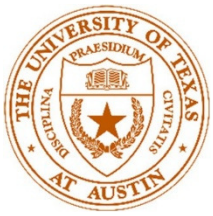

University of Texas at Austin
McCombs School of Business

FIN 684: INVESTMENT THEORY/ACF
Prof. Sury.

Project II (Version A): Minimum Variance Portfolio Optimization

This project focuses on building and comparing minimum variance portfolios constructed using the traditional sample covariance matrix and the Ledoit-Wolf shrinkage estimator for covariance. Students will develop a robust Python program to retrieve financial data from WRDS, perform data coverage validation, apply constrained quadratic optimization for portfolio weights, and analyze out-of-sample performance.

Learning Objectives

- Understand the theory and importance of minimum variance portfolio construction in risk management.
- Develop skills to access, validate, and preprocess financial return data from WRDS.
- Learn to estimate covariance matrices using both sample estimators and Ledoit-Wolf shrinkage.
- Implement constrained quadratic optimization to solve for portfolio weights with position limits.
- Perform rolling window out-of-sample backtesting for portfolio strategy evaluation.
- Analyze and visualize portfolio performance metrics including returns, volatility, turnover, and stability.
- Write clear, modular, and well-documented Python code with commentary.
- Connect quantitative finance theory to practical portfolio optimization challenges faced in real markets.

Professional Relevance

Portfolio optimization is a cornerstone of modern quantitative finance, underpinning asset allocation in investment management. The minimum variance portfolio concept seeks to minimize risk for a given investment universe. However, estimations of covariance matrices from historical data (which are critical inputs to optimization) can be noisy and unstable, leading to poor out-of-sample performance. The Ledoit-Wolf shrinkage estimator addresses this by "shrinking" sample covariance toward a more stable target matrix, improving estimation quality.

The project teaches practical financial concepts and advanced quantitative techniques such as shrinkage covariance estimation, constrained quadratic programming via cvxpy, and detailed portfolio performance analysis. This reflects real-world challenges faced by practitioners in hedge funds, asset managers, and quantitative research, where balancing risk, constraints, and data quality is vital.

Project Requirements and Task Specification

You are tasked with implementing a modular Python program that achieves the following key objectives and deliverables:

1. **Data Acquisition and Coverage Validation**
    - Connect to the WRDS database to retrieve monthly return data for a user-defined set of stock tickers.
    - Implement a ticker data coverage validation routine that ensures stocks have sufficient consecutive return observations in an estimation window, allowing for limited missing months.
    - Design a user-interaction loop to replace insufficient tickers with alternatives until all tickers pass the coverage test.

2. **Portfolio Weight Constraints**
    - Prompt the user to specify maximum long and short position constraints for assets in the portfolio.
    - Design the optimization problem so these constraints can optionally be applied.

3. **Minimum Variance Portfolio Optimization**
    - Calculate the sample covariance matrix from the estimation period return data.
    - Implement Ledoit-Wolf covariance shrinkage estimator using the sklearn package.
    - Solve the minimum variance portfolio by minimizing portfolio variance subject to:
        - Weights summing to 1 (fully invested)
        - User-specified long/short position limits
    - Use cvxpy to perform the constrained quadratic programming optimization efficiently.

4. **Rolling Window Out-of-Sample Backtesting**
    - Implement a rolling window scheme to:
        - Re-estimate covariance matrices monthly with a fixed-length look-back window.
        - Solve for portfolio weights using both sample and Ledoit-Wolf covariance estimates.
        - Compute out-of-sample portfolio returns for the month following each estimation window.

5. **Performance Metrics and Diagnostics**
    - Track and report portfolio performance metrics across the backtest period:
        - Out-of-sample cumulative returns, annualized return, volatility, Sharpe ratio.
        - Portfolio turnover based on weight changes month-to-month.
        - Stability of portfolio weights as measured by cross-sectional standard deviation.
    - Create and interpret visualization plots including:
        - Cumulative return curves.
        - Turnover distributions and time series.

6. **Code Quality and Documentation**
    - Include thorough comments explaining the logic, methodology, and finance concepts at critical points in the code.
    - Structure code in clear, modular functions.
    - Handle exceptions and edge cases gracefully (e.g., failed solver convergence, missing data).
    - Save backtest results as CSV for reporting.

<u>Deliverables</u>

- Fully functioning Python script or Jupyter notebook implementing the above specifications.
- A report (PDF) summarizing the methodology, parameter choices, challenges faced, and results analysis from running the backtest on at least 10 stocks over a multi-year period.

- Annotated plots showing comparative performance, turnover, and stability diagnostics.
- CSV output file containing monthly portfolio statistics for reproducibility.

Suggested Approach

- Begin by implementing the data coverage check module working with WRDS queries for stock returns, ensuring data quality.
- Develop a standalone constrained minimum variance optimizer using cvxpy with weight constraints.
- Integrate Ledoit-Wolf covariance estimation using sklearn.
- Build rolling window logic for repeated optimization and out-of-sample testing.
- Add portfolio diagnostics and plotting functions.
- Test on a small set of liquid stocks before scaling up.
    - Document each step thoroughly for clarity.

APPENDIX: AI Usage Policy for Project Assignment

1. Permitted Uses

- Students may use Large Language Models (LLMs) such as ChatGPT, Claude, or Copilot to assist with programming tasks only.
- Acceptable uses include:
    - Debugging code and troubleshooting errors.
    - Asking for clarification of Python syntax, library functions, or error messages.
    - Generating short code snippets that you then review, adapt, and integrate into your own work.

2. Prohibited Uses

- Students may not use any AI tool to draft or write any portion of their written report/memo. The written report must be entirely student-authored.
- Wholesale outsourcing of coding or analysis to AI tools (e.g., pasting the assignment and asking the LLM to generate the entire solution) is strictly prohibited.
- Students must not rely uncritically on AI outputs. Any AI-generated code must be reviewed, tested, and validated by the student.

3. Citation & Documentation Requirement

- Any use of AI tools must be transparently disclosed in the appendix to your report. This disclosure must include:
    - The tool/model name (e.g., ChatGPT GPT-4, Claude 3.5).
    - The date of usage.
    - The exact prompts provided to the AI.
    - The verbatim responses received.
- All AI-related materials should be placed in an Exhibit (Appendix) to your report and clearly labeled.

4. Responsibility and Verification

- You are fully responsible for the accuracy, validity, and originality of any code you submit, regardless of whether AI tools were consulted.
- Treat AI outputs as *unverified suggestions* — they often contain errors, hallucinations, or incomplete logic. Verification and correctness remain the student's responsibility.
- Failure to disclose AI usage will be treated as an academic integrity violation.

You may use AI tools as a *coding assistant* but not as a *report writer.* Any use must be disclosed with full transparency. AI should support your learning, not replace it!