boldsymbol{\Omega}^{q-1} = \mathbf{0} \tag{3.5}$$

This equation implies that $\boldsymbol{\Omega}$ and $\boldsymbol{S}$ share the same eigenvectors. So, let the spectral decomposition of the sample covariance matrix be $\boldsymbol{S} = \boldsymbol{QLQ^T}$, where $\boldsymbol{L}$ is a $p \times p$ diagonal matrix whose diagonal entries are the eigenvalues of $\boldsymbol{S}$ $(l_1, l_2, ..., l_p)$. The columns of $\boldsymbol{Q}$ correspond to the eigenvalues of the sample covariance matrix. According to equation (3.5), we have

$$l_i - 1/m_i + \rho q m_i^{q-1} = 0 \tag{3.6}$$

Since we are using $q = 1$, some simple algebra gives us the regularized eigenvalues of $\boldsymbol{\Omega}$

$$\hat{m}_i = \frac{1}{l_i + \rho}, \quad i = 1, ..., p \tag{3.7}$$

Now, we can use spectral decomposition again to build a new inverse covariance matrix with the regularized eigenvalues and the original eigenvectors. This estimate is

$$\hat{\boldsymbol{\Omega}} = \boldsymbol{Q\hat{M}Q^T} \tag{3.8}$$

Where $\hat{\boldsymbol{M}} = \operatorname{diag}(\hat{m}_1, ..., \hat{m}_p)$ is a $p \times p$ diagonal matrix and $\hat{m}_1 \geq ... \geq \hat{m}_p$. We now have a new inverse covariance matrix with less dispersed eigenvalues that can be used for calculations in the classical model. Notice that the eigenvectors do not change. This is not because the eigenvectors are without error, but rather but simply because there is no known way to improve them at this time. Additionally, we can use the matrix to solve equation (2.22) with

$$\hat{\boldsymbol{x}} = \frac{\hat{\boldsymbol{\Omega}} \cdot \mathbf{1}}{\mathbf{1^T} \cdot \hat{\boldsymbol{\Omega}} \cdot \mathbf{1}} \tag{3.9}$$

Which provides us with the global minimum variance portfolio with strictly positive weights.

3.1 Choosing the Regularization Parameter

The last point of interest is the regularization parameter, ρ. Notice that when $\rho = 0$, the resulting portfolio is simply the sample minimum variance portfolio. On the other hand, when $\rho \to \infty$, the resulting portfolio is the control portfolio (equal weighting of all assets). From looking at the previous section, it does not appear obvious what the best value for ρ is, so we must find a method to properly balance information and noise.

I selected a parameter for each iteration of the simulation through minimization of the covariance. To accomplish this, I split the data into two separate groups. One group was used to "train" an algorithm, which is this case meant estimating the inverse covariance matrix, $\hat{\mathbf{\Omega}}$, with a particular regularization parameter. The remaining data is used to estimate the resulting portfolio variance if $\hat{\mathbf{\Omega}}$ is used to create the minimum variance portfolio. To accomplish this, I manually split up the data and then used the `scipy.optimize` Python library.

Instead of trying to minimize the out-of-sample covariance, a regularization parameter could have also been selected to maximize the out-of-sample Sharpe ratio or out-of-sample return. In my case, I found comparable results when using the out-of-sample Sharpe ratio, and worse results when using the out-of-sample return.

4 Setting Up the Simulation

To evaluate performance of these two methods, I am using the publicly available data set https://www.kaggle.com/datasets/camnugent/sandp500, which contains historical data for S&P 500 stocks from 2013-2018. I will then grade performance based on the resulting out-of-sample Sharpe ratio (SR), which is a measure of how well the return of an asset compensates the investor for the risk taken.

For each round of simulation, I will employ a "rolling-window" procedure starting from the beginning of the data set. Let us denote the rolling-window length by n ($n < T$), where T is the total number of days in the data set. I will form portfolios based on three months of daily data points and then evaluate them on the subsequent one month of daily data points, giving 90 data points for the prediction phase ($n = 90$) and 90 for the evaluation phase. Then, I will repeat this process, each time shifting the data set towards the present by including data for the next month and dropping data for the earliest month. This procedure is repeated until the end of the data set is reached. At the end of this, there are $T-n$ portfolio weight vectors generated for each portfolio strategy, denotes as x_t, for $t = n, n+1, ..., T-1$.

