APS1022

Financial Engineering II

Project Report

Name: Ellie (Yun) Wang, Sally (Shuman) Zhao

Student Number: 1006256631, 1005843991

Instructor: Professor Roy Kwon

University of Toronto

Summer 2025

1. Introduction

This report assumes that the investment universe contains 20 stocks from the S&P 500. It constructs investment portfolios based on four different optimization methods, including Mean-Variance Optimization (MVO), Robust Mean-Variance Optimization with 90% and 95% confidence intervals, Risk Parity, and a Market Capitalization–Weighted Portfolio, and their performances are evaluated respectively.

These portfolios are analyzed in two parts. In Part A, portfolios are trained using stock price data from December 2007 to June 2011 to estimate the sample means and covariances, and then tested using the returns from July 2011. In part B, a 6-month rolling window is used, where each portfolio is built using the previous 6 months of data and tested on the return of the following month, from July to November 2011. To quantify and compare the performances, this report analyzes each portfolio's return, variance, standard deviation, and Sharpe ratio.

2. Data and Parameter Estimation

Using monthly adjusted closing stock prices from December 2007 to June 2011, log returns are computed and used to estimate asset returns. Log scale is applied because it is more likely to be normally distributed, making it suitable for models that rely on normal distribution assumptions, such as MVO. The sample mean and covariance matrix are then computed based on the log returns. These monthly variables are multiplied by m=12 to obtain the annual return and covariance matrix, which is used in constructing the portfolio. The risk aversion coefficient in this report is estimated based on Idzorek's approach:

$$\lambda = \frac{\mu_{mkt} - r_f}{\sigma_{mkt}^2}$$

where μ_{mkt} is the expected annual return of the market portfolio, r_f is the annual risk-free rate, and σ_{mkt}^2 is the variance of market returns.

Market capitalizations are not available from Yahoo Finance, and each stock's market cap is estimated by multiplying the price on June 30, 2011 with its latest shares outstanding. These estimated market capitalizations are used to build the market cap-weighted portfolio, which is treated as a proxy for the overall market. Expected annual return (μ_{mkt}) and variance (σ_{mkt}^2) of the market portfolio are then calculated from this market-cap-weighted portfolio. The risk-free rate (r_f) is retrieved from FRED, and this report uses the U.S. 3-month Treasury Bill rate for June 2011. Based on these variables, the risk aversion coefficient (λ) is determined to be 0.9588, which is applied in all MVO and robust MVO models.

3. Methodology

In this report, four types of portfolios are constructed, and this section outlines the optimization formulas used for each. The methods include MVO, Robust MVO with 90% and 95% confidence intervals, Risk Parity, and Market Capitalization–Weighted. In this report, shorting is allowed in MVO models; therefore, all weights x_i for these models can be negative. However, during the rolling test, short selling is disallowed for all models to mitigate the high instability of the covariance matrix. All models are implemented in MATLAB and are used to generate weights for the 20-stock portfolio.

3.1 Mean-Variance Optimization (MVO)

The MVO method determines the portfolio by maximizing the expected returns (mean) while minimizing the portfolio risk (variance). The formulation for this method is shown below:

$$\min_x \lambda x^T Q x - \mu^T x$$
$$\text{s.t. } e^T x = 1$$

where Q is the annualized covariance matrix, μ is the annualized expected return vector, x is the portfolio weights, and λ is the risk aversion coefficient.

This model is simple and interpretable, and theoretically it allows the determined optimal portfolio to lie on the efficient frontier. However, since the portfolio is determined based on sample estimates of mean and variance, it is very sensitive to estimation errors in μ and Q. Although having more data improves the estimates of variance, it does not improve the estimates of mean. Even a small variation in the mean can lead to a drastic change in portfolio weights. As a result, the portfolio can be unreliable. Moreover, it can also produce an over-concentrated portfolio that weights heavily on a few assets that have high expected returns or low variance, leading to a very unbalanced portfolio.

3.2 Robust Mean-Variance Optimization

Robust MVO improves the traditional MVO by adding an ellipsoidal uncertainty set to account for the estimation error in expected returns. Its formulation is shown as:

$$\min_x \lambda x^T Q x - \mu^T x + \varepsilon \left|\left|\sqrt{\Theta} x\right|\right|_2$$
$$\text{s.t. } e^T x = 1$$

where $\Theta = \text{diag}(Q)/T$ is the measure of uncertainty that serves to standardize the estimated expected return, T is the number of time periods used, and $\epsilon = \sqrt{\chi^2_\alpha(n)}$ is the radius that bounds the standardized distance between μ and μ^{true} which controls the robustness at confidence level α.

