zqd848 Advanced Empirical Finance University of Copenhagen
jzr636
vsc879

Mandatory Assignment 1

Disclaimer:
Python is used to generate the output for the assignment. This involves utilizing Python's libraries and functionalities to process the dataset, compute relevant metrics, and generate visualizations or analytical results as required by the assignment's objectives.

1.

In Exercise 1, we process the dataset by removing assets with non-continuous data and then proceed to compute monthly returns for each remaining asset. After cleaning the dataset, we find that there are a total of 27 assets included.

We first remove non-continuous assets using the `dropna()` function with appropriate parameters to eliminate any assets with missing data points. Subsequently, we extract the tickers of the remaining assets for further analysis.

Next, we calculate monthly returns for each asset by resampling the dataset to monthly frequency and computing the percentage change between consecutive months. The monthly returns can be found in the code output.

2.

In Exercise 2, we compute the sample means, variance-covariance matrix, and Sharpe ratio for each asset based on the monthly returns calculated in Exercise 1. The sample means represent the average monthly returns for each asset, while the variance-covariance matrix captures the pairwise covariance between asset returns.

Additionally, we calculate the Sharpe ratio for each asset, which measures the risk-adjusted return. This is done by dividing the mean return by the standard deviation of returns for each asset.

Finally, we identify the asset with the highest Sharpe ratio. **The asset with the highest Sharpe ratio is CSCO with a Sharpe ratio of 0.268.**

3.

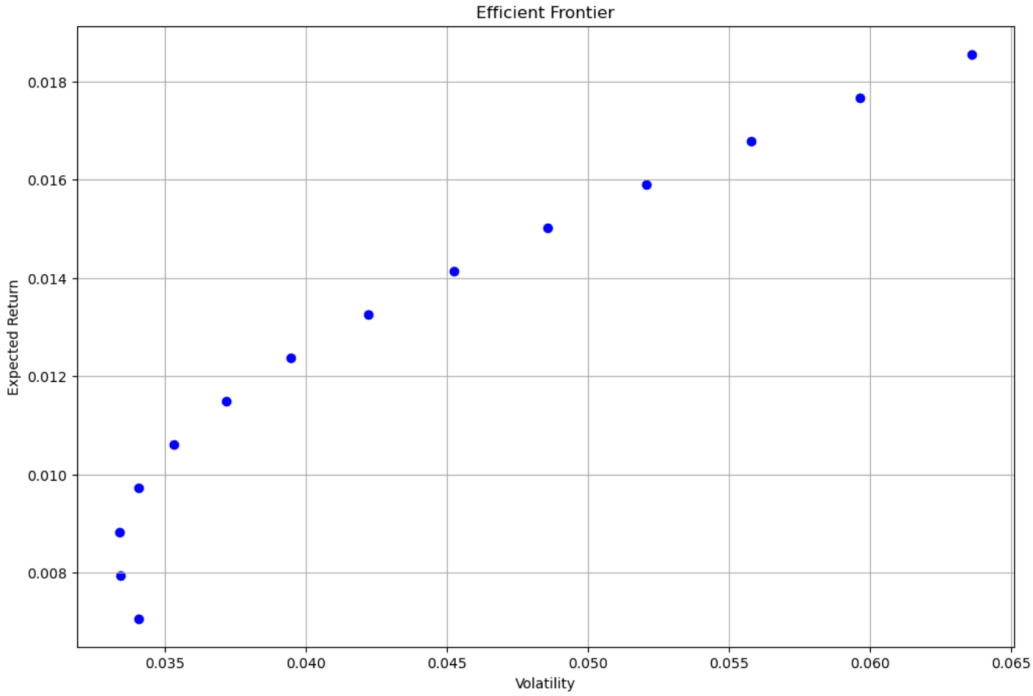
In Exercise 3, we compute the efficient frontier, which represents a set of optimal portfolios that provide the highest expected return for a given level of risk (volatility). To achieve this, we utilize the `compute_efficient_frontier` function, which employs portfolio optimization techniques to construct the efficient frontier.