Holding the portfolio x_t for 90 days gives the out-of-sample return at time $t + 90$ as

$$R_{p,\boldsymbol{x_t}} = \boldsymbol{x}_t^T \boldsymbol{\mu}_{t+90}, \tag{4.1}$$

where μ_{t+90} is a vector of length 10 (number of assets) of asset returns at time $t + 90$. The out-of-sample mean and variance of R_{p,x_t} are given by

$$\hat{\mu_x} = \frac{1}{T-n}\sum_{t=n}^{T-1} R_t, \tag{4.2}$$

and

$$\hat{\sigma}_x^2 = \frac{1}{T-n-1}\sum_{t=n}^{T-1}(R_t - \hat{\mu_x})^2, \tag{4.3}$$

respectively. Finally, the out-of-sample Sharpe ratio is defined as the ratio of the mean of R_t to its standard deviation, given by

$$\hat{SR} = \frac{\hat{\mu_x}}{\hat{\sigma_x}} \tag{4.4}$$

5 Running the Simulation

The simulation worked by randomly selecting 10 stocks, computing the relevant portfolios for each rolling-window, testing those portfolios, and then averaging the results. This was repeated 100 times, using new stock selections each time. Here, I will walk through an example. First, 10 stocks are randomly selected, and their average rate of return and variance over a 90 day window is calculated. Their prices are as follows

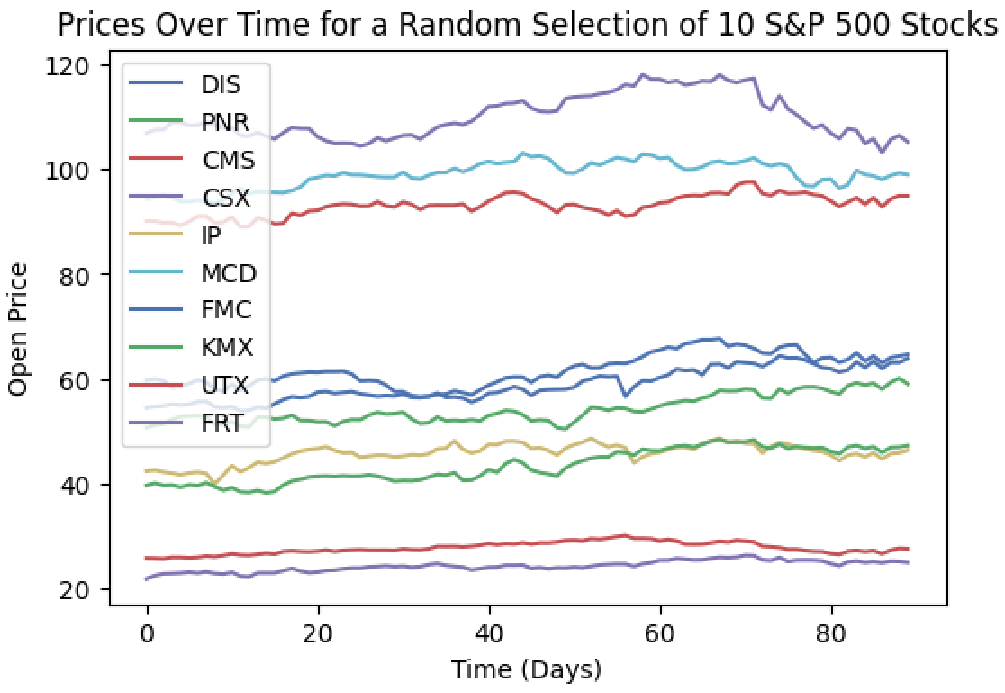

Figure 1: Prices Over 90 Days for a Selection of 10 Stocks

and their mean return and variance over this period is shown in the table below.