In this report, two confidence levels are tested, which are $\alpha = 0.90$ and $\alpha = 0.95$. A higher confidence level α leads to a larger penalty term, so the portfolio relies less on potentially noisy return estimates. By penalizing the portfolio for relying too heavily on uncertain μ estimates, the produced portfolio is more stable and less sensitive to noisy or extreme values in μ. This model also tends to produce more balanced portfolios than standard MVO, which avoids over-concentration in a few assets.

3.3 Risk Parity

The Risk Parity model attempts to generate portfolios by diversifying risks to ensure each asset contributes the same level of risk, which is also known as "Equal Risk Contribution"(ERC). Its formulation is shown as:

$$\min_{x,\theta} \sum_{i=1}^{n} (x_i(Qx)_i - \theta)^2$$
$$\text{s.t. } e^T x = 1$$
$$x_i \geq 0$$

where $R_i = x_i(Qx)_i$ is the risk contribution for each asset, and θ is the target risk contribution.

Since this model relies entirely on risk and does not use expected returns at all, the computed result is more robust to estimation errors, especially in μ, which is very noisy and hard to estimate accurately. As a result, risk parity often produces more stable portfolios compared to standard MVO. The model also produces more diversified portfolios, since weights cannot be zero to avoid zero risk. However, because it completely ignores the expected return, the resulting portfolio may not lie on the efficient frontier, which sacrifices potential return for robustness.

3.4 Market Capitalization–Weighted Portfolio

The market capitalization–weighted portfolio is a passive model that tries to reflect how the real market allocates capital, and is used as a baseline to compare performance from other actively optimized strategies. Each stock's weight is proportional to its market capitalization and is computed as:

$$x_i = \frac{MarketCap_i}{\sum_{j=1}^{n} MarketCap_j}$$

The resulting portfolio is treated as a representation of the overall market.

4. Results and Discussion

Portfolio weights from five strategies are evaluated using both a single-period test (July 2011) and a 6-month rolling backtest from July to November 2011. This section discusses the performance evaluation for all portfolio strategies.

4.1 Single-Period Test (July 2011)

Using data from December 2007 to June 2011 to estimate the inputs, five portfolios are constructed and their performances are evaluated using the returns in July 2011. The table below shows the portfolio return, variance, standard deviation, and Sharpe ratio for each strategy.

Table 1: Portfolio Evaluation for July 2011

Strategy	Return	Variance	StdDev	Sharpe Ratio
MVO	1.3445	2.3241	1.5245	0.88167
Robust MVO 90% CI	0.21134	0.12179	0.34899	0.60443
Robust MVO 95% CI	0.18236	0.095128	0.30843	0.58996
Risk Parity	-0.0042342	0.0266	0.16309	-0.028414
Market Cap	-0.020456	0.045596	0.21353	-0.09767

The Sharpe ratio measures how much additional return is received in excess of the risk-free rate for each additional unit of volatility. A higher Sharpe ratio means that the portfolio is obtaining more return per unit of risk; therefore, it is considered to have better risk-adjusted performance.

From Table 1, it is observed that MVO achieves the highest portfolio return (~134%) and the highest Sharpe ratio (~0.88) among all strategies. However, it also yields the highest portfolio volatility (~1.52), which indicates is high return comes with a very high level of risk. As discussed in Section 3.1, standard MVO is highly sensitive to small errors in the mean estimates. A small, inaccurate estimation of expected return can cause a significant shift in its weight allocation, making the portfolio less reliable.

The Robust MVO strategy with 90% confidence also performs well. It achieves a high return (~21.1%) and a high Sharpe ratio (~0.60), while having a much lower volatility (~0.349) compared to MVO. This shows that Robust MVO not only performs well but also manages risk more effectively. The model penalizes portfolios for relying heavily on uncertain mean estimates, so the resulting portfolio is more stable and balanced.

