CS109 Challenge Project - Portfolio Explorer

Chandra Suda

1 Introduction

Web App Link: https://portfolioexplorer.vercel.app/
Investors constantly seek to balance risk and return when constructing financial portfolios. Modern portfolio theory provides the mathematical framework to do so, but applying these techniques in a dynamic, real-world application presents significant computational and statistical challenges. I developed *Portfolio Explorer* as part of the CS109 Probability Challenge to address these issues. The project integrates real-world financial data fetched from the Twelve Data API with advanced probabilistic modeling and simulation techniques. I used a Node.js backend to handle API requests and perform all the numerical analysis—from Bayesian return estimation to Monte Carlo portfolio optimization and efficient frontier charts—to provide actionable insights into asset allocations.

2 Methodology

2.1 Bayesian Return Estimation

Rather than relying solely on sample means, I estimate each asset's expected return using Bayesian inference. I begin with a prior distribution (assumed normal) representing historical beliefs about an asset's return. As new data D (such as recent daily returns) becomes available, I update my belief via Bayes' theorem:

$$P(\mu \mid D) = \frac{P(D \mid \mu)\, P(\mu)}{P(D)} \ . \tag{1}$$

Assuming conjugate Normal priors, the update is performed with a precision-weighted scheme. In my JavaScript implementation, the update is carried out as follows:

```
const priorPrec = 1 / Math.pow(priorStd, 2);
const samplePrec = sampleVar > 0 ? n / sampleVar : 0;
const posteriorPrec = priorPrec + samplePrec;
const posteriorStd = Math.sqrt(1 / posteriorPrec);
const posteriorMean = ((priorMean * priorPrec) + (sampleMean * samplePrec)) *
Math.pow(posteriorStd, 2);
```

Here, `priorMean` and `priorStd` represent my initial beliefs, while `sampleMean` and `sampleVar` are calculated from the most recent 30 days of return data (with sample size `n`). This approach produces a posterior estimate that blends historical knowledge with fresh evidence, yielding more stable and realistic return predictions.

2.2 Portfolio Optimization

The next step is portfolio optimization, where I explore thousands of possible asset allocations using a Monte Carlo simulation approach.

2.2.1 Monte Carlo Simulation (runPortfolioSimulation)

I generate a large number of random portfolios by calling a `generateRandomWeights` function that ensures the weights sum to 1 (i.e., a 100% allocation). For each portfolio, I calculate:

- **Expected Return:**
$$E[R_p] = \mathbf{w}^T \boldsymbol{\mu} \times 252,$$
where $\boldsymbol{\mu}$ contains the Bayesian-updated returns.

- **Volatility:**
$$\sigma_p = \sqrt{\mathbf{w}^T \Sigma \, \mathbf{w}} \times \sqrt{252},$$
where Σ is the covariance matrix of asset returns. The covariance between two assets i and j is given by:
$$\mathrm{Cov}(R_i, R_j) = \frac{1}{N-1} \sum_{k=1}^{N} (R_{i,k} - \overline{R}_i)(R_{j,k} - \overline{R}_j).$$

- **Sharpe Ratio:**
$$S = \frac{E[R_p] - R_f}{\sigma_p},$$
where R_f is the risk-free rate.

This simulation produces thousands of candidate portfolios. I then select the **optimal portfolio** by choosing the one with the highest Sharpe ratio, and also identify the **minimum volatility portfolio** by finding the allocation with the lowest risk.

2.2.2 Efficient Frontier Construction (findEfficientFrontier)

To visualize the best trade-offs between risk and return, I construct the efficient frontier. I sort all simulated portfolios by volatility and, for each volatility level, retain the portfolio with the highest return. This process creates a curve—the efficient frontier—that represents the optimal return for each level of risk, clearly showing the best return/risk tradeoffs available.

