Portfolio Optimization:
Minimum Variance Portfolio (MVP) and Maximum Sharpe Ratio Portfolio (MSRP)

Mahed Ali
M.Sc. Quantitative Finance - UniBo
mahed.ali@hotmail.com
linkedin.com/mahedali
github.com/mahedalii

August 8, 2025

Abstract

This project presents a portfolio optimization approach using historical stock price data. The project implements and compares two foundational portfolio optimization strategies: the Minimum Variance Portfolio (MVP) and the Maximum Sharpe Ratio Portfolio (MSRP). The objective is to calculate optimal asset allocations for both strategies and the optimization is carried out under the constraints of full allocation and no short-selling. Using adjusted daily closing prices of ten large-cap US stocks from Yahoo Finance (via the 'yfinance' API), we estimate expected returns and the covariance matrix, construct optimal portfolios using constrained optimization, and compare results against the PyPortfolioOpt library. The project is implemented from first principles using NumPy, Pandas, and SciPy, with accompanying visualizations to interpret allocations and risk-return profiles.

Contents

1 Introduction

Portfolio optimization lies at the core of modern quantitative finance and investment management. It entails constructing a portfolio of financial assets that achieves an optimal balance between expected return and risk, typically under a set of predefined constraints. Among the most well-established and widely adopted strategies in this domain are the **Minimum Variance Portfolio (MVP)** and the **Maximum Sharpe Ratio Portfolio (MSRP)**.

The **Minimum Variance Portfolio** seeks to minimize the overall volatility of the portfolio, irrespective of expected return. It is particularly suited for highly risk-averse investors, as it emphasizes the reduction of total portfolio variance. The optimization process is based on the principles of Modern Portfolio Theory[1], which involves identifying a set of portfolio weights that yield the lowest possible risk, subject to budget and non-negativity constraints. Conversely, the **Maximum Sharpe Ratio Portfolio** is designed to maximize the portfolio's risk-adjusted return, measured by the Sharpe Ratio[2], defined as the excess return over the risk-free rate per unit of risk. This approach inherently balances risk and return, resulting in a more return-focused strategy compared to the MVP. Although the resulting optimization problem is nonlinear, it can be solved efficiently using numerical methods.

We implement both optimization strategies manually using Python libraries including `NumPy`, `Pandas`, and `SciPy`. The results of these implementations are validated against those generated by the industry-standard `PyPortfolioOpt` library. Our analysis is based on historical daily data over the period from January 1, 2020 to January 1, 2025.

2 Data and Preprocessing

To construct a diversified and representative portfolio, we selected ten large-cap U.S. equities spanning a variety of sectors. These stocks were chosen to balance exposure across technology, financials, consumer goods, industrials, and energy. The dataset comprises daily adjusted closing prices for the following stocks:

Ticker	**Company**
AAPL	Apple Inc.
GOOGL	Alphabet Inc.
JPM	JPMorgan Chase & Co.
XOM	Exxon Mobil Corp
NVDA	NVIDIA Corporation
TSLA	Tesla Inc.
PG	Procter & Gamble
V	Visa Inc.
JNJ	Johnson & Johnson
CAT	Caterpillar Inc.

[1]Markowitz, H. (1952). *Portfolio Selection.* The Journal of Finance, 7(1), 77–91.

[2]Sharpe, W. F. (1966). *Mutual Fund Performance.* Journal of Business, 39(1), 119–138.

The data was sourced using the `yfinance` API, which provides access to historical price data including adjustments for stock splits and dividends. Only the adjusted closing prices were retained to ensure consistency in return computation. Daily returns were computed using the percentage change method:

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

where P_t denotes the adjusted closing price at time t. The first row (which lacks a prior reference) and any rows with missing data were discarded to maintain clean time series inputs across all assets. To align with industry standards, daily returns were annualized under the assumption of 252 trading days per year:

$$\mu_{\text{annual}} = \mu_{\text{daily}} \times 252, \quad \Sigma_{\text{annual}} = \Sigma_{\text{daily}} \times 252$$

Here, μ denotes the vector of expected returns and Σ is the sample covariance matrix of asset returns. These annualized estimates form the basis for the portfolio optimization procedures discussed in subsequent sections.