The strategy with Robust MVO with 95% confidence has very similar results to Robust MVO with 90% confidence, with only slightly lower return (~18.2%) and Sharpe Ratio (~0.59), and slightly higher volatility (~0.308). This is because this strategy applies a stricter penalty on the uncertain mean estimates; therefore, the model relies less on noisy return estimates, and the resulting allocation is more conservative, leading to slightly lower returns.

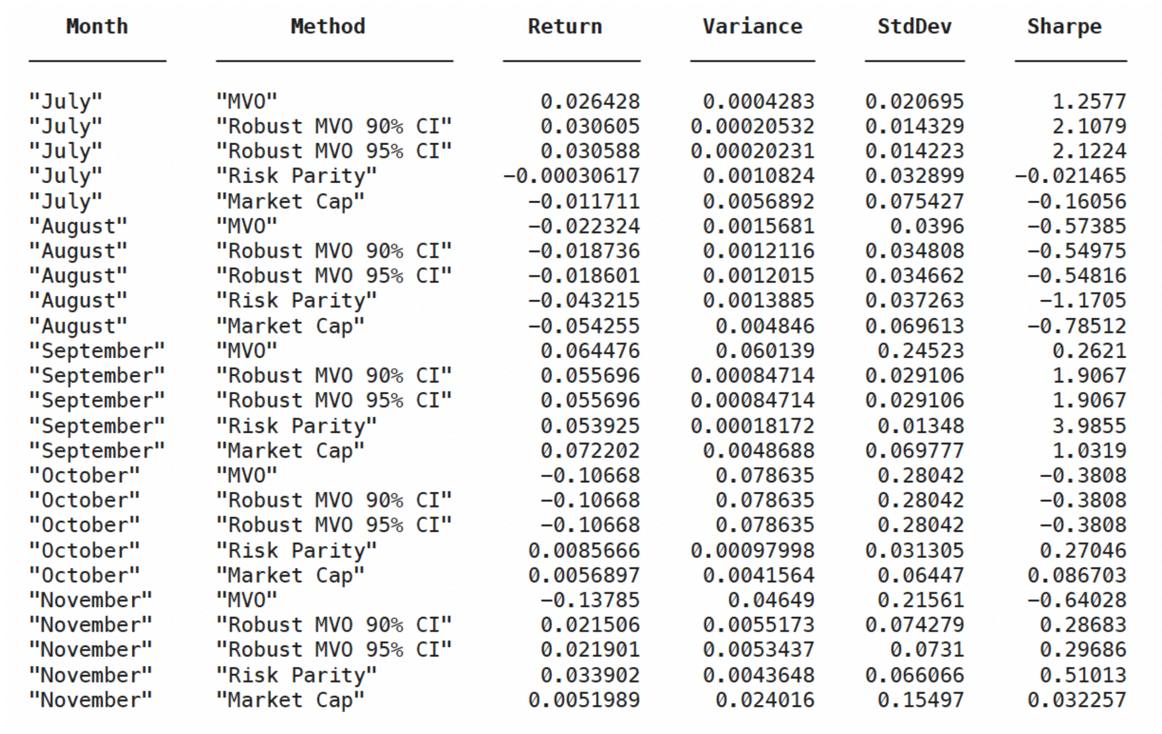
The Risk Parity strategy, despite diversifying risk, also generates a negative return (-0.42%) and a negative Sharpe ratio (-0.028). It has the second-highest volatility, which is around 0.163. Since this strategy does not consider expected returns, the resulting portfolio may not lie on the efficient frontier, and weights may be allocated to assets that have poor returns. Therefore, this method is stable, but it may not perform well when returns differences play a huge role, which is the case here.

However, these four strategies all perform better than the Market Capitalization–Weighted portfolio, which is intended to represent the market portfolio. It has the lowest return (-2.05%) and most negative Sharpe ratio (-0.0977) while having the highest volatility (0.2135). This shows that although not all optimized portfolios achieved positive returns, they still outperform the benchmark; therefore, these strategies are meaningful.

4.2 Rolling 6-Month Horizon Test (July - November 2011)

The second evaluation is conducted using a rolling window approach, where each portfolio is optimized based on the past six months of data and tested on the returns of the following month. As a result, portfolio performances are evaluated over five months, from July to November 2011, and the results are shown in Table 2.

