Financial Math Mini-Project (Part 2)

Ahmer Nadeem Khan

29 March 2025

1 Introduction

In this part of the mini-project, we want to take values of a representative set of stocks in the US Stock Market over a short past period and compute the Global Minimum Variance Portfolio given by this historical data and the results of Portfolio theory. We then want to verify theoretical predictions about the portfolio and discuss briefly some of its properties. We extend the dataset from Part 1, and repeat our analysis, discussing the necessary theoretical and practical changes, and the results.

2 Objectives and Methodology

Note that this section is very similar to Part 1, and Part 1 can be consulted for details. Our objectives changes are the following:

1. We consider the top 400 stocks (by weight) in the index S&P500.
2. We obtain weekly adjusted closing prices of each stock for the past 27 weeks, starting from the current date. This means getting the adjusted closing prices starting on 03/28/2025 ("Today") and retrieving data for the preceding 26 weeks. Note that "Today" corresponds to market close on last Friday. Then, close prices are obtained for the required Fridays before. We consider adjusted close prices because these account for dividend payments, stock splits, and other
3. We then compute the weekly returns on the i-th stock by the formula
$$R^{(i)}_{weekly} = \frac{P^{(i)}_{t+1} - P^{(i)}_{t}}{P^{(i)}_{t}}.$$
4. We then obtain an estimate for the weekly risk-free return, and compute the excess returns on each stock in each period by the formula
$$R_{excess} = R_{total} - R_{risk-free}.$$
This can also be done after the computations; henceforth, return refers to excess return.
5. We compute the covariance matrix, the holdings vector for the Portfolio C (GMV Portfolio), the return, variance, and the standard deviation (risk) of the portfolio using the following formulas:
$$\mathbf{V} = \frac{(\mathbf{r}-\mathbf{f})(\mathbf{r}-\mathbf{f})^\top}{n} \qquad \Sigma = (\lambda^2 - l^2)\,\mathbf{h}\mathbf{h}^\top + \left(\frac{n}{p}\right) l^2\,\mathbf{I}$$
$$\mathbf{h}_c = \frac{\Sigma^{-1}\,\mathbf{e}}{\mathbf{e}^\top\,\Sigma^{-1}\,\mathbf{e}} \qquad f_c = \mathbf{h}_c^\top \mathbf{f} \qquad \sigma_c^2 = \mathbf{e}^\top\,\mathbf{V}\,\mathbf{e}$$
where $\mathbf{e} = \mathbf{1}$ is a vector of ones, $\mathbf{V}$ is the covariance matrix, $\mathbf{h}_c$ is the holdings vector of the Portfolio C, $\mathbf{r}$ is the matrix of returns, σ_c^2 is the variance of the portfolio C, and the expected returns vector is

$\mathbf{f} = \mathbb{E}[\mathbf{r}]$.
Σ is an estimator for the covariance matrix, where λ^2 is the leading eigenvalue of V, $\mathbf{h}$ is the corresponding unit eigenvector, $\mathbf{I}$ is the identity matrix of size n, and l^2 is defined as

$$l^2 = \frac{\mathbf{tr}(V) - \lambda^2}{n-1}$$

where $\mathbf{tr}(V)$ is the trace of V. The need for this estimator will be covered later.

6. We then perform the same computations (with some contingencies) and present and compare our results and verify properties of the portfolio C.

Please note the following observations:

- We have $p = 400$, the number of assets i.e. 400 stocks.
- We have $n = 26$, the number of time periods over which we have returns, i.e. 26 weeks. The data for closing prices has to contain 27 data points so that the returns can have 26 data points (1 less, by construction).
- The size of the matrix of returns, $\mathbf{r}$, is $p \times n$, where each row corresponds to the returns of the i-th asset over the specified period. For our case this becomes 400×26.
- There are many ways to obtain an estimate for the weekly risk-free return.
- $\mathbf{f} = \mathbb{E}[\mathbf{r}]$ is computed by averaging across each row to get an *estimate* of the expected (weekly) return of the particular stock. We choose a simple arithmetic mean as our expected return.

2.1 Data Collection and Implementation

We need to collect the following data:

- Constituents of the S&P500
- Stock Market price data
- Risk-free rate

Here is how we obtain and pre-process this data:

- The constituents stocks are available in order of cap-weight in the index on Yahoo Finance or various websites. [2] We use the *Slick Charts* website due to the ease of data scraping off of the basic HTML, compared to JavaScript on *Yahoo Finance*. We can also retrieve the constituent stocks of the S&P500 from Wikipedia [1], which is the most reliable and regularly updated source, but this is an unordered list. We can compare out ordered list with the unordered list to check if our sources are reliable.
- The adjusted daily close price data was scraped off of Yahoo Finance historical data web pages (tables) for each stock, for the maximum of two years (500 days) from the following date range: 2023-03-31 to 2025-03-28. An example of the table is Apple Inc., the largest stock in the index currently, and the URL is in the bibliography [4]. The weekly close were then sliced from this data very easily. The daily data will act as the raw data, and also help us compute for finer (larger) values of n. This will become relevant later.
- To obtain the appropriate estimate of the risk-free return, we download the .csv files provided by the US Department of the Treasury website [3], and scale to get the weekly return on the T-bill from the annual return using the formula:

$$r_{weekly} = (1 + return_{annual})^{1/52} - 1$$

which is obvious from properties of compounding.

- We present annual returns (compounded) and variances wherever appropriate, and use the following formulas to convert from weekly:

$$r_{annual} = (1 + return_{weekly})^{52} - 1$$

$$\sigma^2_{annual} = 52 \times \sigma^2_{weekly}$$

$$\sigma_{annual} = \sqrt{52} \times \sigma_{weekly}$$

The data is available as comma-separated files (.csv format) and can be downloaded.

We implement our computation in Python using a Jupyter Notebook (.ipynb file) and produce the computations and outputs in different blocks of code for easier reproducibility. We use the basic libraries *pandas* for handling and extracting data, and *numpy* for computations on arrays. For simple mathematical computations we use the library *math*, and *matplotlib* for producing graphs/tables when necessary. For some plotting and latex code generation, I used generative AI (specifically ChatGPT 4-o). The files for scraping the datasets were written as separate .py scripts, and are also available. I will write a separate document explaining the data scraping process.