Stock	Mean Return	Variance
DIS	0.179641	0.012681
PNR	0.162826	0.021254
CMS	0.071821	0.009764
CSX	0.144627	0.026420
IP	0.106187	0.017526
MCD	0.050916	0.007771
FMC	0.077419	0.026022
KMX	0.187655	0.027887
UTX	0.056597	0.011263
FRT	0.011378	0.010834

The mean return also served as the expected return of each asset. Next, the sample covariance matrix was constructed and used to compute the normalized inverse covariance matrix. To accomplish this, the regularization parameter was selected to minimize the out-of-sample variance on the training data (code for this process can be found in the appendix). The resulting value was $\rho = 15.04097$. This was then used to compute the normalized inverse covariance matrix. Portfolios for the classical and improved model were both created based on equation (3.9). The weights of each portfolio are shown below.

Stock	Classical Model	Improved Model	Control Group
DIS	0.08996007	0.09342995	0.1
PNR	0.00742752	0.03942135	0.1
CMS	0.22991396	0.18501461	0.1
CSX	0.01873386	0.04682012	0.1
IP	-0.14698045	-0.06162193	0.1
MCD	0.45056164	0.32940459	0.1
FMC	0.00248144	0.03618467	0.1
KMX	-0.05037616	0.0015951	0.1
UTX	0.40236815	0.29786718	0.1
FRT	-0.00409004	0.03188435	0.1

The mean, variance, and Sharpe ratio for each portfolio were then computed on the out-of-sample data, resulting in the following values

Portfolio Type	Return	Variance	Sharpe Ratio
Classical	0.09186	0.0059066	1.195230
Improved	0.08443	0.0056446	1.123813
Control	0.070372	0.007459	0.814804

It is also helpful to visual the portfolios, plotted in the (μ, σ^2) plane.

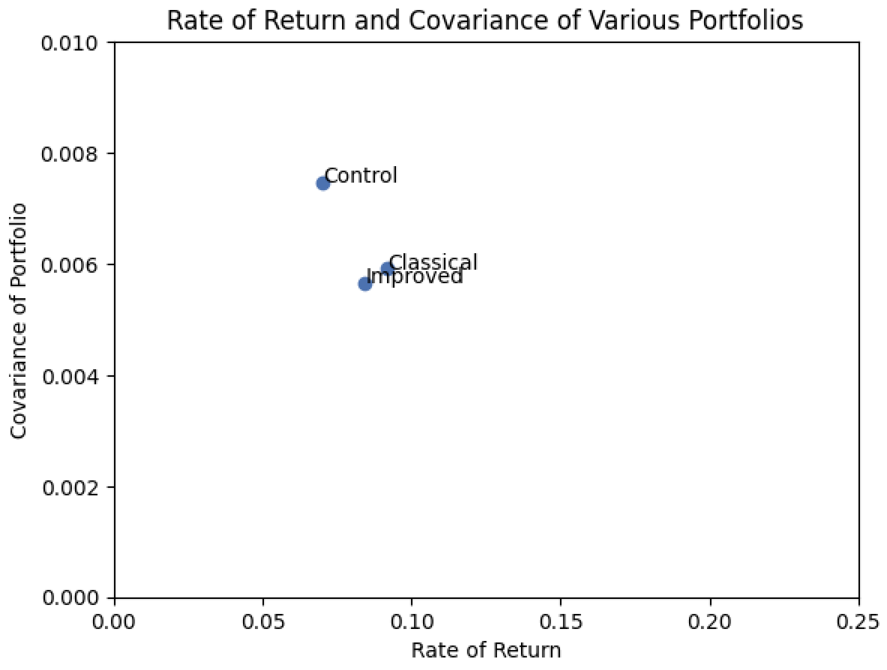

Figure 2: Portfolios Graphed in the μ-σ^2 Plane

5.1 Final Results

In the particular case shown in the previous subsection, the classical model slightly outperformed our improved model. However, this is one isolated case. This simulation was run in the manner described above 100 times, on different random selections of stocks each time. The final results of the simulation can be found below.

Portfolio Type	Return	Variance	Sharpe Ratio
Classical	0.04037	0.0067580	0.504496
Improved	0.04271	0.0060911	0.547266
Control	0.04511	0.0072473	0.510709

The difference in the Sharpe Ratio between the Improved group and the Classical and Control groups had p-values of $p_{Improved,Classical} = 0.01963$ and $p_{Improved,Control} = 0.0225$ respectively, which are both statistically significant using a threshold of $p \leq 0.05$.

