Introduction to FinTech

Homework 3

1 Announcement

There are 4 programming problems.

2 Preliminaries

1. Create your homework repository by clicking this link and selecting your name from the list.
2. Clone the project to your machine and enter the homework directory.
3. Create Python virtual environment: `python3 -m venv venv`
4. Activate the virtual environment: `source venv/bin/activate`
5. Install necessary dependencies: `pip3 install -r requirements.txt`
6. To leave the virtual environment: `deactivate`

3 Reminder

1. There is **NO** late submission allowed. After the deadline, you will automatically lose write access to the repository. Please ensure you push your code changes to GitHub before the deadline.
2. **You are NOT allowed to modify any protected files, such as .github/*, and we will provide these paths for you. You will receive zero points if any protected file is modified.**
   - `.github/*`
   - `grader.py`
   - `grader_2.py`
   - `Answer/*`

4 Portfolio Theory (70 points)

A portfolio problem in finance involves the allocation of assets in such a way that maximizes returns and minimizes risks under various constraints. Given m assets, the task is to decide the weight vector $\omega \in \mathbb{R}^m$ for each asset at every rebalance date. This assignment focuses on strategies used to manage a portfolio of the 11 sectors in the S&P 500 index, using daily rebalance, with SPY as the benchmark. Transaction costs are ignored for simplicity, and the backtest period is from January 1, 2019, to April 1, 2024.

Students are required to complete the assignment in two Python files. For the first three problems, all work should be completed in `Markowitz.py`. The fourth problem should be completed in `Markowitz 2.py`. The backtesting system is already implemented, so students only need to focus on implementing the main logic of each strategy, which is deciding the weight vector for each rebalance date.

4.1 Equal Weight Portfolio (20 points)

Implement the class `EqualWeightPortfolio`. In an equal weight portfolio, each asset is allocated an identical proportion of the total investment, regardless of the characteristics of the assets. This strategy assumes all assets will contribute equally to the portfolio's risk and return.

4.2 Risk Parity Portfolio (20 points)

Implement the class `RiskParity`. Risk Parity is a portfolio allocation strategy aimed at allocating capital such that each asset contributes equally to the overall risk of the portfolio. This objective is typically achieved by adjusting the portfolio weights inversely to the volatility of the assets. The method can be described mathematically as follows:

Let σ_i be the volatility of asset i, which is typically measured as the standard deviation of the asset's returns. The inverse volatility weight w_i for asset i is given by

$$w_i = \frac{\frac{1}{\sigma_i}}{\sum_{j=1}^{m} \frac{1}{\sigma_j}},$$

where m is the number of assets in the portfolio. This formulation ensures that the weight of each asset is inversely proportional to its risk, thereby attempting to equalize the risk contribution of each asset to the portfolio.

4.3 Mean Variance Portfolio (30 points)

Implement the class `MeanVariancePortfolio`. The Mean-Variance Portfolio strategy employs modern portfolio theory's foundational principles, aiming to maximize the risk-adjusted return. The mathematical optimization problem is expressed as follows:

$$\max_{w} \left(w^\top \mu - \frac{\gamma}{2} w^\top \Sigma w \right)$$

subject to:

$$w_i \geq 0 \quad \forall i \quad \text{(long-only constraint)},$$

$$\sum_{i=1}^{m} w_i = 1 \quad \text{(no leverage constraint)},$$

where:

- w is the vector of portfolio weights for each asset.
- μ is the vector of expected returns for each asset.
- Σ is the covariance matrix of asset returns.
- γ is the risk aversion coefficient, which modulates the trade-off between return and risk. A higher value of γ signifies a higher aversion to risk.

4.4 Judge

Please use the following test to check your final score:

- Equal Weight Portfolio: `python Markowitz.py --score eqw`
- Risk Parity Portfolio: `python Markowitz.py --score rp`
- Mean Variance Portfolio: `python Markowitz.py --score mv`
- All Problems: `python Markowitz.py --score all`

5 My Portfolio (30 points)

In `Markowitz_2.py`, students will develop their own strategy with two specific goals, each corresponding to 15 points:

- Achieve a Sharpe Ratio greater than 1 during the backtest period from 2019 to 2024.
- Outperform the SPY's Sharpe Ratio during the backtest period from 2012 to 2024.

5.1 Judge

Please use the following test to check your final score:

- Sharp Ratio greater than 1: `python Markowitz 2.py --score one`
- Sharp Ratio greater than SPY: `python Markowitz 2.py --score spy`

6 Testing

There are several testing tools, please refer to the `README.md` file for more information.

The provided functions are designed to assist you in visualizing and analyzing the performance of your portfolio strategies. Below is a detailed explanation of each function:

- `plot_performance()`: This function generates a plot of cumulative returns for the SPY (as a benchmark) and your portfolio strategy. It helps in visually comparing the growth of $1 invested in both the SPY and your strategy over time.
- `plot_allocation()`: This function visualizes the asset allocation over time in an area plot format. It shows how the weights of different assets in your portfolio vary over the specified period.
- `report_metrics()`: Computes and displays a full report of various performance metrics including Sharpe Ratio, using the `QuantStats` library.

7 Note

- If you need to install additional Python packages for your homework, list them in the `requirements.txt` file to ensure reproducibility and compatibility in the evaluation environment.
- TA will check your score on Git Action, so please ensure your assignment can run correctly.