Table 2: Portfolio Evaluation for July - November 2011



Month	Method	Return	Variance	StdDev	Sharpe
"July"	"MVO"	0.026428	0.0004283	0.020695	1.2577
"July"	"Robust MVO 90% CI"	0.030605	0.00020532	0.014329	2.1079
"July"	"Robust MVO 95% CI"	0.030588	0.00020231	0.014223	2.1224
"July"	"Risk Parity"	-0.00030617	0.0010824	0.032899	-0.021465
"July"	"Market Cap"	-0.011711	0.0056892	0.075427	-0.16056
"August"	"MVO"	-0.022324	0.0015681	0.0396	-0.57385
"August"	"Robust MVO 90% CI"	-0.018736	0.0012116	0.034808	-0.54975
"August"	"Robust MVO 95% CI"	-0.018601	0.0012015	0.034662	-0.54816
"August"	"Risk Parity"	-0.043215	0.0013885	0.037263	-1.1705
"August"	"Market Cap"	-0.054255	0.004846	0.069613	-0.78512
"September"	"MVO"	0.064476	0.060139	0.24523	0.2621
"September"	"Robust MVO 90% CI"	0.055696	0.00084714	0.029106	1.9067
"September"	"Robust MVO 95% CI"	0.055696	0.00084714	0.029106	1.9067
"September"	"Risk Parity"	0.053925	0.00018172	0.01348	3.9855
"September"	"Market Cap"	0.072202	0.0048688	0.069777	1.0319
"October"	"MVO"	-0.10668	0.078635	0.28042	-0.3808
"October"	"Robust MVO 90% CI"	-0.10668	0.078635	0.28042	-0.3808
"October"	"Robust MVO 95% CI"	-0.10668	0.078635	0.28042	-0.3808
"October"	"Risk Parity"	0.0085666	0.00097998	0.031305	0.27046
"October"	"Market Cap"	0.0056897	0.0041564	0.06447	0.086703
"November"	"MVO"	-0.13785	0.04649	0.21561	-0.64028
"November"	"Robust MVO 90% CI"	0.021506	0.0055173	0.074279	0.28683
"November"	"Robust MVO 95% CI"	0.021901	0.0053437	0.0731	0.29686
"November"	"Risk Parity"	0.033902	0.0043648	0.066066	0.51013
"November"	"Market Cap"	0.0051989	0.024016	0.15497	0.032257