Additionally, there was no statistical difference found between the Classical and Control groups ($p_{Classical,Control} = 0.5484$).

6 Discussion

As can be seen from the results above, the improvement to the classical model proposed in this paper resulted in significantly better out-of-sample Sharpe Ratios for the minimum variance portfolios compared to both the classical model and the control group. On the other hand, the classical model did not have significantly different results than simply equally weighting all 10 stocks. These results are surprising in that it appears the classical model suffers seriously from noise as a result of uncertainty in the covariance matrix. This uncertainty creates so much noise, that the classical method loses nearly all of its effectiveness in a practical setting. Because of this, normalizing the covariance matrix through some method, including the one detailed in this paper, appears to be a necessary step in mean-variance portfolio optimization.

It is interesting to note that for most selections of stocks, the portfolio produced by the Improved model typically falls somewhere between the Control and Classical groups in terms of performance. Additionally, it is usually closer in performance to the better performing of those two groups. On the other hand, the Control and Classical groups each appear to perform the best out of the 3 groups around 50% of the time respectively. What this implies is that with proper selection of the regularization parameter, the portfolio produced by Shi's Improved model is able to somehow detect the level of noise in the predictions, and adjust itself accordingly to make the resulting portfolio resemble the Classical portfolio, the Control portfolio, or something in between. In isolated examples, the resulting portfolio does not always perform optimally, but over many iterations it vastly out-competes the other models.

Based on the results of this simulation, it would be interesting to repeat the experiment with a larger selection of assets and larger portfolio sizes. Additionally, it may be useful to run the experiment with more than 5 years of historical data, which is all I could get access to. This would allow for conditions that more closely resemble the real world, where investors are using large amounts of data and typically hold an asset for at least 9-12 months. Finally, additional research is needed to find an optimal way to select the regularization parameter. In this simulation, I chose a simple method that could certainly have room for improvement. Since proper selection of the regularization parameter appears incredibly influential in portfolio performance, more research in this area could be worthwhile.

Overall, the method of alleviating over-dispersion of eigenvalues in the inverse covariance matrix appears to be extremely promising as a method to select optimal portfolios in the real-world. Practically, after an investor has narrowed down a pool of stocks that they find attractive for whatever reasons, the method explored in this paper can be used to design a portfolio that appears to provide a risk-return ratio better than the classical model for mean-variance portfolio optimization. For myself, this paper was an extremely interesting mathematical exercise and will serve as an inspiration for many further pursuits in the field of economics.

References

[1] Markowitz, Harry. “Portfolio Selection.” The Journal of Finance 7, no. 1 (1952): 77–91. https://doi.org/10.2307/2975974.

[2] Michael J. Best, Robert R. Grauer, On the Sensitivity of Mean-Variance-Efficient Portfolios to Changes in Asset Means: Some Analytical and Computational Results, The Review of Financial Studies, Volume 4, Issue 2, April 1991, Pages 315–342, https://doi.org/10.1093/rfs/4.2.315

[3] Shi, Fangquan, Lianjie Shu, Aijun Yang, and Fangyi He. “Improving Minimum-Variance Portfolios by Alleviating Overdispersion of Eigenvalues.” Journal of Financial & Quantitative Analysis 55, no. 8 (December 1, 2020): 2700–2731. doi:10.1017/S0022109019000899.

[4] Witten DM, Tibshirani R. Covariance-regularized regression and classification for high-dimensional problems. J R Stat Soc Series B Stat Methodol. 2009 Feb 20;71(3):615-636. doi: 10.1111/j.1467-9868.2009.00699.x. PMID: 20084176; PMCID: PMC2806603.

[5] Zivot, Eric. “Portfolio Theory with Matrix Algebra.” Seattle, WA: University of Washington, August 13, 2013. https://faculty.washington.edu/ezivot/econ424/portfolioTheoryMatrix.pdf.