2.2 Covariance Estimator Matrix Σ

For our setting, our returns matrix $\mathbf{r}$ has size 400×26 where $p = 400$ is the number of assets and $n = 26$ is how often we sample the historical returns (weekly, for 26 weeks). This means when we form the covariance matrix

$$\mathbf{V} = \frac{(\mathbf{r} - \mathbf{f})(\mathbf{r} - \mathbf{f})^\top}{n}$$

where we have de-meaned the returns matrix by subtracting from $\mathbf{r}$ the row-wise means, from each value row-wise. The resulting matrix has size 400×400, but since $p > n$ in this case, the covariance matrix is rank-deficient, i.e it has rank at max 26 $\implies$ $\mathbf{rank}(\mathbf{V}) \leq 26$. In particular this means that $\mathbf{V}$ is not invertible; which is why our earlier expressions to compute the holdings vectors and associated quantities cannot be used. We instead estimate the covariance matrix with

$$\Sigma = (\lambda^2 - l^2)\,\mathbf{h}\mathbf{h}^\top + \left(\frac{n}{p}\right) l^2\,\mathbf{I}$$

where

- λ^2 is the leading eigenvalue of $\mathbf{V}$.
- $\mathbf{h}$ is the corresponding leading eigenvector of V.
- l^2 is defined as

  $$l^2 = \frac{\mathbf{tr}(V) - \lambda^2}{n - 1}$$

  This means that l^2 is the average of the eigenvalues of $\mathbf{V}$ smaller than λ^2 (since it is the leading eigenvalue). This is because the trace of a matrix is the sum of its eigenvalues.

Note that $\mathbf{V}$ is a real symmetric positive semi-definite matrix, so it has all real eigenvalues. Our estimator approximates the effect of $\mathbf{V}$ in the most dominant direction by choosing the leading eigenvector, and adding to the diagonal a scaled average of the other eigenvalues. This comes from the single-factor market model, and is called the **Ledoit-Wolf Single-Factor Shrinkage Estimator**. We will prove certain results about out estimator matrix. We first show that it is symmetric positive definite, and therefore, invertible.

Proposition 2.1. Σ *is a spd matrix.*

Proof. Note first that $\Sigma^\top = (\lambda^2 - l^2)\,(\mathbf{h}\mathbf{h}^\top)^\top + \left(\frac{n}{p}\right) l^2\,\mathbf{I}^\top = (\lambda^2 - l^2)\,\mathbf{h}\mathbf{h}^\top + \left(\frac{n}{p}\right) l^2\,\mathbf{I} = \Sigma$ using basic properties of transpose, meaning Σ is symmetric. Then let $\mathbf{x} \neq \mathbf{0}$, and consider

$$\begin{aligned}\mathbf{x}^\top \Sigma \mathbf{x} &= (\lambda^2 - l^2)\,\mathbf{x}^\top \mathbf{h}\mathbf{h}^\top \mathbf{x} + \left(\frac{n}{p}\right) l^2\,\mathbf{x}^\top \mathbf{I}\mathbf{x} \\ &= (\lambda^2 - l^2)\,\langle \mathbf{h}, \mathbf{x}\rangle^2 + \left(\frac{n}{p}\right) l^2\,||\mathbf{x}||^2 > 0\end{aligned}$$

from the following inequalities: $\langle \mathbf{h}, \mathbf{x}\rangle^2 \geq 0$, $||\mathbf{x}||^2 > 0$ for all $\mathbf{x} \neq 0$. If we reasonably assume at least two positive eigenvalues for $\mathbf{V}$ (since V is positive semi-definite it has non-negative eigenvalues), then we also get that $(\lambda^2 - l^2) > 0$ and $l^2 > 0$ from the definitions of λ^2 and l^2. Thus Σ is spd. □

Next we show that the $\mathbf{h}$ is a leading eigenvector of Σ

Proposition 2.2. ***h*** *is the the leading eigenvector of* Σ.

Proof. Consider the expression

$$\Sigma \mathbf{v} = (\lambda^2 - l^2)\,\mathbf{h}\mathbf{h}^\top \mathbf{v} + \left(\frac{n}{p}\right) l^2\,\mathbf{I}\mathbf{v}$$

If $\mathbf{v} = \mathbf{h}$, we get the expression

$$\begin{aligned}\Sigma \mathbf{h} &= (\lambda^2 - l^2)\,\mathbf{h}(\mathbf{h}^\top \mathbf{h}) + \left(\frac{n}{p}\right) l^2\,\mathbf{h} \\ &= (\lambda^2 - l^2)\,\mathbf{h} + \left(\frac{n}{p}\right) l^2\,\mathbf{h} \\ &= \left((\lambda^2 - l^2) + \left(\frac{n}{p}\right) l^2\right) \mathbf{h}\end{aligned}$$

since $\mathbf{h}^\top \mathbf{h} = 1$ as $\mathbf{h}$ is the unit eigenvector. This shows that $\mathbf{h}$ is a unit eigenvector of Σ, with the above eigenvalue. To prove that this is the leading eigenpair, we show that all other eigenvalues are smaller. Assume $\mathbf{v}$ is a different eigenvector. WLOG, we can assume that $\mathbf{v}$ is orthogonal to $\mathbf{h}$ since we can find the spectral decomposition of an spd matrix (from the Real Spectral Theorem. Then the first part of the sum becomes 0, and we are left with