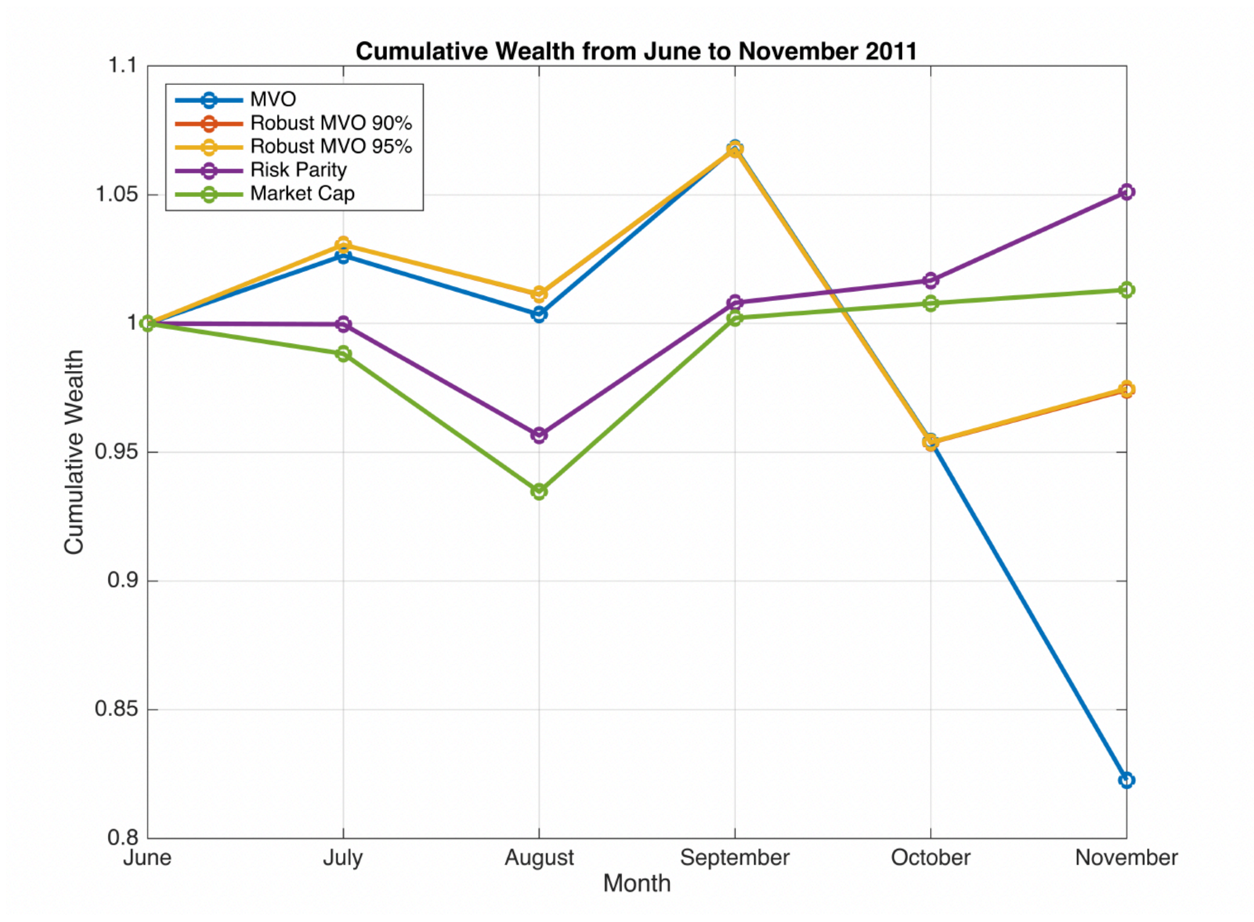


Figure 1: Cumulative wealth from June to November 2011

Figure 1 shows the cumulative wealth of each portfolio from June to November 2011, with the assumption that initial wealth starts from 1. It is observed that Robust MVO with 90% and 95% confidence intervals follow very similar trends, and perform the best at the beginning, from June to September, achieving the highest wealth in September. However, they drop sharply in October, and although they rise again in November, they still end with a lower wealth compared with to initial wealth.

MVO follows a similar trend to Robust MVO strategies initially, and has some gains from July to September; however, its wealth level drops significantly in October and November, leading it to end with the lowest wealth overall. This shows that MVO is highly sensitive to changing market conditions and estimation noise.

Risky Parity portfolio and Market Cap portfolio follow a similar trend; however, Risky Parity consistently remains above the Market Cap portfolio and eventually ends with the highest cumulative wealth. This shows that the Risk Parity strategy is more stable in the long term, even though in the early months, it does not perform as well as the other strategies. The Market Cap portfolio acts as a passive benchmark, stays relatively flat, and ends with a wealth slightly above the initial value.