Appendices

A Code Implementation

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import datetime
import random
import os
from pathlib import Path
import sys
import pypfopt
from pypfopt import risk_models
from pypfopt import plotting
from pypfopt import expected_returns
from pypfopt.efficient_frontier import EfficientFrontier
from scipy.optimize import minimize
import scipy
import math
import scipy.stats as st

%matplotlib inline

# STEPS
#
# Randomly select 10 stocks
# Compute expected return of portfolio
# Compute estimated covariance matrix
# Compute improved inverse covariance matrix
# Find global minimum variance portfolio with both covariance matrices
# Compare results
#
csv_path = "/kaggle/input/sandp500/individual_stocks_5yr/individual_stocks_5yr/"

###################################################################################
# RANDOMLY SELECT 10 STOCKS AND THEIR DATA
###################################################################################
def get_stock_price(csv_path):
    prices = pd.DataFrame()
    file_list = list(Path(csv_path).rglob(f"*.csv"))
    while len(prices.columns) != 10:
        stock_index = random.randint(0, 499)
        price = pd.read_csv(str(file_list[stock_index]))
        price = price[["open", "Name"]]
        prices[price.iloc[1, 1]] = price["open"]
        ## No NaN values, keep getting stocks until we have 10 with 5 full years of data
        prices = prices.dropna(axis='columns')
    return prices

##################################################################################
# COMPUTE ESTIMATE COVARIANCE MATRIX
##################################################################################
def build_estimate_covariance_matrix(portfolio, window_size):
    return risk_models.sample_cov(portfolio, frequency=window_size).to_numpy()

##################################################################################
# COMPUTE INVERSE ESTIMATE COVARIANCE MATRIX
##################################################################################
def compute_normalized_inverse_sigma(sigma, rho):
    inverse_sigma = np.linalg.inv(sigma)
    eigen_vals, eigenvectors = np.linalg.eigh(inverse_sigma)

    normalized_eigen_list = []
    # Normalize eigenvalues
    for i in range(len(eigen_vals)):
        eigen_vals[i] = eigen_vals[i] + rho

    #print(rho)
    # Rebuild with spectral decomposition
    Lambda = np.diag(eigen_vals)
    normalized_inverse_sigma = np.dot(np.dot(eigenvectors, Lambda), eigenvectors.transpose())

    return normalized_inverse_sigma

##################################################################################
# COMPUTE RHO
##################################################################################
def loss_function(rho, sigma_train, sigma_valid):
    inverse_sigma_train = compute_normalized_inverse_sigma(sigma_train, rho)
    w_train = global_min_var_portfolio(inverse_sigma_train)
    return np.dot(np.dot(w_train, sigma_valid), w_train.T)

def inequality_constraint(rho):
    return rho

def compute_rho(sigma_train, sigma_valid):
    initial_guess = 10
    constraints = ({'type': 'ineq', 'fun': inequality_constraint})
    result = minimize(loss_function, initial_guess, args=(sigma_train, sigma_valid),
    method='COBYLA', constraints=constraints)
    return result.x

####################################################################################
# FIND GLOBAL MINIMUM VARIANCE PORTFOLIO
####################################################################################
def global_min_var_portfolio(inverse_sigma):
    dim = len(inverse_sigma)
    ones = np.ones(dim)

    return (np.dot(inverse_sigma, ones))/(np.dot(np.dot(ones.T, inverse_sigma), ones))



####################################################################################
# RUN SIMULATION EXAMPLE
####################################################################################
def run_example():
    portfolio = get_stock_price(csv_path)

    # Print 90 day price chart
    portfolio.iloc[:90].plot(figsize=(6,4))
    plt.xlabel("Time (Days)")
    plt.ylabel("Open Price")
    plt.title("Prices Over Time for a Random Selection of 10 S&P 500 Stocks")

    # Get expected returns
    #expected_return_vector = expected_portfolio_return(0, 89, portfolio.iloc[:90])
    mu = expected_returns.mean_historical_return(portfolio.iloc[:90], frequency=90,
    compounding=False).to_numpy()
    mu_test = expected_returns.mean_historical_return(portfolio.iloc[90:120],
    frequency=90, compounding=False).to_numpy()