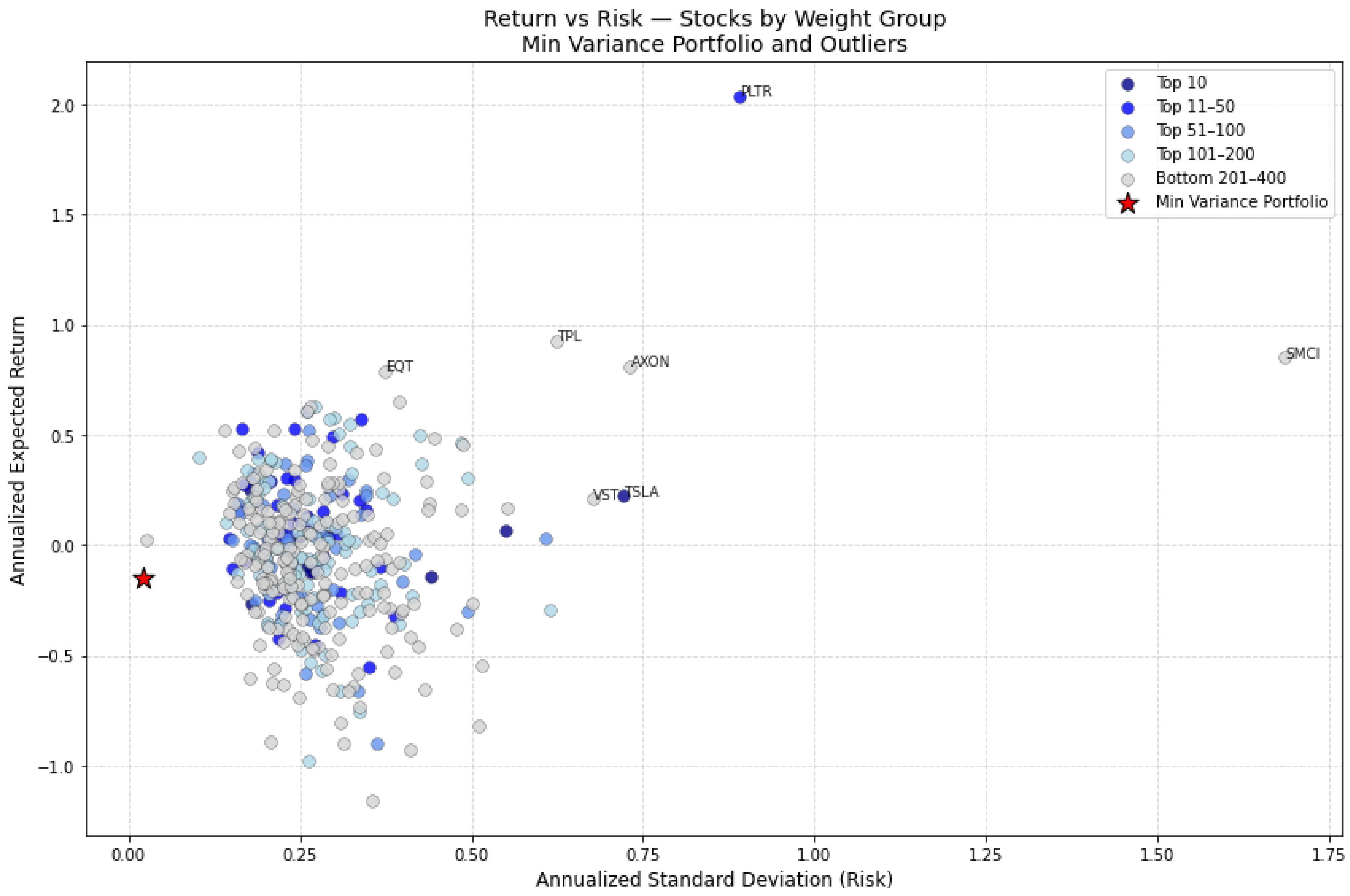
$$\Sigma \mathbf{v} = \left(\frac{nl^2}{p}\right) \mathbf{v}$$

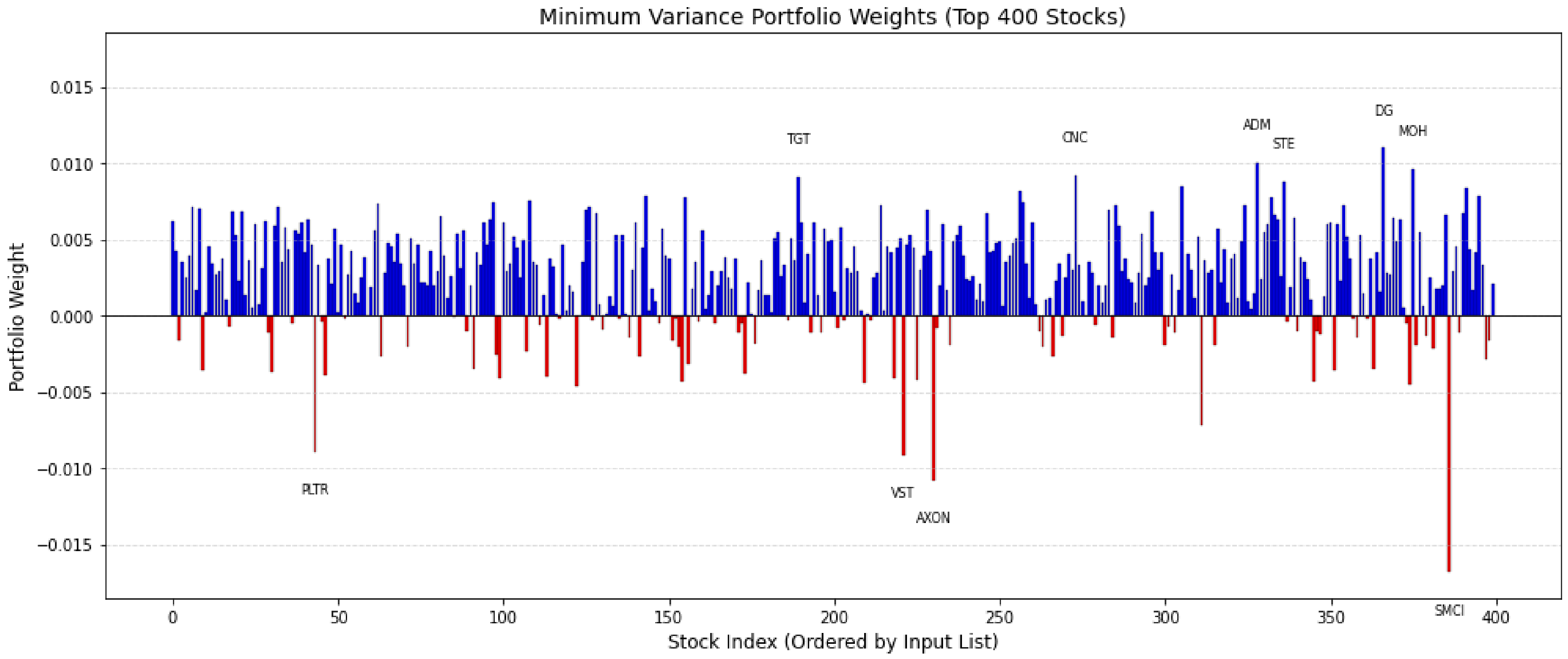
for all other eigenvectors (note all of them have the same eigenvalue - the scaled average of all eigenvalues of $\mathbf{V}$ smaller than λ^2). Clearly, this value is smaller since it is less the positive value $(\lambda^2 - l^2)$. This completes the proof. □

3 Results and Conclusions

I will first simply display the required values and variables obtained from the code. All quantities are annualized. I used graphs and charts instead of vector lists, wherever appropriate.