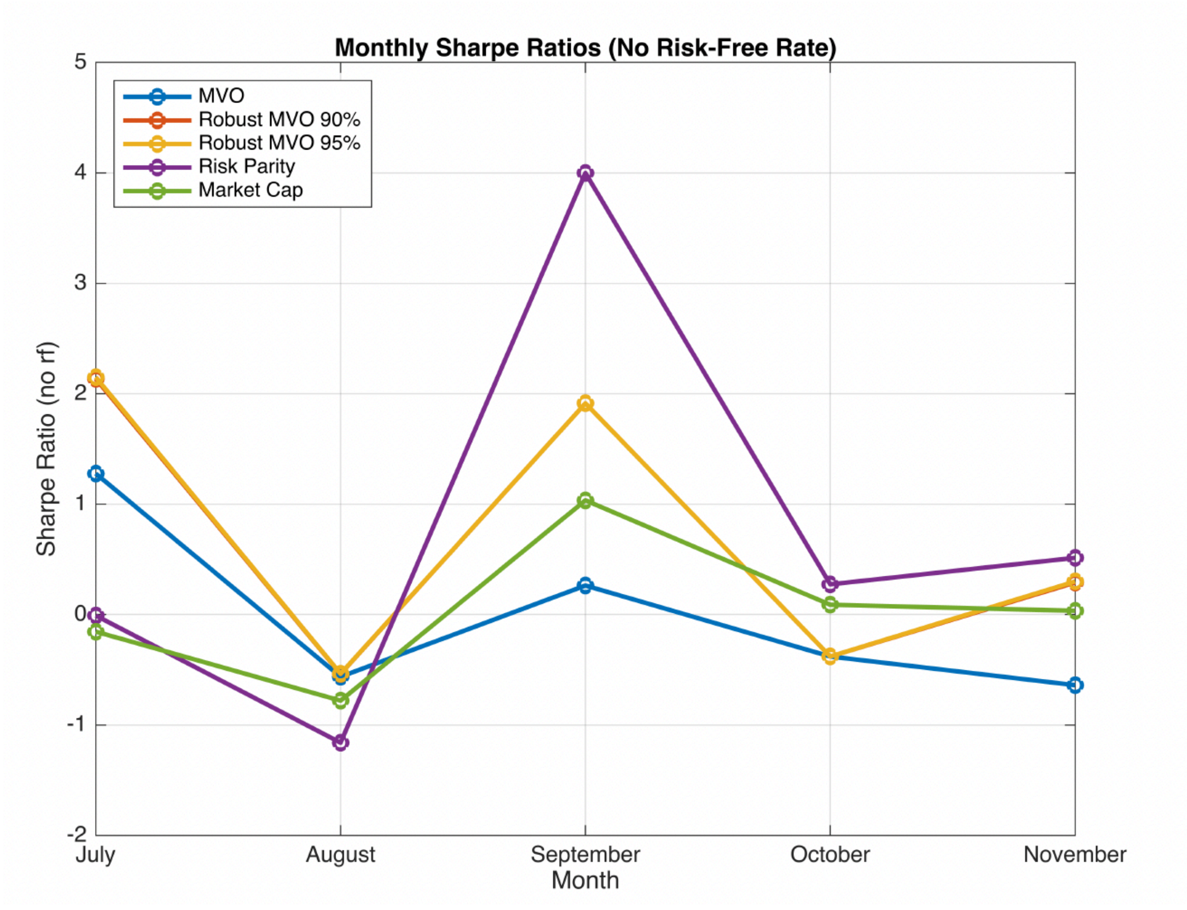


Figure 2: Monthly Sharpe ratios from July to November 2011

Figure 2 shows the monthly Sharpe ratios from July to November 2011. From the plot, it is observed that the Risk Parity strategy yields the lowest Sharpe ratio in August, but then achieves the highest in September at around 4.0, and remains the highest in October and November. This fluctuation shows that although Risk Parity assigns equal risk contribution to assets, its performance can still fluctuate significantly when the market volatility changes. However, it is still very effective in yielding strong risk-adjusted returns. The two Robust MVO strategies are more stable in terms of Sharpe ratios compared to Risk Parity, and achieve the highest Sharpe ratio in July and August. This shows that this method is effective and stable in delivering returns relative to risk under varying market conditions. MVO underperforms in most months, yielding either low or negative Sharpe ratios. Market Cap strategies act as a benchmark, and it is observed that in most months, Risk Parity and Robust MVO perform better than the market portfolio. Overall, Robust MVO and Risk-Parity strategies are effective in delivering strong risk-adjusted returns over time.

5. Conclusion

This report evaluates five portfolio-building strategies, including MVO, Robust MVO with 90% and 95% confidence levels, Risk Parity, and Market Capitalization–Weighted Portfolio. Their performances are evaluated through a one-month test and a five-month rolling window analysis.

The results show that Robust MVO strategies, especially the strategy with 95% confidence level, perform the best in the short term. This is because Robust MVO adds a penalty term to the expected returns, which protects it against errors in estimation. As a result, the portfolio is less sensitive to the noise in returns estimates. Since in the short run, markets usually behave unexpectedly, and estimation noise is very high, this method helps to generate more stable and balanced portfolios, with both high Sharpe ratios and low volatility.

The Risk Parity Strategy is determined to outperform all other strategies in the long term. This is because Risk Parity assigns an equal level of risk to its assets and does not rely on the return estimates. As a result, it is very robust to fluctuations in market conditions. In the long run, the diversified and steady approach helps the portfolio to ride through the market ups and downs more smoothly, and when the short-term noise averages out, the equal risk allocation allows Risk Parity to get higher accumulated wealth over time.

Standard MVO underperforms in both analyses, which is because they are highly sensitive to mean estimation errors and often produces over-concentrated and unreliable portfolios.