3 Methodology

This section outlines the mathematical formulation and computational approach for constructing the Minimum Variance Portfolio (MVP) and Maximum Sharpe Ratio Portfolio (MSRP). Both optimization problems are solved manually using Python's `scipy.optimize` module with the Sequential Least Squares Programming (SLSQP) algorithm.

3.1 Constraints

The optimization tasks are subject to the following standard portfolio constraints:

- **Full capital investment:** $\sum_{i=1}^{n} w_i = 1$
- **No short selling:** $w_i \geq 0 \quad \forall i \in \{1, 2, ..., n\}$

3.2 Minimum Variance Portfolio (MVP)

The objective of the MVP is to construct a portfolio with the lowest possible overall risk, regardless of expected return, subject to the constraints mentioned above. This is achieved by minimizing the portfolio's variance. Mathematically, the optimization problem is:

$$\min_{w} \quad w^\top \Sigma w \qquad \text{subject to:} \quad \sum_{i=1}^{n} w_i = 1, \quad w_i \geq 0 \quad \forall i$$

where:

- $w \in \mathbb{R}^n$ is the vector of asset weights in the portfolio,
- $\Sigma \in \mathbb{R}^{n \times n}$ is the covariance matrix of asset returns.

The problem is solved using the Sequential Least Squares Programming (SLSQP) algorithm via `scipy.optimize.minimize`.

3.3 Maximum Sharpe Ratio Portfolio (MSRP)

The *Sharpe Ratio* is defined as:

$$SR = \frac{w^\top \mu - r_f}{\sqrt{w^\top \Sigma w}}$$

where:

- $\mu \in \mathbb{R}^n$ is the vector of expected returns,
- r_f is the annual risk-free rate (assumed to be 3%)
- $\Sigma \in \mathbb{R}^{n \times n}$ is the covariance matrix

The MSRP formulation seeks to maximize the portfolio's Sharpe Ratio, or equivalently, minimize its negative equivalent:

$$\max_w \quad \frac{w^\top \mu - r_f}{\sqrt{w^\top \Sigma w}} \quad \Longleftrightarrow \quad \min_w \quad -\frac{w^\top \mu - r_f}{\sqrt{w^\top \Sigma w}}$$

Subject to the same constraints:

- $\sum_{i=1}^n w_i = 1$
- $w_i \geq 0$ for all i

This optimization problem is non-linear due to the square root in the denominator, and cannot be solved analytically. We make use of the SLSQP algorithm to solve it numerically.

3.4 Numerical Optimization Using `scipy.optimize`

The `scipy.optimize` module is essential in this context, as mentioned before because the optimization problems such as MSRP involve constraints and nonlinear objective functions which cannot be solved analytically. The `scipy.optimize.minimize` function is used to solve both MVP and MSRP problems. The Sequential Least Squares Programming (SLSQP) algorithm is particularly suitable here as it allows for both equality and inequality constraints, making it ideal for solving constrained optimization problems such as these.

3.5 Validation using PyPortfolioOpt

To validate the manual implementation, we used the `PyPortfolioOpt` library to compare our results with those from the industry-standard. This library provides streamlined functions and efficient methods to compute optimized portfolios using historical mean-variance models. Specifically, the following built-in functions were used:

- `EfficientFrontier.min_volatility()` for MVP
- `EfficientFrontier.max_sharpe()` for MSRP

The resulting weight allocations were cleaned using `.clean_weights()` and compared against our manual results for consistency. Visual comparisons and numerical checks confirmed the correctness of the manual optimization routines.

4 Results and Discussion

4.1 Annualized Returns and Risk Profile

Table 1 presents the annualized mean returns and standard deviations of the selected stocks. High-growth tech stocks such as NVDA (76.86%) and TSLA (75.53%) exhibit exceptional returns, but with elevated volatility levels of 3.39% and 4.23%, respectively. In contrast, consumer staples like JJ (4.52%) and PG (10.80%) show modest returns accompanies by lower risk.

Table 1: Annualized Returns & Standard Deviations

Ticker	Company	Mean Return (%)	Std Dev (%)
AAPL	Apple Inc.	29.79	2.00
GOOGL	Alphabet Inc.	25.76	2.05
JPM	J.P.Morgan Chase & Co.	18.79	2.05
XOM	Exxon Mobil Corp	19.13	2.17
NVDA	NVIDIA Corporation	76.86	3.39
TSLA	Tesla Inc.	75.53	4.23
PG	Procter & Gamble	10.80	1.32
V	Visa Inc.	14.67	1.76
JNJ	Johnson & Johnson	4.52	1.26
CAT	Caterpillar Inc.	25.13	2.05

Figure 1 shows the covariance structure across assets. As expected, high-growth tech stocks tend to exhibit stronger covariance with each other, reducing their diversification benefits when included jointly in a risk-averse portfolio.