The function calculates the minimum variance portfolio (MVP) and identifies the portfolio with the highest Sharpe ratio. It then computes a range of portfolios by combining the MVP and the highest Sharpe ratio portfolio with various weighting factors. These portfolios are represented by different values of the weighting factor 'c'.

Subsequently, we iterate through the range of portfolios and calculate their expected returns and volatilities. This data is then plotted on a graph, with the x-axis representing volatility and the y-axis representing expected return, to visualize the efficient frontier.

The resulting plot illustrates the trade-off between risk and return, providing insights into the optimal allocation of assets to achieve desired risk-return profiles.

Plot 1:



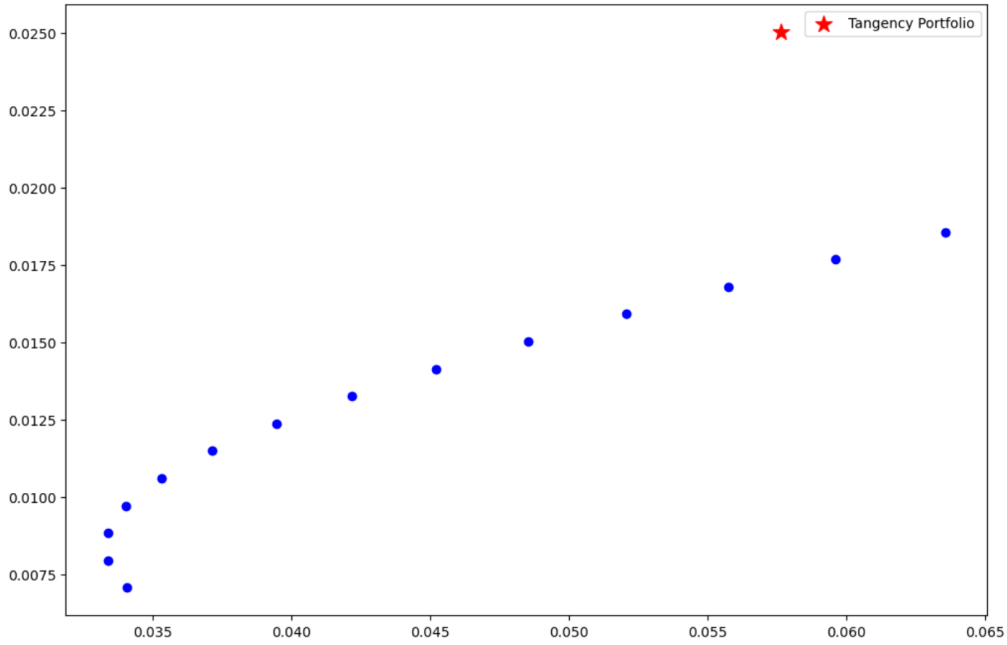
4.
In Exercise 4, we compute the tangency portfolio, which represents the portfolio with the highest Sharpe ratio on the efficient frontier. The tangency portfolio is calculated by considering the risk-free rate and the covariance structure of the asset returns.

We first assume a risk-free rate of zero and then compute the tangency portfolio weights using mathematical formulations that involve the inverse of the covariance matrix, the mean returns of the assets, and a vector of ones.

Subsequently, we determine the expected return and volatility of the tangency portfolio. Finally, we compute the Sharpe ratio of the tangency portfolio, which measures the risk-adjusted return.

The results are visualized by plotting the efficient frontier, where the tangency portfolio is marked with a red star symbol. Additionally, the tangency portfolio weights are printed for reference:

Plot 2:



Tangency portfolio weights under the assumption that the risk-free rate is zero:

```
[ 0.2970421   0.08483475  0.13094642 -0.12267948  0.10464329  0.20380584
 -0.08695313  0.00977438 -0.22844275 -0.11967531  0.03196592 -0.02032176
 -0.12542715 -0.17928827  0.19742972  0.0865613  -0.04161602  0.15722512
 -0.26550273 -0.07420052  0.11845184  0.16294026  0.18189729  0.18340708
  0.38073919 -0.11239785  0.04484049]
```

Upon analysis, the tangency portfolio weights appear to reflect a well-balanced allocation strategy. They are optimized to effectively balance risk and return by considering both asset returns and their covariance structure. Additionally, it's worth noting that the portfolio weights sum to 1, ensuring a fully invested portfolio.