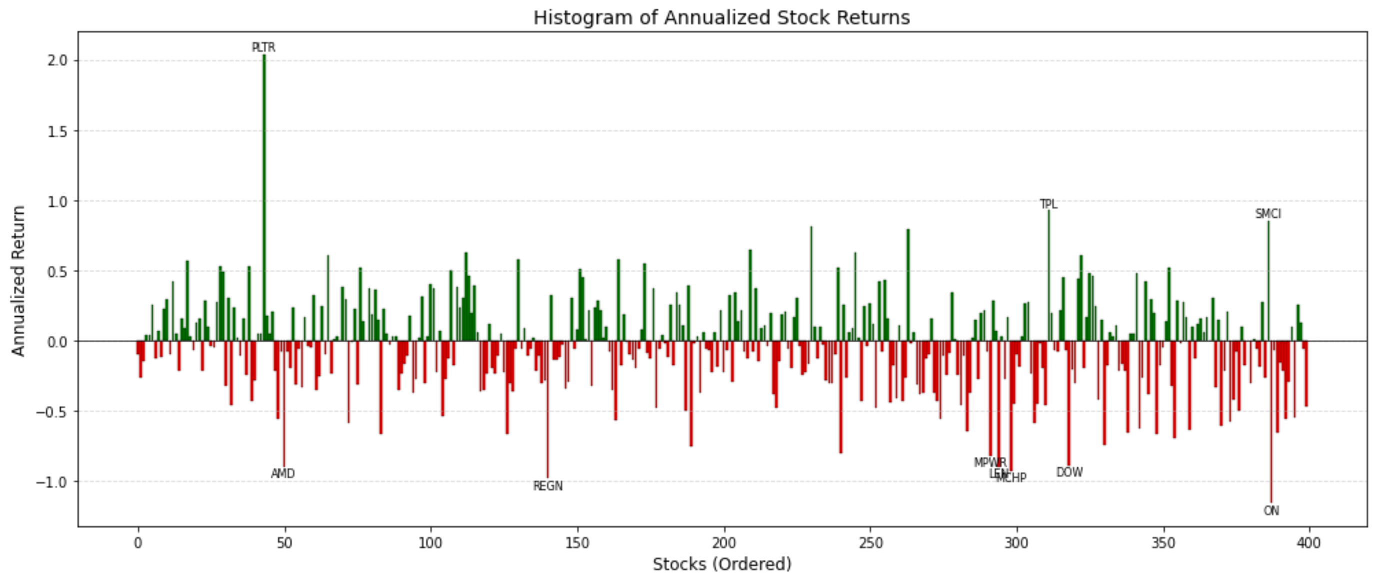
Metric	Value
Minimum Variance Portfolio Return	-0.1468
Minimum Variance Portfolio Variance	0.0005
Minimum Variance Portfolio Std Dev	0.0224
Sum of Portfolio Weights	1.0000
Max Individual Stock Return	2.0382
Min Individual Stock Return	-1.1562
Max Individual Stock Variance	2.8417
Min Individual Stock Variance	0.0007

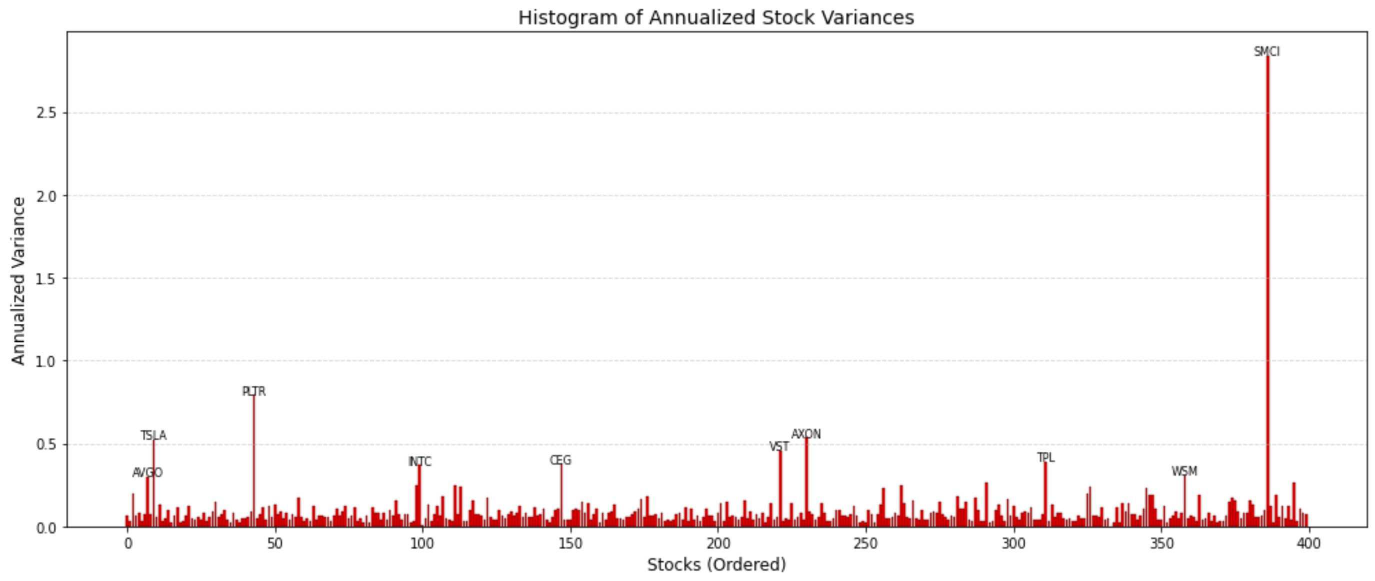

Return vs Risk — Stocks by Weight Group
Min Variance Portfolio and Outliers
Annualized Expected Return
Annualized Standard Deviation (Risk)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
PLTR
TPL
AXON
SMCI
EQT
VST
TSLA
Top 10
Top 11–50
Top 51–100
Top 101–200
Bottom 201–400
Min Variance Portfolio


Minimum Variance Portfolio Weights (Top 400 Stocks)
Portfolio Weight
0.015
0.010
0.005
0.000
−0.005
−0.010
−0.015
Stock Index (Ordered by Input List)
0
50
100
150
200
250
300
350
400
TGT
CNC
ADM
STE
DG
MOH
PLTR
VST
AXON
SMCI

The first graph is the risk-return diagram of the individual stocks and the GMV portfolio. The second graph shows the distribution of long (positive) and short (negative) positions in the portfolio. Outliers on both graphs are also labeled. Note that the order of all stocks (x-axes) in these and the following graphs is the original cap-weight order - this makes the data more meaningful.