Appendix - A

Data Obtaining Sample Code

```
import yfinance as yf
import pandas as pd
from google.colab import files

# Define stock tickers
tickers = ['F', 'CAT', 'DIS', 'MCD', 'KO', 'PEP', 'WMT', 'C', 'WFC', 'JPM',
         'AAPL', 'IBM', 'PFE', 'JNJ', 'XOM', 'COP', 'ED', 'T', 'VZ', 'NEM']

# Date range
start_date = '2011-01-01'
end_date = '2011-11-02'

# Download monthly adjusted close prices
data = yf.download(tickers, start=start_date, end=end_date, interval='1mo',
auto_adjust=True)['Close']

# Drop rows with any missing values
data = data.dropna()

# Save to CSV for later use
data.to_csv('monthly_prices_jan_to_nov.csv')

print("Data shape:", data.shape)
data.tail()

# Download to computer
files.download('monthly_prices_jan_to_nov.csv')
```

Appendix - B

MATLAB Code

Part a

Load data

```
data = readtable('monthly_prices_dec_to_june.csv');

dates = data{:,1};

prices = data{:,2:end};

returns = diff(log(prices));  % log returns
```

Estimate parameters

```
mu = mean(returns);              % monthly mean return

Q = cov(returns);                % monthly covariance matrix

mu_annual = mu * 12;

Q_annual = Q * 12;
```

Obtain risk free rate

```
% Retrieve the actual U.S. risk-free rate for June 2011,

% using the 3-month Treasury Bill rate from FRED, which is

% a standard proxy in portfolio analysis.

c = fred;

data = fetch(c, 'TB3MS', '06/01/2011', '06/30/2011');  % 3-month T-bill

rf = data.Data(end,2) / 100;  % convert % to decimal
```

Compute market mean return using shares and prices data from yfinance

```
% We assume 20 stocks is the market. Since yfinance doesn't provide
historical market caps
```

```
% directly, we estimate it by using formula market cap = shares
outstanding * close price,

% where shares and close price are downloaded from yfinance.

% We will use latest shares outstanding with June 2011 prices (given
that float doesn't

% change much month-to-month) because yfinance only gives current shares
outstanding.

T = readtable('price_and_shares_2011.csv');

prices = T.Price_2011_06_30;

shares = T.Shares_Outstanding;

tickers = T.Ticker;

% Compute Market Capitalizations and Weights

market_caps = prices .* shares;

weights_mktcap = market_caps / sum(market_caps);

r_mkt_series = returns * weights_mktcap;  % full time series of market
return

mean_mkt = mean(r_mkt_series) * 12;       % annualized mean

var_mkt = var(r_mkt_series) * 12;         % annualized variance

lambda_idzorek = (mean_mkt - rf) / var_mkt;

fprintf('lambda = %.4f\n', lambda_idzorek);
```

Construct MVO

```
% Assume:

% - mu_annual: 1×n vector of annualized expected returns

% - Q_annual:  n×n annualized covariance matrix

% - lambda_idzorek: scalar

% - n = number of assets

n = length(mu_annual);

% Define MVO objective function
```

```
mvo_obj = @(x) lambda_idzorek * (x' * Q_annual * x) - mu_annual * x;

% Constraints

Aeq = ones(1, n);

beq = 1;

lb = [];

ub = [];

x0 = ones(n,1)/n;

% fmincon solver

opts = optimoptions('fmincon','Display','none','Algorithm','sqp');

[x_mvo, fval] = fmincon(mvo_obj, x0, [], [], Aeq, beq, lb, ub, [],
opts);

% Display weights

disp('MVO Portfolio Weights:')

disp(array2table(x_mvo', 'VariableNames', tickers))

% Convert weights into table

x_mvo_table = array2table(x_mvo, 'VariableNames', {'MVO Portfolio
Weights'}, 'RowNames', tickers);

disp(x_mvo_table)
```