    # estimate covariance matrix
    estimate_covariance = build_estimate_covariance_matrix(portfolio.iloc[:90], 90)
    covariance_test = build_estimate_covariance_matrix(portfolio.iloc[90:120], 90)

    print(risk_models.sample_cov(portfolio, frequency=90))
    print(expected_returns.mean_historical_return(portfolio.iloc[:90], frequency=90,
    compounding=False))

    #compute rho
    rho_estimate_covariance = build_estimate_covariance_matrix(portfolio.iloc[:60], 90)
    rho_covariance_test = build_estimate_covariance_matrix(portfolio.iloc[60:90], 90)
    rho = compute_rho(rho_estimate_covariance, rho_covariance_test)
    print(rho)

    # normalized inverse estimate covariance matrix
```

```
normalized_inverse_sigma = compute_normalized_inverse_sigma(estimate_covariance, rho)


# Compute minimum variance portfolio
MVP = global_min_var_portfolio(np.linalg.inv(estimate_covariance))
MVP_Improved = global_min_var_portfolio(normalized_inverse_sigma)
Control = np.ones(10)*0.1

print("\n")
print("Portfolio Weights:")
print(MVP)
print(MVP_Improved)
print(Control)
print("\n")

MVP_return = np.dot(MVP, mu_test)
MVP_Improved_return = np.dot(MVP_Improved, mu_test)
Control_return = np.dot(Control, mu_test)

print("Return Classical: ", MVP_return)
print("Return Improved: ", MVP_Improved_return)
print("Return Control: ", Control_return)

print("\n")


MVP_cov = np.dot(np.dot(MVP.T, covariance_test), MVP)
MVP_Improved_cov = np.dot(np.dot(MVP_Improved.T, covariance_test), MVP_Improved)
Control_cov = np.dot(np.dot(Control.T, covariance_test), Control)

print("Variance Classical: ", MVP_cov)
print("Variance Improved: ", MVP_Improved_cov)
print("Variance Control: ", Control_cov)

print("\n")

MVP_sharpe = MVP_return/math.sqrt(MVP_cov)
MVP_Improved_sharpe = MVP_Improved_return/math.sqrt(MVP_Improved_cov)
Control_sharpe = Control_return/math.sqrt(Control_cov)

print("Sharpe Ratio Classical: ", MVP_sharpe)
print("Sharpe Ratio Improved: ", MVP_Improved_sharpe)
print("Sharpe Ratio Control: ", Control_sharpe)

x = [MVP_return, MVP_Improved_return, Control_return]
y = [MVP_cov, MVP_Improved_cov, Control_cov]
labels = ["Classical", "Improved", "Control"]
```

```
    fig, ax = plt.subplots()
    ax.scatter(x, y)
    plt.xlabel("Rate of Return")
    plt.ylabel("Covariance of Portfolio")
    plt.title("Rate of Return and Covariance of Various Portfolios")

    for i, txt in enumerate(labels):
        ax.annotate(txt, (x[i], y[i]))

    ax.set_ylim(bottom=0)
    ax.set_xlim(left=0)
    ax.set_ylim(top=0.01)
    ax.set_xlim(right=0.25)

    plt.show()



run_example()

####################################################################################
# RUN SIMULATION
####################################################################################
def run_simulation():

    num_rolls = 0
    while(num_rolls == 0):
        portfolio = get_stock_price(csv_path)
        num_rolls = int((len(portfolio)-180)/30)

    mu_classic = 0
    mu_improved = 0
    sigma_classic = 0
    sigma_improved = 0
    mu_control = 0
    sigma_control = 0

    for i in range(num_rolls):
        start = i*30
        end_train = start+90
        end_test = end_train+90

        mu = expected_returns.mean_historical_return(portfolio.iloc[start:end_train],
        frequency=90, compounding=False).to_numpy()
        mu_test = expected_returns.mean_historical_return(portfolio.iloc[end_train:end_test],
        frequency=90, compounding=False).to_numpy()
        # estimate covariance matrix
        estimate_covariance =
        build_estimate_covariance_matrix(portfolio.iloc[start:end_train], 90)
        covariance_test =
        build_estimate_covariance_matrix(portfolio.iloc[end_train:end_test], 90)