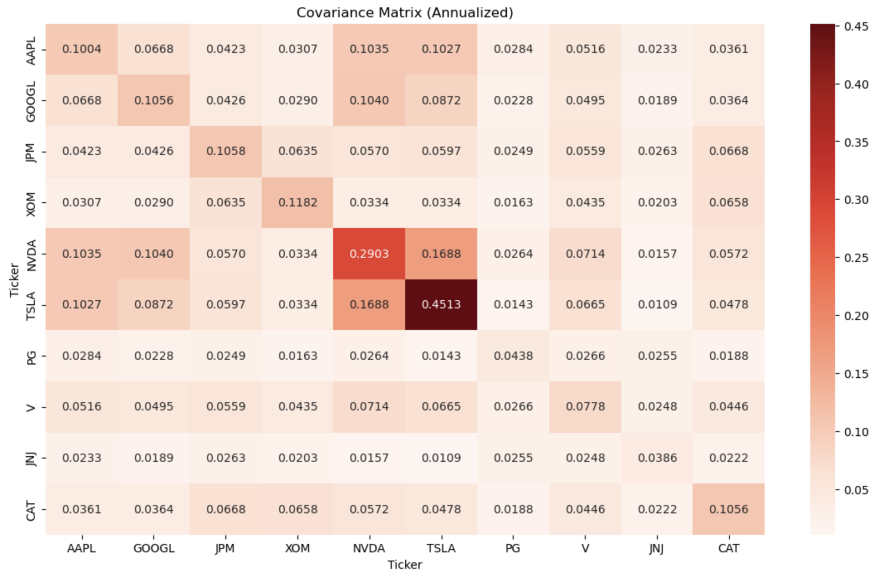


Figure 1: Covariance Matrix

4.2 Minimum Variance Portfolio (MVP)

The MVP allocates capital to minimize total portfolio volatility. As shown in Table 2, the optimization results in a highly conservative allocation, heavily weighting low-risk stocks. JNJ receives the largest allocation (44.63%), followed by PG (32.69%). High-return but volatile assets like NVDA and TSLA are either minimally included or entirely excluded.

Table 2: Minimum Variance Portfolio (MVP) Weights

Ticker	**Company**	**MVP Weight (%)**
AAPL	Apple Inc.	0.00
GOOGL	Alphabet Inc.	8.00
JPM	J.P.Morgan Chase & Co.	0.00
XOM	Exxon Mobil Corp	8.30
NVDA	NVIDIA Corporation	0.00
TSLA	Tesla Inc.	1.90
PG	Procter & Gamble	32.69
V	Visa Inc.	0.00
JNJ	Johnson & Johnson	44.63
CAT	Caterpillar Inc.	4.47

The resulting annualized volatility of the MVP is **17.32%**, reflecting the defensive nature of the portfolio. This portfolio is best suited for risk-averse investors prioritizing capital preservation over return maximization. The optimizer naturally avoids high-variance assets.

4.3 Maximum Sharpe Ratio Portfolio (MSRP)

The MSRP seeks to maximize the risk-adjusted return (Sharpe Ratio) by balancing expected returns against volatility. Under the assumption of a 3% risk-free rate, the optimizer assigns the majority of capital to NVDA (54.48%) and TSLA (22.57%), both of which offer high returns despite their risk levels. Moderate allocations are also made to CAT and XOM.

Table 3: Maximum Sharpe Ratio Portfolio (MSRP) Weights

Ticker	**Company**	**MSRP Weight (%)**
AAPL	Apple Inc.	0.00
GOOGL	Alphabet Inc.	0.00
JPM	J.P.Morgan Chase & Co.	0.00
XOM	Exxon Mobil Corp	8.94
NVDA	NVIDIA Corporation	54.48
TSLA	Tesla Inc.	22.57
PG	Procter & Gamble	0.69
V	Visa Inc.	0.00
JNJ	Johnson & Johnson	0.00
CAT	Caterpillar Inc.	13.32

The MSRP portfolio exhibits an annualized volatility of **41.37%**, significantly higher than the MVP. This trade-off is intentional, aiming to exploit high-return opportunities by concentrating risk in a smaller set of aggressive assets. The exclusion of many stocks, including those with strong historical performance like Apple and Alphabet, indicates that their Sharpe ratios were not competitive under the no-short-selling constraint.