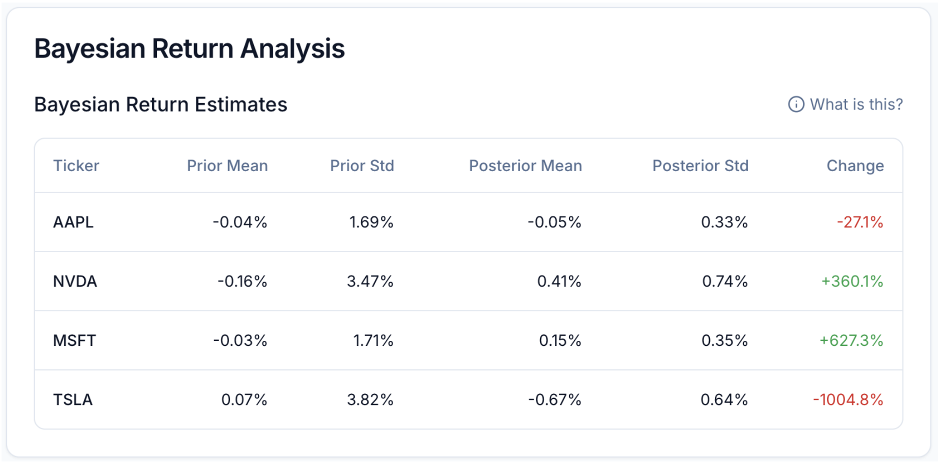
3 Implementation and Technology Stack

I built the backend of *Portfolio Explorer* using Node.js with an Express.js server. This server fetches live financial data from the Twelve Data API and handles all the heavy numerical computations. On the frontend, I employed a modern React 18 framework with TypeScript, ensuring a robust and scalable user interface. For styling and rapid UI development, I used TailwindCSS, which enabled me to efficiently create responsive, modern designs. These key technologies allowed me to deliver a seamless, interactive experience where users can explore financial data, visualize portfolio simulations, and understand the underlying risk-return tradeoffs.

4 Results and Discussion

I tested *Portfolio Explorer* on a sample set of assets using one year of historical daily data. The Bayesian update mechanism effectively reduced uncertainty in return estimates by shifting and narrowing the distribution, as demonstrated in Figure 1. Running the Monte Carlo simulation produced a wide array of portfolio configurations. The resulting risk-return scatter plot revealed a clear efficient frontier, as shown in Figure 2, where portfolios with the highest returns for each risk level were identified.

The optimal portfolio, with the highest Sharpe ratio, achieved an expected annual return of approximately 8.5% and a volatility of 5.4%, yielding a Sharpe ratio near 1.57. The computation of the covariance matrix Σ was critical, as it captured the correlations between asset returns, directly affecting the portfolio volatility via the quadratic form $\mathbf{w}^T \Sigma \mathbf{w}$. This project not only identifies the optimal portfolio based on the highest Sharpe ratio but also provides valuable insights into the behavior of asset returns under varying market conditions. Future work may include more sophisticated prior models and optimization algorithms to further enhance performance and expand the range of assets analyzed.



Bayesian Return Analysis

Bayesian Return Estimates

What is this?

Ticker	Prior Mean	Prior Std	Posterior Mean	Posterior Std	Change
AAPL	-0.04%	1.69%	-0.05%	0.33%	-27.1%
NVDA	-0.16%	3.47%	0.41%	0.74%	+360.1%
MSFT	-0.03%	1.71%	0.15%	0.35%	+627.3%
TSLA	0.07%	3.82%	-0.67%	0.64%	-1004.8%

Figure 1: Prior vs. posterior distribution of expected return for one asset. The Bayesian update narrows uncertainty and shifts the mean towards the sample average.

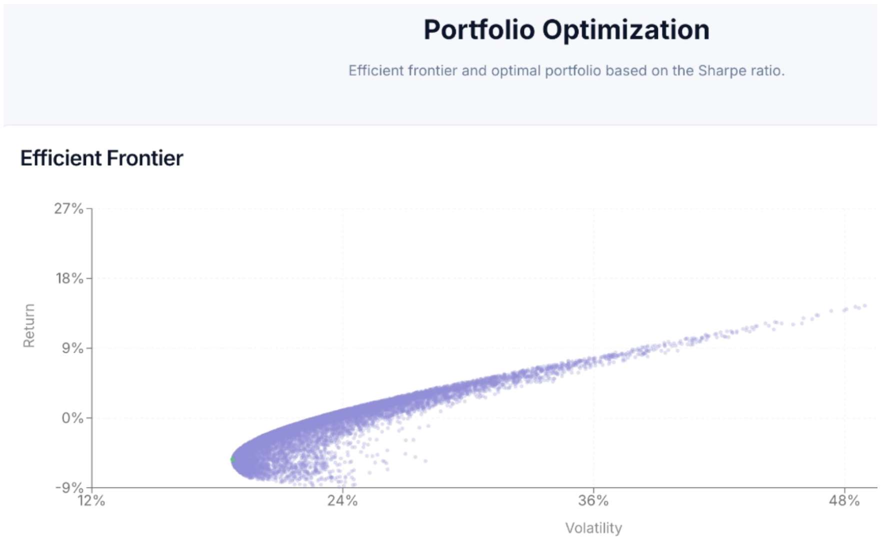


Figure 2: Monte Carlo simulation results for portfolio allocations. Each point represents a portfolio; the efficient frontier (upper boundary) is highlighted, with the optimal portfolio marked.