        # compute rho
        rho_estimate_covariance =
        build_estimate_covariance_matrix(portfolio.iloc[start:end_train-30], 90)
        rho_covariance_test =
        build_estimate_covariance_matrix(portfolio.iloc[end_train-30:end_test-30], 90)
        rho = compute_rho(rho_estimate_covariance, rho_covariance_test)

        # normalized inverse estimate covariance matrix
        normalized_inverse_sigma = compute_normalized_inverse_sigma(estimate_covariance, rho)

        MVP = global_min_var_portfolio(np.linalg.inv(estimate_covariance))
        MVP_Improved = global_min_var_portfolio(normalized_inverse_sigma)
        Control = np.ones(10)*0.1

        mu_classic += np.dot(MVP, mu_test)
        mu_improved += np.dot(MVP_Improved, mu_test)
        mu_control += np.dot(Control, mu_test)

        sigma_classic += np.dot(np.dot(MVP.T, covariance_test), MVP)
        sigma_improved += np.dot(np.dot(MVP_Improved.T, covariance_test), MVP_Improved)
        sigma_control += np.dot(np.dot(Control.T, covariance_test), Control)


    mu_classic /= num_rolls
    mu_improved /= num_rolls
    sigma_classic /= num_rolls
    sigma_improved /= num_rolls
    mu_control /= num_rolls
    sigma_control /= num_rolls

    return mu_classic, mu_improved, mu_control, sigma_classic, sigma_improved, sigma_control

mu_classic_f = []
mu_improved_f = []
mu_control_f = []

sigma_classic_f = []
sigma_improved_f = []
sigma_control_f = []
```

```
sharpe_classic_f = []
sharpe_control_f = []

for i in range(100):
    mu_classic, mu_improved, mu_control, sigma_classic, sigma_improved,
    sigma_control = run_simulation()

    mu_classic_f.append(mu_classic)
    mu_improved_f.append(mu_improved)
    mu_control_f.append(mu_control)

    sigma_classic_f.append(sigma_classic)
    sigma_improved_f.append(sigma_improved)
    sigma_control_f.append(sigma_control)

    sharpe_classic_f.append(mu_classic/math.sqrt(sigma_classic))
    sharpe_control_f.append(mu_improved/math.sqrt(sigma_control))

mu_classic_f = np.array(mu_classic_f)
mu_improved_f = np.array(mu_improved_f)
mu_control_f = np.array(mu_control_f)

sigma_classic_f = np.array(sigma_classic_f)
sigma_improved_f = np.array(sigma_improved_f)
sigma_control_f = np.array(sigma_control_f)

sharpe_classic_f = np.array(sharpe_classic_f)
sharpe_control_f = np.array(sharpe_control_f)

print(np.average(mu_classic_f))
print(np.average(mu_improved_f))
print(np.average(mu_control_f))
print("\n")
print(np.average(sigma_classic_f))
print(np.average(sigma_improved_f))
print(np.average(sigma_control_f))
print("\n")

sharpe_classical = np.average(sharpe_classic_f)
sharpe_improved = np.average(mu_improved_f)/math.sqrt(np.average(sigma_improved_f))
sharpe_control = np.average(sharpe_control_f)
print(sharpe_classical)
print(sharpe_improved)
print(sharpe_control)
print("\n")
```

```
std_dev_classical_sharpe = np.std(sharpe_classic_f)
std_dev_control_sharpe = np.std(sharpe_control_f)

# COMPARE IMPROVED TO CLASSICAL
z_ci = (sharpe_improved - sharpe_classical)/(std_dev_classical_sharpe/math.sqrt(100))
p_ci = 1 - st.norm.cdf(z_ci)
print("P-Value of Improved and Classical", p_ci)

# COMPARE IMPROVED TO CONTROL
z_oi = (sharpe_improved - sharpe_control)/(std_dev_control_sharpe/math.sqrt(100))
p_oi = 1 - st.norm.cdf(z_oi)
print("P-Value of Improved and CONTROL", p_oi)

# COMPARE CLASSICAL TO CONTROL
z_co = (sharpe_classical - sharpe_control)/(sharpe_control/math.sqrt(100))
p_co = 1 - st.norm.cdf(z_co)
print("P-Value of CLASSICAL and CONTROL", p_co)
```