4.4 Comparative Observations

- **Diversification vs. Concentration:** The MVP spreads risk across a broader set of low-volatility assets, whereas the MSRP is highly concentrated in a few high-return stocks. This reflects their opposing objectives: variance minimization versus risk-adjusted return maximization.
- **Effect of Constraints:** The no-short-selling constraint has a visible impact. Several high-return assets (e.g., AAPL, GOOGL) are excluded in MSRP because they do not contribute enough excess return per unit of risk relative to other assets. In MVP, only those assets that actively reduce overall variance are retained.
- **Risk Preference Reflection:** The MVP suits conservative investors focused on minimizing downside risk, while the MSRP appeals to return-seeking investors willing to accept higher volatility in pursuit of excess performance.
- **Real-World Interpretation:** The results mimic realistic investment behavior: defensive portfolios naturally include consumer staples and healthcare (e.g., PG, JNJ), whereas return-seeking portfolios overweight growth stocks (e.g., NVDA, TSLA). The model behavior aligns with financial intuition.

4.5 Conclusion from the Optimization Results

The two optimized portfolios highlight the fundamental trade-off between minimizing risk and maximizing return. While both use the same input data and constraints, their compositions diverge significantly due to their different objective functions. This exercise illustrates how portfolio theory can be practically applied to tailor investment strategies based on individual risk tolerance and return expectations.

5 Validation Using PyPortfolioOpt

To validate the correctness and robustness of our manual optimization approach, we replicated both the Minimum Variance Portfolio (MVP) and the Maximum Sharpe Ratio Portfolio (MSRP) using the `PyPortfolioOpt` library, an industry-standard tool for portfolio optimization. For consistency, the expected returns and covariance matrix were estimated using the same inputs as our manual method:

- **Expected Returns:** Computed using historical mean returns:

$$\mu = \texttt{mean\_historical\_return(close, compounding=False)}$$

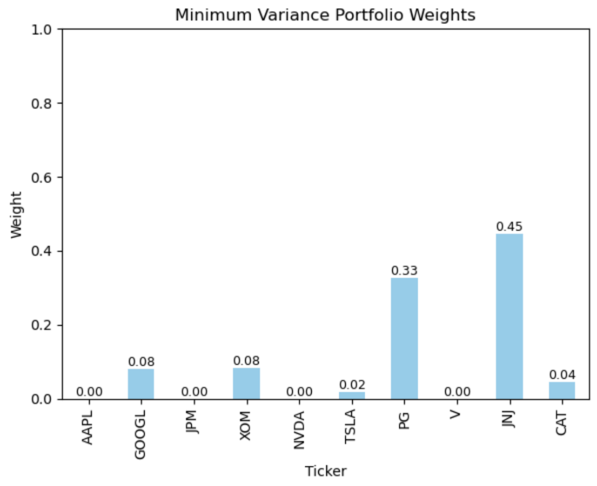
- **Covariance Matrix:** Estimated using the sample covariance:

$$\Sigma = \texttt{risk\_matrix(close, method="sample\_cov")}$$

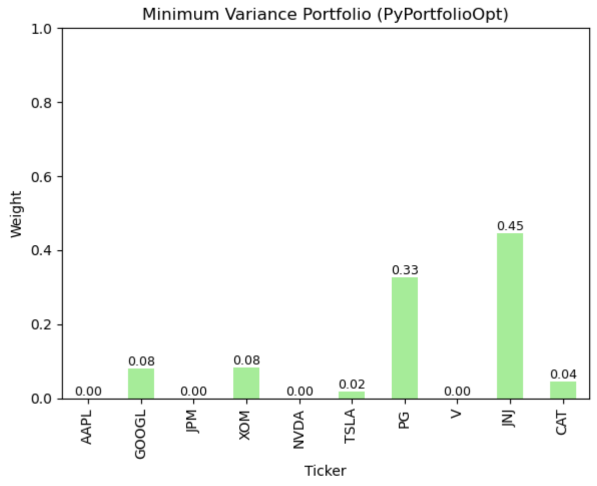
The portfolio optimization was then performed using the `EfficientFrontier` class with the following methods:

- `min_volatility()` to obtain the MVP weights,
- `max_sharpe()` to obtain the MSRP weights (assuming a 3% annual risk-free rate).