We can also now plot the returns and variances of the individual assets. Again, a histogram is the best choice for this visualization. Vectors can be accessed from the code!





Conclusions: We can first easily verify that the portfolio is fully invested by noticing that the sum equals 1 (this is done in the code and displayed in the table). We also have both long and short positions in the portfolio. From the table and the graph, we can immediately deduce that C does indeed have the smallest risk, at least when compared individually to the market (top 400 stocks). If any stock had a smaller variance, we would be completely invested in that one stock - as it turns out, we can minimize variance by a combination of stocks. In terms of minimizing risk, this is the optimal fully invested portfolio. The distribution of long and short positions is also visualized.

The short positions in volatile stocks like Palantir and Super Micro Computer Inc. indicate the short horizon of our historical data, since these may be imprudent investments given macroeconomic trends or fundamental analyses - the true, overall volatility of these stocks is not priced in. SMCI and Palantir stocks are both heavily affected by trends in AI (e.g. DeepSeek). It also appears from the expected negative return that we are capturing a overall downturn in the market, at least in these major stocks. If the GMV portfolio actually had negative expected (excess) return, it would not make sense to be invested in it; instead, you would fully invest in the risk-free asset to get guaranteed positive return (0 excess return) at 0 risk. On the diagram, this means moving towards the origin, we reduce risk and increase expected returns. However, theoretically, the properties of C are verified: it is indeed a low-risk portfolio to invest in.

We can now also investigate the risk-return preferences of the minimizer by the following diagrams:

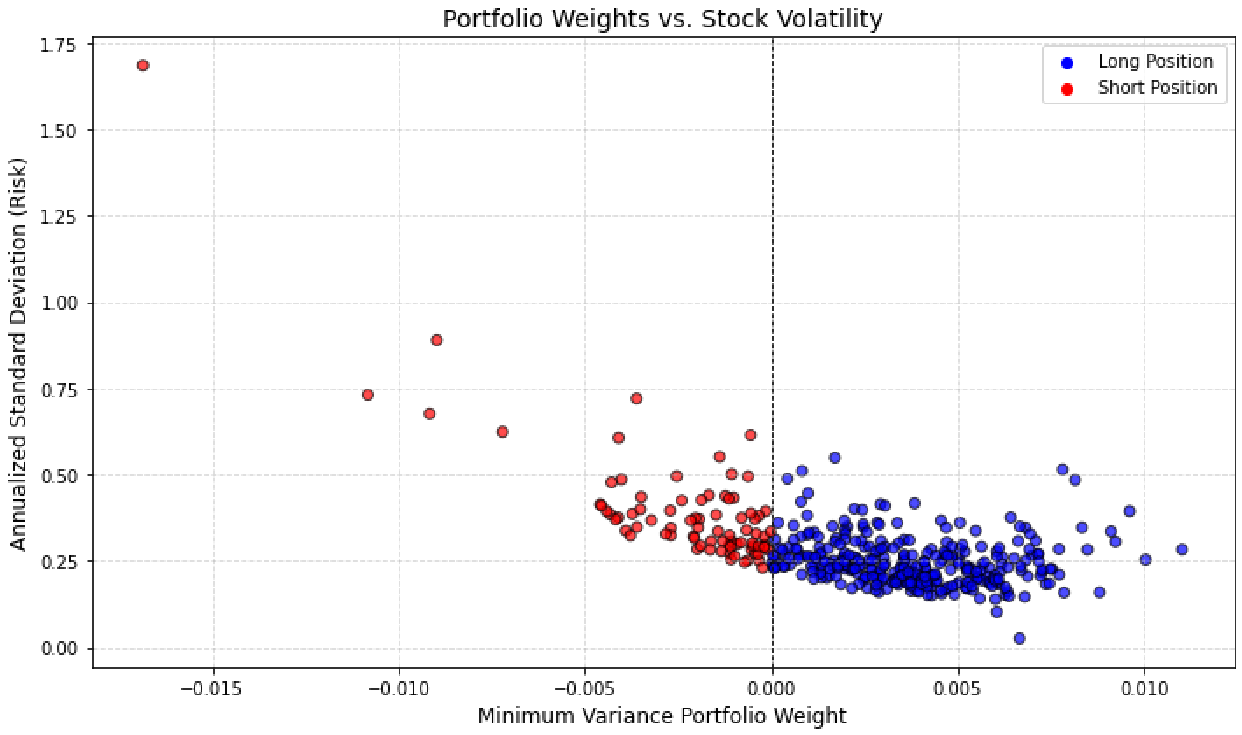


Figure 1: Risk Preferences

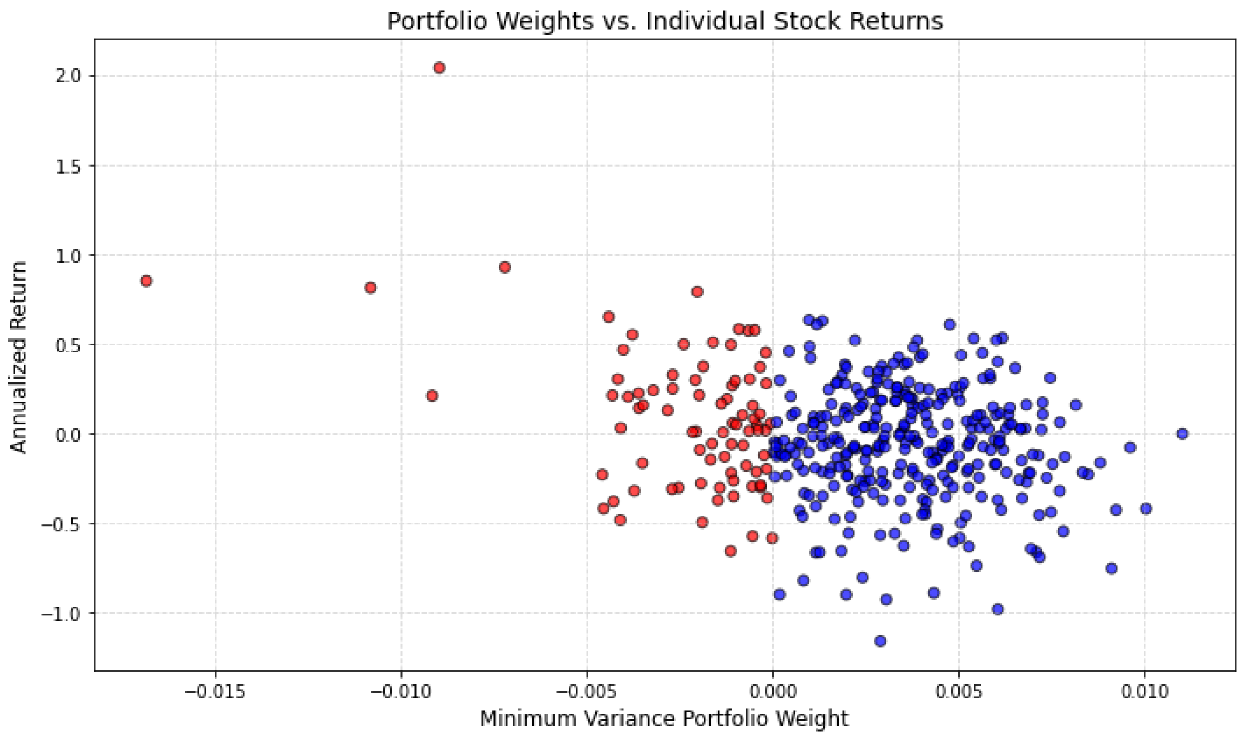
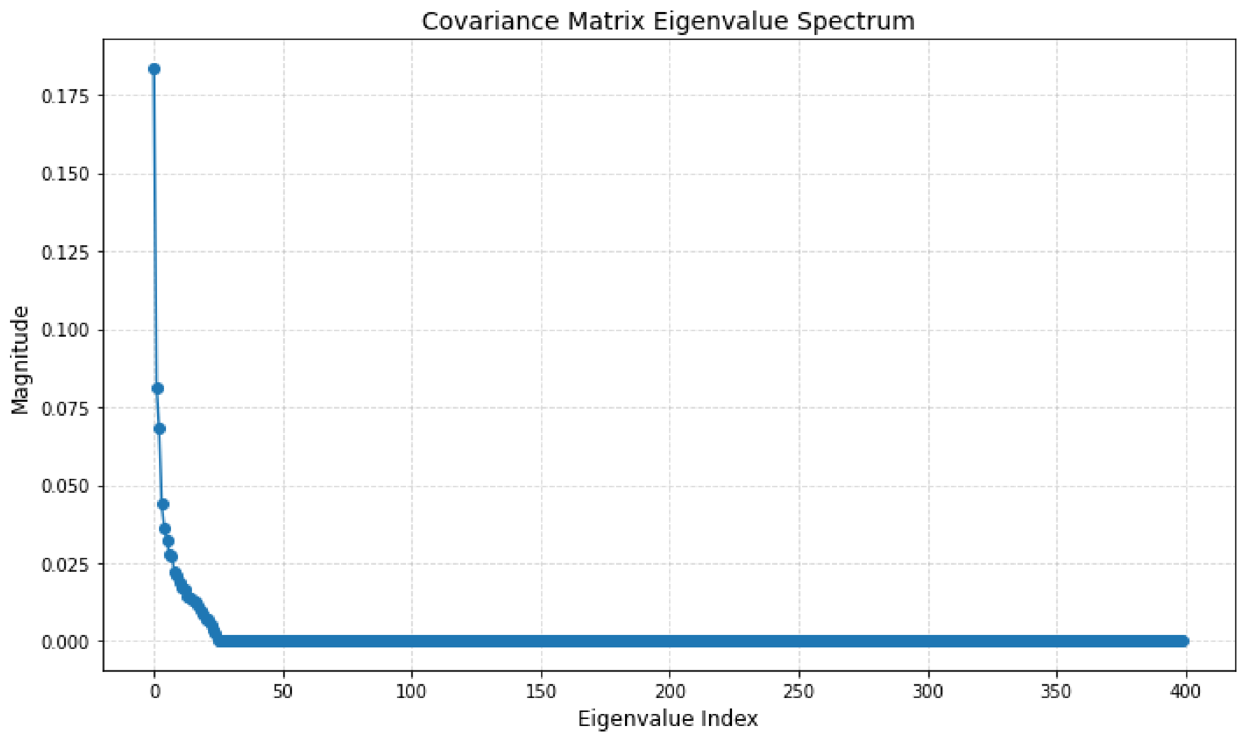


Figure 2: Return Preferences

Conclusions: We can notice that the minimizer generally prefers stocks with lower risk very tightly, but is much more ambivalent towards returns (still prefers lower returns, but the correlation is much weaker).

Finally, we can analyze and justify the use of the estimator covariance matrix by analyzing the eigenspectrum:



Conclusions: The steep decay in the eigenvalue magnitudes of the covariance matrix implies a very dominant direction, or exposure to risk from one or a very few factors. The largest exposure in equity markets is usually the entire market itself - thus the 'market' model. This means that our estimator matrix provides a very good estimate to the effect of the covariance matrix.

4 2 Year Horizon with Daily Data

Since we extracted daily data for two years, which is 500 values of close prices (*adj), we can in fact get a full rank returns matrix by using $n \geq 400$. In this case, we still use 400 stocks, but over 500 days. This allows us to directly use the covariance matrix in the formulas. However, note the following issues:

- Some stocks do not have data going back the full 500 days - e.g the stock KVUE had its Initial Public Offering (IPO) in May 2023, which is after the earliest date. Since the constituents of the S&P500 are changing and are effect to re-balancing every quarter, this is expected. I identified 5 stocks that do not have data going back 2 years, with Apple as reference:

Ticker	**Days**
SW	183
GEV	250
SOLV	250
VLTO	372
KVUE	477
AAPL	500

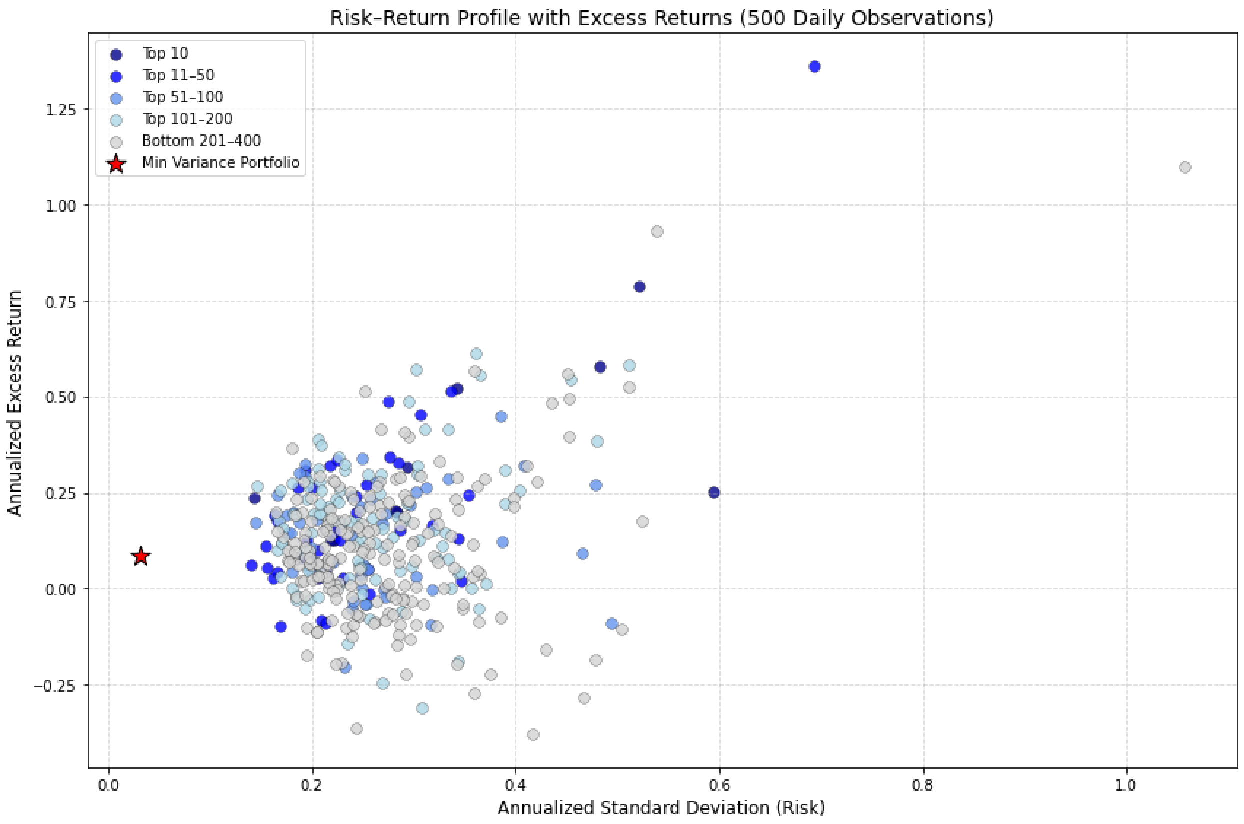
Table 1: Number of Valid Price Observations for Selected Stocks

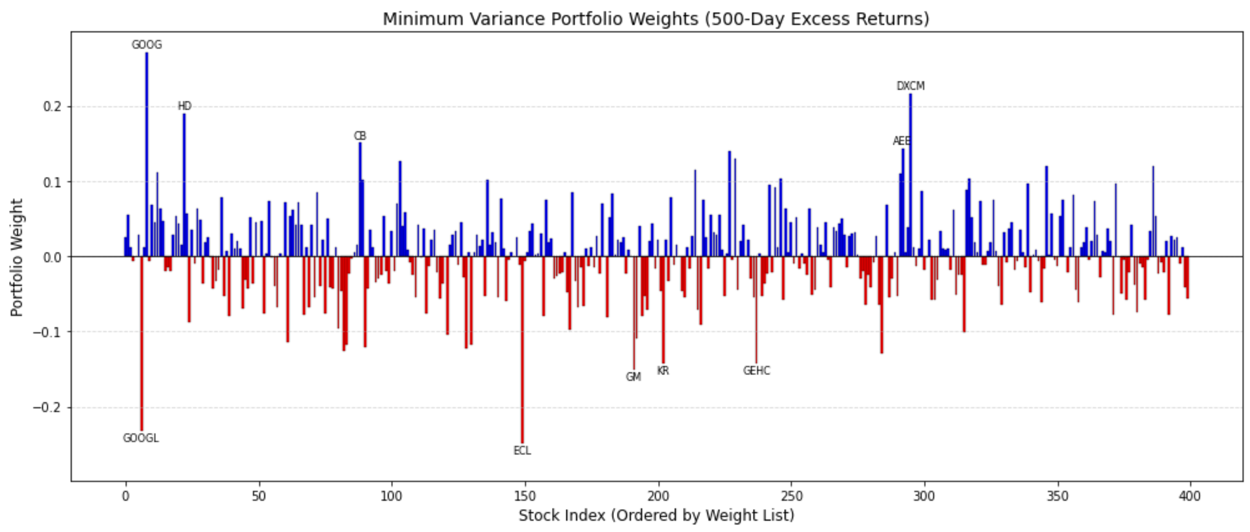
I removed these stocks first, and then picked the top 400 stocks after.

- Note for the 26-weeks horizon, this was not an issue, since we aggregate the returns into a mean vector. The NaN values in the left tails are simply ignored in the aggregation - this amounts to us having lesser samples, which is not a big issue, especially since for 26 weeks, only one stock (SW) is affected; all the other ones have enough data points.

We get the following results for this method:

Metric	Value
Minimum Variance Portfolio Return	0.084869
Minimum Variance Portfolio Variance	0.000950
Minimum Variance Portfolio Std Dev	0.030828
Sum of Portfolio Weights	1.000000
Max Individual Stock Return	1.359834
Min Individual Stock Return	-0.378068
Max Individual Stock Variance	1.116759
Min Individual Stock Variance	0.019707





Conclusions: Note the much more realistic results over this longer horizon - the GMV has much better risk than the market, while still has positive return, and better return than many stocks. We also take a lot more short positions, and also much larger exposures, which also makes sense, since there is no penalty to do so.