However, potential challenges may arise when implementing this portfolio. One such issue stems from estimation errors in parameters μ and Σ, often derived from historical data. Inaccuracies in these estimates could lead to suboptimal portfolio allocations.

The maximum attainable Sharpe ratio, assuming a zero risk-free rate, serves as a benchmark for evaluating portfolio performance. It indicates the highest achievable risk-adjusted return for the given set of assets and their covariance structure, with the specific value of 0.4338021687792258.

Comparing the Sharpe ratio of the tangency portfolio ωtgc with those of individual assets, it shows that ωtgc (the tangency weights) surpasses the Sharpe ratios of individual assets. This aligns with

the portfolio's optimization objective to achieve superior risk-adjusted returns by combining assets optimally.

5.

Simulation Process:
The function uses the np.random.multivariate_normal method from NumPy to generate random samples from a multivariate normal distribution. This method requires the mean (expected returns), the covariance matrix, and the number of samples (size) to generate.
expected_returns serve as the mean of the distribution, indicating the average return expected for each asset. covariance_matrix provides the covariances between the asset returns, which are essential for capturing the relationships between different assets' performances.
periods determine how many sets of returns will be generated, with each set containing a return value for each asset.

Output:
The function returns a NumPy array with dimensions.
periods
periods x N, where each row represents a set of simulated returns for all assets in one period, and N is the number of assets.

In summary, the simulate_returns function is a tool for generating synthetic asset return data based on specified expected returns and covariance among the assets.
This simulated data can be used to study the properties of financial models, test investment strategies, or understand the impact of estimation uncertainty on portfolio optimization, such as the construction of efficient frontiers.

6.
Firstly, we generate simulated returns spanning a defined period using the provided expected returns μ and covariance matrix Σ.

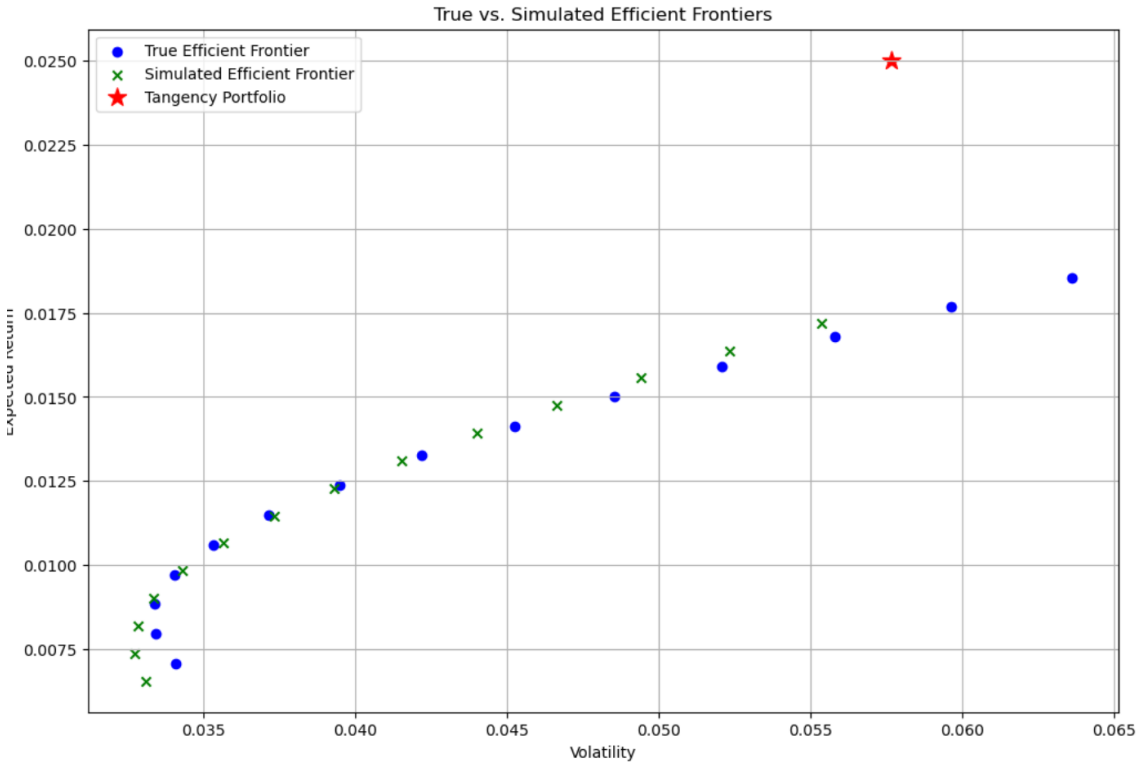
Next, we compute the sample mean and variance-covariance matrix based on the simulated returns, denoted as sample_mu and sample_Sigma, respectively.

