UNIT TESTS - Portfolio Optimization Test Suite Documentation

The portfolio optimization framework provided in the code includes a comprehensive test suite to ensure the correct implementation and functionality of various components. The key objectives of the test suite are:

1. **Objective Functions Validation**: Validate the implementation of all the available objective functions, such as minimum variance, mean-variance, maximum Sharpe ratio, and robust optimization. This is crucial to ensure the optimization engine can accurately calculate the optimal portfolio based on the specified objectives.
2. **Constraint Handling**: Thoroughly test the framework's ability to handle different types of constraints, including group constraints, box constraints, turnover constraints, and tracking error constraints. This verifies that the optimized portfolios adhere to the specified investment guidelines and restrictions.
3. **Robust Optimization**: Assess the robustness of the optimization process by validating the calculation of robust metrics, such as worst-case return, diversification ratio, and estimation uncertainty. This helps ensure the framework can effectively address parameter uncertainty in the optimization.
4. **Backtesting and Rebalancing**: Validate the backtesting functionality, including the rebalancing logic, transaction cost calculations, and the ability to handle out-of-sample testing. This ensures the backtest results accurately reflect the real-world implementation of the portfolio strategies.
5. **Efficient Frontier Analysis**: Test the computation of the efficient frontier, verifying the frontier's properties, such as monotonic risk, sufficient number of points, and valid Sharpe ratios. This validates the framework's ability to provide comprehensive insights for portfolio selection and asset allocation decisions.
6. **Reporting and Analytics**: Ensure the reporting and analytics functionalities, such as performance reporting and risk decomposition, are correctly implemented. This allows users to effectively evaluate and communicate the portfolio's risk and return characteristics.

The test suite uses synthetic data with realistic features, including time-varying volatility, sector-based correlations, and momentum/mean-reversion in expected returns. This ensures the tests cover a wide range of market conditions and investment scenarios, providing confidence in the framework's ability to handle real-world portfolio optimization challenges.

Test result:

A screenshot of a computer

Description automatically generated

 **test\_backtest\_initialization**

* Purpose: Verify proper initialization of the RobustBacktestOptimizer class.
* Key Verification: Ensure the lookback\_window and rebalance\_frequency parameters are set correctly.
* Test Parameters: returns\_df, expected\_returns\_df, epsilon\_df, alpha\_df, benchmark\_returns, risk\_free\_rate.

 **test\_objective\_functions**

* Purpose: Validate the implementation of all the available objective functions.
* Key Verification: Ensure the objective function methods return a valid float value without any issues (NaN or Inf).
* Test Parameters: test\_returns, test\_mu, test\_sigma, n\_assets, test\_weights.

 **test\_hierarchical\_risk\_parity**

* Purpose: Test the implementation of the Hierarchical Risk Parity (HRP) method.
* Key Verification: Ensure the clustering and quasi-diagonalization functionalities work as expected, and that all assets are included in the clusters.
* Test Parameters: returns.

 **test\_cvxpy\_optimization**

* Purpose: Validate the CVXPY optimization method and compare the results with the SCIPY optimization method.
* Key Verification: Ensure the total returns from the CVXPY and SCIPY optimizations are similar within a specified tolerance.
* Test Parameters: returns\_df, expected\_returns\_df, epsilon\_df, alpha\_df, constraints.
* Improvement: Added more explicit error checking and handling in the \_optimize\_cvxpy method.

 **test\_robust\_metrics\_calculation**

* Purpose: Test the calculation of robust optimization metrics.
* Key Verification: Ensure the required metrics are calculated correctly and that the values are within expected ranges.
* Test Parameters: returns\_df, expected\_returns\_df, epsilon\_df, alpha\_df.
* Improvement: Adjusted the tolerance for the np.isclose assertion to address numerical precision issues.

 **test\_constraint\_handling**

* Purpose: Verify the comprehensive constraint handling functionality.
* Key Verification: Ensure the weights meet the specified constraints, including long-only, box constraints, tracking error, and turnover.
* Test Parameters: returns\_df, expected\_returns\_df, epsilon\_df, alpha\_df, benchmark\_returns.
* Improvement: Modified the \_validate\_weights method in the PortfolioOptimizer class to properly normalize the weights and ensure they sum to 1 within the specified tolerance.

 **test\_efficient\_frontier\_with\_constraints**

* Purpose: Test the efficient frontier computation with various constraints.
* Key Verification: Ensure the frontier points have the expected properties, such as monotonic risk, sufficient number of points, and valid Sharpe ratios. Also, verify the weight constraints for each frontier point.
* Test Parameters: returns\_df, expected\_returns\_df, epsilon\_df, alpha\_df, constraints.
* Improvement: Handled optimization failures more gracefully in the RobustEfficientFrontier class to increase the number of valid frontier points.

 **test\_reporting\_functionality**

* Purpose: Validate the comprehensive reporting functionality.
* Key Verification: Ensure the performance report is generated correctly and that the calculated metrics (total return, volatility, maximum drawdown) are within reasonable ranges.
* Test Parameters: returns\_df, expected\_returns\_df, epsilon\_df, alpha\_df, benchmark\_returns.

 **test\_portfolio\_rebalancing**

* Purpose: Test the portfolio rebalancing logic.
* Key Verification: Ensure the number of rebalances is within the expected range and that the realized costs are within the specified constraints.
* Test Parameters: returns\_df, expected\_returns\_df, epsilon\_df, alpha\_df.

 **test\_risk\_decomposition**

* Purpose: Validate the risk decomposition calculations.
* Key Verification: Ensure the marginal risk contributions sum up to the total portfolio risk and that the contributions are non-negative.
* Test Parameters: returns\_df, expected\_returns\_df, epsilon\_df, alpha\_df, constraints.