5 Code Appendix

Libraries

```
import numpy as np
import math
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Raw Data

```
df = pd.read_csv("sp500_adjusted_close_2y_full.csv", parse_dates=["Date"])
ordered_tickers = pd.read_csv("sp500_tickers.csv")['Symbol'].tolist()

df = df.pivot(index="Ticker", columns="Date", values="Adj Close")
df = df.loc[ordered_tickers]

print(df.head())

# Load T-bill Rates files
df_2024 = pd.read_csv("daily-treasury-rates-2024.csv")
df_2025 = pd.read_csv("daily-treasury-rates-2025.csv")

# Combine them
df_rates = pd.concat([df_2024, df_2025], ignore_index=True)

# Convert 'Date' column to datetime
df_rates['Date'] = pd.to_datetime(df_rates['Date'])

# Sort by date (just in case)
```

```
df_rates = df_rates.sort_values("Date").reset_index(drop=True)

# Preview
print(df_rates.head())
print("Shape:", df_rates.shape)
```

Pre-processing

```
non_null_counts = df.notna().sum(axis=1).sort_values()
print(non_null_counts.head(10))  # Tickers with the least data

# Choose the ticker you want to inspect
view_ticker = 'SW'

# Check if the ticker exists in the pivoted DataFrame
if view_ticker in df.index:
    print("\n",df.loc[view_ticker])
else:
    print(f"{view_ticker} not found in dataset.")

p = 400
df_top = df.iloc[:p]

print(df_top.shape)
print(df_top.index[:5])  # Preview first 5 tickers

# Step 1: Transpose (dates as index)
df_t = df_top.transpose()
df_t.index = pd.to_datetime(df_t.index)  # Ensure datetime index

# Step 2: Resample to weekly (Friday close)
df_weekly = df_t.resample("W-FRI").last()

n=26
n_dates = n+1
# Step 3: Get last 27 weeks
df_weekly_recent = df_weekly.tail(n_dates)

# Step 4: Transpose back (tickers as rows)
df_weekly_recent = df_weekly_recent.transpose()

# Done | preview
print("Shape:", df_weekly_recent.shape)        # (400, 27)
print("First few tickers:", df_weekly_recent.index[:5])
print("First few dates:", df_weekly_recent.columns[:5])
print("Last date:", df_weekly_recent.columns[-1])
```

Excess and De-meaned Returns

```
# Get weekly prices as NumPy array
prices = df_weekly_recent.values  # shape: (400, 27)

# Compute returns row-wise
returns = (prices[:, 1:] - prices[:, :-1]) / prices[:, :-1]  # shape: (400, 26)
```

```
# Preview
print("Returns shape:", returns.shape)
print("Sample (first ticker):", returns[0])

# Drop rows with missing values in the 26-week column
df_rates = df_rates.dropna(subset=["26 WEEKS COUPON EQUIVALENT"])

# Get the most recent available rate (last row)
latest_rate = df_rates["26 WEEKS COUPON EQUIVALENT"].iloc[-1]

print(latest_rate)

risk_free_weekly = latest_rate / 100 / 52  # convert to decimal and per week

excess_returns = returns - risk_free_weekly

# Step 1: Compute row-wise mean
row_means = excess_returns.mean(axis=1, keepdims=True)  # shape: (400, 1)

# Step 2: Subtract row mean from each row (using Broadcasting automatically does what we want)
excess_returns_demeaned = excess_returns - row_means

# Check result
print("Shape:", excess_returns_demeaned.shape)
print("First row mean (should be ~0):", excess_returns_demeaned[0].mean())
```

Covariance Matrix Y and Ledoit-Wolf Single-Factor Estimator

```
# excess_returns_demeaned shape: (400, 26)
cov_matrix = (excess_returns_demeaned @ excess_returns_demeaned.T) / n

print("Covariance matrix shape:", cov_matrix.shape)
print("Sample (top-left 5x5):")
print(cov_matrix[:5, :5])

eigvals, eigvecs = np.linalg.eigh(cov_matrix)  # symmetric eigendecomposition

# Step 1: Leading eigenvalue and eigenvector
lambda_sq = eigvals[-1]            # largest eigenvalue (lambda²)
h = eigvecs[:, -1]                 # unit eigenvector corresponding to lambda²

# Step 2: Compute l²
trace_S = np.trace(cov_matrix)
ell_sq = (trace_S - lambda_sq) / (n - 1)

# Step 3: Build the estimator
term1 = (lambda_sq - ell_sq) * np.outer(h, h)
term2 = (n / p) * ell_sq * np.eye(p)

Sigma = term1 + term2

# Preview
print("Estimated Sigma shape:", Sigma.shape)
print("l2 =", ell_sq)
print("lambda2 =", lambda_sq)
```

GMV Portfolio

```
# Vector of ones
e = np.ones((p, 1))

# Invert Sigma
Sigma_inv = np.linalg.inv(Sigma)

# Minimum variance weights
numerator = Sigma_inv @ e
denominator = e.T @ Sigma_inv @ e
h_c = numerator / denominator  # shape: (p, 1)

# Mean return vector from excess_returns
f = excess_returns.mean(axis=1, keepdims=True)  # shape: (p, 1)

# Weekly portfolio return and variance
f_c_weekly = float(h_c.T @ f)
sigma_c_sq_weekly = float(h_c.T @ Sigma @ h_c)

# Annualize
f_c_annual = f_c_weekly * 52
sigma_c_sq_annual = sigma_c_sq_weekly * 52
sigma_c_annual = np.sqrt(sigma_c_sq_annual)
```

Output

```
# Output
print("Annualized minimum variance portfolio return:", f_c_annual)
print("Annualized variance:", sigma_c_sq_annual)
print("Annualized standard deviation:", sigma_c_annual)
print("Sum of the holdings vector (Fully invested condition):", np.sum(h_c))
```

Note: Visualization code is available in the Jupyter notebook file *mini_project_2.ipynb*.

References

[1] Open Knowledge Foundation. S&p 500 companies dataset. `https://github.com/datasets/s-and-p-500-companies`, 2023. Data licensed under the Open Data Commons Public Domain Dedication and License. Code under MIT/BSD license.

[2] Slickcharts. S&P 500 Companies by Weight. Accessed: 2025-03-31.

[3] U.S. Department of the Treasury. Daily Treasury Bill Rates - 2025. `https://home.treasury.gov/resource-center/data-chart-center/interest-rates/TextView?type=daily_treasury_bill_rates&field_tdr_date_value=2025`, 2025. Accessed: 2025-03-31.

[4] Yahoo! Finance. Apple Inc. (AAPL) Stock Historical Data. `https://finance.yahoo.com/quote/AAPL/history/?interval=1wk&filter=history&frequency=1d&period1=1680220800&period2=1743379200`, 2025. Accessed: 2025-03-31.