Then, we compute the efficient frontier for the simulated data using the sample estimates, employing the previously defined function `compute_efficient_frontier`. This yields the sample_efficient_frontier_df dataframe containing the simulated efficient frontier data.

We proceed to compute the expected returns and volatilities for the simulated efficient frontier, iterating through the dataframe rows. These values are stored in sample_expected_returns and sample_volatilities lists.

Finally, we plot both the true and simulated efficient frontiers on the same graph, allowing for a visual comparison. Additionally, we plot the tangency portfolio on the true frontier for reference.

Plot 3:



The comparison highlights the deviation of the simulated efficient frontiers from the theoretically optimal one. This deviation occurs because the sample estimates, derived from historical data, may not fully capture the true underlying return distribution represented by Σ and μ. Consequently, the simulated efficient frontiers deviate as they are based on these imperfect estimates rather than the true parameters.

7.

In Exercise 7, we perform 100 simulations to analyze the variability of the efficient frontier estimates.

Initially, we plot the true efficient frontier based on the provided expected returns μ and covariance matrix Σ.

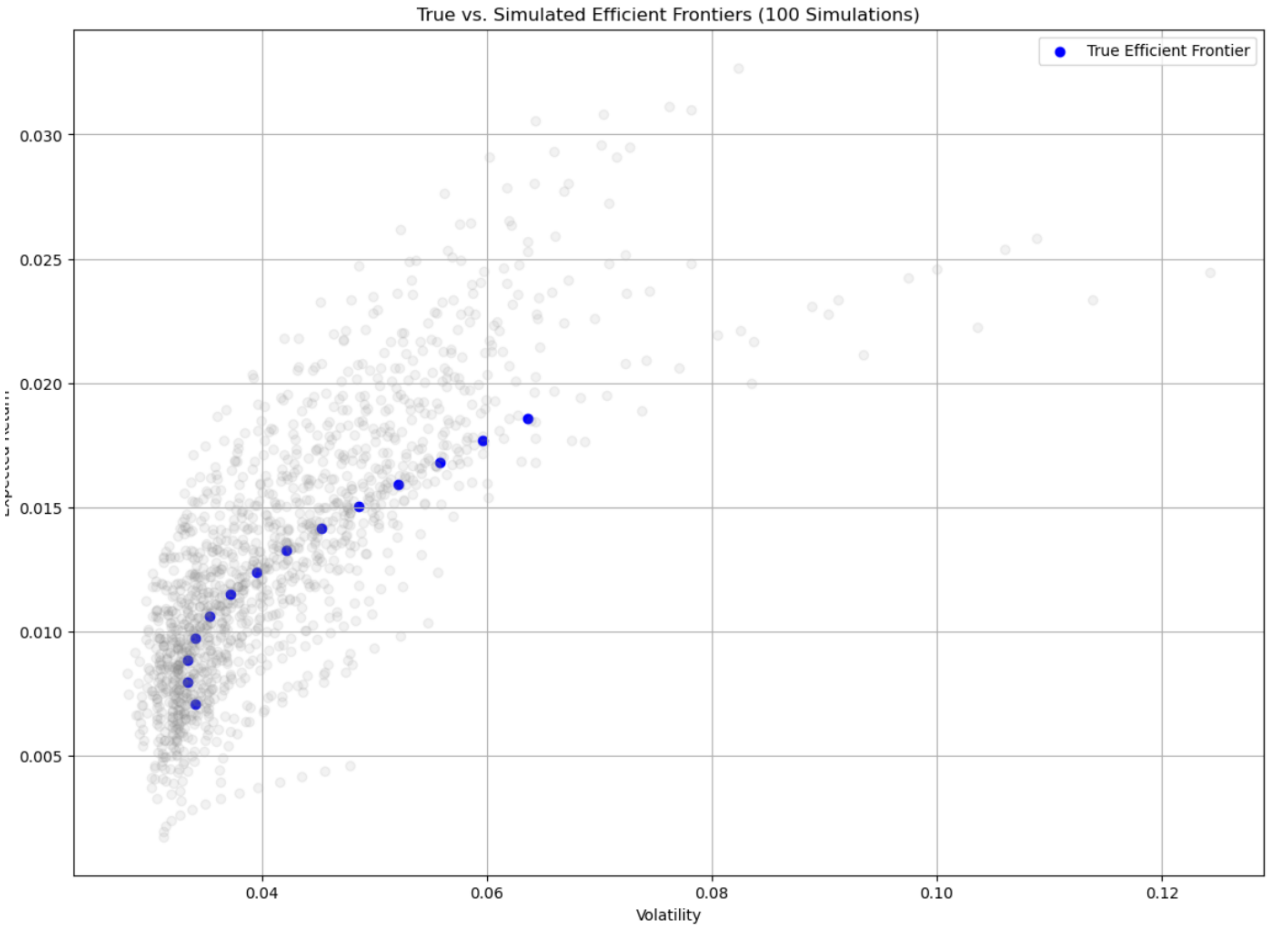
Subsequently, we repeat the simulation process, each time generating simulated returns over a specified number of periods. For each simulation, we compute the sample mean and variance-covariance matrix from the simulated returns.

Next, we compute the efficient frontier for the simulated data using the sample estimates, and then calculate the expected returns and volatilities for each simulated efficient frontier.

We plot each simulated efficient frontier on the same graph, providing insight into the variability and dispersion of the frontier estimates across different simulations.

From the figure, we can conclude that the simulated efficient frontiers exhibit variability around the true efficient frontier. The dispersion of the simulated frontiers suggests the uncertainty associated with estimating the efficient frontier from finite sample data:

Plot 4:



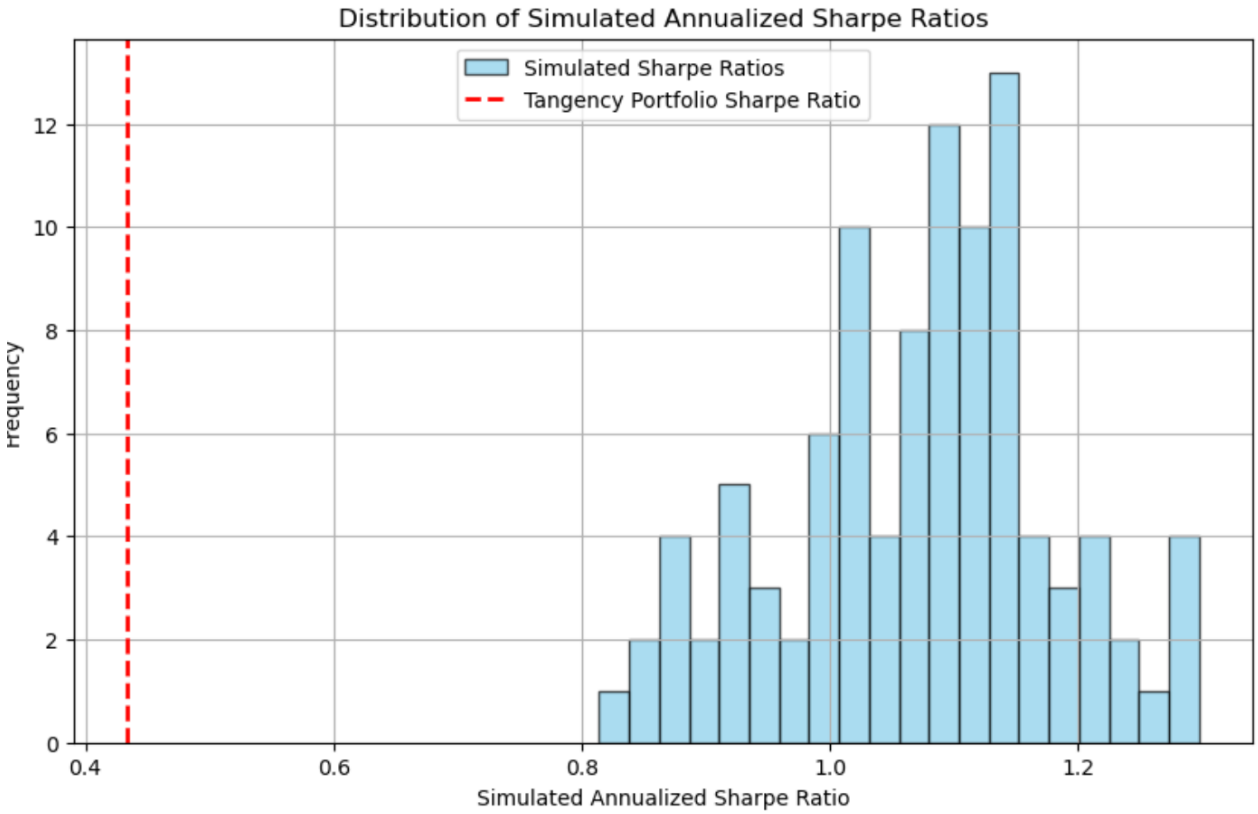
The result of the simulation may depend on the sample size (periods) used for generating returns. Generally, larger sample sizes provide more accurate estimates of the true efficient frontier, as they capture more data points and reduce sampling variability. However, excessively large sample sizes may lead to overfitting or computational challenges. Therefore, the choice of sample size should strike a balance between accuracy and practical considerations.

8 and 9.

In exercise 8 we simulate the Sharpe ratios of the tangency portfolio over 100 iterations. For each simulation, we generate returns, calculate sample mean and covariance matrix, then compute the

tangency portfolio weights using these estimates. From the tangency portfolio, we derive expected return and volatility, subsequently annualizing the Sharpe ratio. Finally, we plot the distribution of simulated Sharpe ratios, alongside a dashed line representing the Sharpe ratio of the tangency portfolio:

Plot 5:



10.

For Exercise 8 and 9, as the sample size (periods) increases, the results tend to stabilize and converge towards more accurate estimates of the true Sharpe ratio distribution. Larger sample sizes provide more data points for estimating parameters such as expected returns and covariance matrices, which improves accuracy in the Sharpe ratio.

When adjusting for the sample size “periods”, we observe that as the sample size increases, the distribution of simulated Sharpe ratios tends to narrow around the true value, indicating reduced variability and more precise estimation of portfolio performance. This suggests that larger sample sizes lead to more reliable estimates and better reflect the underlying characteristics of the asset returns.

Considering Bayesian estimation and robust optimization as potential strategies to improve plug-in estimates, we could possibly return estimation results. We haven’t succeeded in implementing this in our code.