The optimized weights from `PyPortfolioOpt` were then compared with our manually computed weights to ensure alignment.

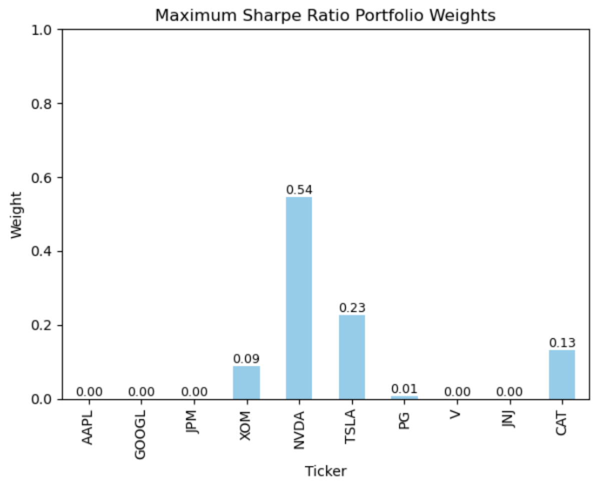


(a) MVP Manual

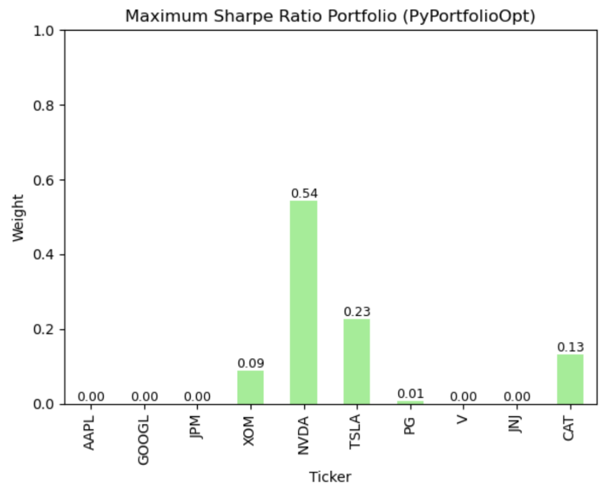


(b) MVP PyPortfolioOpt

Figure 2: Comparison of manually computed and PyPortfolioOpt weights for MVP



(a) MSRP Manual



(b) MSRP PyPortfolioOpt

Figure 3: Comparison of manually computed and PyPortfolioOpt weights for MSRP

The close match between both sets of weights confirms the internal consistency of our implementation and the numerical reliability of the optimization logic. Any minor discrepancies are attributed to differences in solver tolerances, internal normalization, or numerical precision.

6 Conclusion

This project demonstrated the manual implementation of two fundamental portfolio optimization strategies, the Minimum Variance Portfolio (MVP) and the Maximum Sharpe Ratio Portfolio (MSRP), using Python libraries such as `NumPy`, `Pandas`, and `SciPy`. By formulating and solving these optimization problems from first principles, we developed a deeper understanding of how expected returns, risk, and investment constraints interact to determine optimal asset allocation.

The results of our manual implementation closely matched those produced by the industry-standard `PyPortfolioOpt` library, validating the mathematical correctness and robustness of our approach. While MVP resulted in a low-risk, stable allocation suitable for risk-averse investors, the MSRP achieved higher expected returns by taking on greater volatility, aligning with theoretical expectations of risk-return trade-offs.

This exercise also underscored the value of implementing financial models from scratch. Although libraries like `PyPortfolioOpt` offer efficient and reliable solutions, building the models manually helped uncover subtle aspects of optimization logic, constraint handling, and numerical stability often hidden behind abstraction layers.

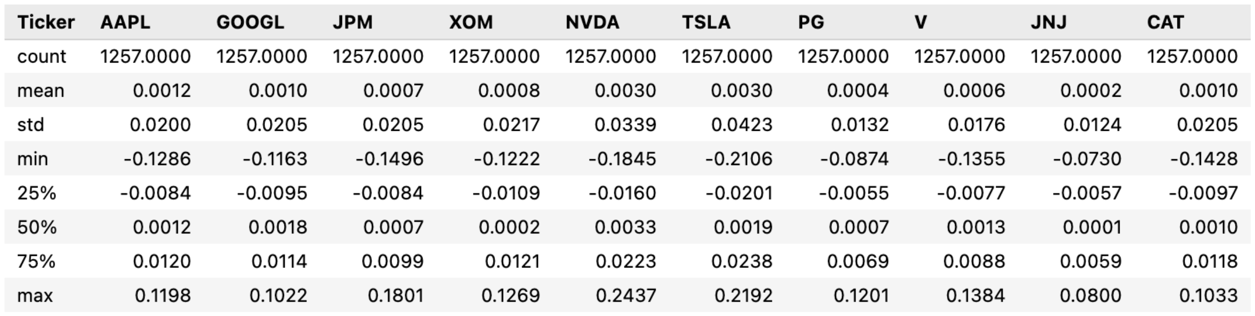
Looking ahead, several extensions can further enhance this work:

- Incorporating realistic portfolio constraints such as sector caps or turnover limits.
- Applying exponential weighting to emphasize more recent returns.
- Performing backtesting and out-of-sample evaluation to assess real-world applicability.
- Exploring more advanced methods such as robust optimization or dynamic portfolio rebalancing.

Overall, this project reinforces the importance of understanding the theoretical and computational foundations of portfolio optimization, especially in quantitative finance contexts where transparency, interpretability, and flexibility are essential.

A Appendix

A. Descriptive Statistics for Daily Returns of the Stocks



Ticker	AAPL	GOOGL	JPM	XOM	NVDA	TSLA	PG	V	JNJ	CAT
count	1257.0000	1257.0000	1257.0000	1257.0000	1257.0000	1257.0000	1257.0000	1257.0000	1257.0000	1257.0000
mean	0.0012	0.0010	0.0007	0.0008	0.0030	0.0030	0.0004	0.0006	0.0002	0.0010
std	0.0200	0.0205	0.0205	0.0217	0.0339	0.0423	0.0132	0.0176	0.0124	0.0205
min	-0.1286	-0.1163	-0.1496	-0.1222	-0.1845	-0.2106	-0.0874	-0.1355	-0.0730	-0.1428
25%	-0.0084	-0.0095	-0.0084	-0.0109	-0.0160	-0.0201	-0.0055	-0.0077	-0.0057	-0.0097
50%	0.0012	0.0018	0.0007	0.0002	0.0033	0.0019	0.0007	0.0013	0.0001	0.0010
75%	0.0120	0.0114	0.0099	0.0121	0.0223	0.0238	0.0069	0.0088	0.0059	0.0118
max	0.1198	0.1022	0.1801	0.1269	0.2437	0.2192	0.1201	0.1384	0.0800	0.1033

Figure 4: Descriptive Stats for Daily Returns

B. Python Libraries Used

- numpy
- pandas
- scipy
- matplotlib
- seaborn
- yfinance
- PyPortfolioOpt

C. Risk-Free Rate Justification

A 3% risk-free rate was assumed for MSRP calculation, representing a conservative estimate of long-term U.S. Treasury bond yields during the study period.

D. GitHub Repository

The complete source code, data preprocessing steps, and portfolio optimization notebook are available on GitHub:

`github.com/mahedalii/portfolio-optimization-mvp-msrp`

References

[1] Markowitz, H. (1952). *Portfolio Selection.* The Journal of Finance, 7(1), 77–91.

[2] Merton, R. C. (1972). *An Analytic Derivation of the Efficient Portfolio Frontier.* The Journal of Financial and Quantitative Analysis, 7(4), 1851–1872.

[3] PyPortfolioOpt Documentation. Retrieved from `https://pyportfolioopt.readthedocs.io`

[4] Ran, A. (2019). *yfinance: Yahoo! Finance market data downloader.* GitHub Repository. `https://github.com/ranaroussi/yfinance`

[5] Sharpe, W. F. (1966). *Mutual Fund Performance.* Journal of Business, 39(1), 119–138.

[6] Virtanen, P., et al. (2020). *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python.* Nature Methods, 17, 261–272.