Construct robust MVO with CI = 90%

```
T_obs = size(returns, 1);            % Number of time periods (e.g., 43
months)

alpha = 0.90;                        % 90% confidence level

epsilon = sqrt(chi2inv(alpha, n));  % sqrt of chi-squared critical value

Theta = diag(diag(Q_annual)) / T_obs;

% Set Up Optimization Problem

robust_obj_90 = @(x) lambda_idzorek * (x' * Q_annual * x) ...

                - mu_annual * x ...

                + epsilon * norm(sqrtm(Theta) * x, 2);
```

```
% Define Constraints

Aeq = ones(1, n);     % sum(x) = 1

beq = 1;

lb = [];     % x >= 0

ub = [];              % no upper bound

x0 = ones(n, 1) / n;  % initial guess: equal weights

% Solve Using fmincon

opts = optimoptions('fmincon', 'Display', 'off', 'Algorithm', 'sqp');

[x_robust_90, fval] = fmincon(robust_obj_90, x0, [], [], Aeq, beq, lb,
ub, [], opts);

% Display weights

disp('Robust MVO Weights (ellipsoidal, 90%):')

disp(array2table(x_robust_90', 'VariableNames', tickers'))

% Convert weights into table

x_robust90_table = array2table(x_robust_90, 'VariableNames', {'Robust
MVO Weights (ellipsoidal, 90%)'}, 'RowNames', tickers);

disp(x_robust90_table)
```

Construct robust MVO with CI = 95%

```
T_obs = size(returns, 1);            % Number of time periods (e.g., 43
months)

alpha = 0.95;                        % 95% confidence level

epsilon = sqrt(chi2inv(alpha, n));  % sqrt of chi-squared critical value

Theta = diag(diag(Q_annual)) / T_obs;

% Set Up Optimization Problem

robust_obj_95 = @(x) lambda_idzorek * (x' * Q_annual * x) ...

                - mu_annual * x ...

                + epsilon * norm(sqrtm(Theta) * x, 2);
```

```
% Define Constraints

Aeq = ones(1, n);     % sum(x) = 1

beq = 1;

lb = [];     % x >= 0

ub = [];              % no upper bound

x0 = ones(n, 1) / n;  % initial guess: equal weights

% Solve Using fmincon

opts = optimoptions('fmincon', 'Display', 'off', 'Algorithm', 'sqp');

[x_robust_95, fval] = fmincon(robust_obj_95, x0, [], [], Aeq, beq, lb,
ub, [], opts);

% Display weights

disp('Robust MVO Weights (ellipsoidal, 95%):')

disp(array2table(x_robust_95', 'VariableNames', tickers'))

% Convert weights into table

x_robust95_table = array2table(x_robust_95, 'VariableNames', {'Robust
MVO Weights (ellipsoidal, 95%)'}, 'RowNames', tickers);

disp(x_robust95_table)
```

Construct Risk Parity Portfolio

```
% define objective function

% Q_annual: covariance matrix (n×n)

n = size(Q_annual, 1);

% Objective function: sum of squared deviations from θ

risk_parity_obj = @(z) sum((z(1:n) .* (Q_annual * z(1:n)) - z(end)).^2);

% set constraints

Aeq = [ones(1, n), 0];     % sum(x) = 1

beq = 1;
```

```
lb = [zeros(n, 1); -Inf];  % x ≥ 0, θ unrestricted

ub = [];                   % no upper bound

x0 = [ones(n, 1)/n; 0.01]; % initial guess for [x; θ]

% solve Using fmincon

opts = optimoptions('fmincon', 'Display', 'off', 'Algorithm', 'sqp');

[z_opt, fval] = fmincon(risk_parity_obj, x0, [], [], Aeq, beq, lb, ub,
[], opts);

x_risk_parity = z_opt(1:n);    % extract weights

% display portfolio weights

disp('Risk Parity Portfolio Weights:')

disp(array2table(x_risk_parity', 'VariableNames', tickers))

% Convert weights into table

x_rp_table = array2table(x_risk_parity, 'VariableNames', {'Risk Parity
Portfolio Weights'}, 'RowNames', tickers);

disp(x_rp_table)
```

Construct a portfolio where weights are based on market capitalizations

```
% the weights are already computed above in the section of

% computing market mean return.

% Convert weights into table

x_mkt_table = array2table(weights_mktcap, 'VariableNames', {'Market Cap
Portfolio Weights'}, 'RowNames', tickers);

disp(x_mkt_table)
```

Compute metrics for each portfolios using July data

```
% load file

P = readtable('monthly_prices_july.csv');

P.Date = datetime(P.Date);  % convert string to datetime

% extract prices

price_0630 = P{P.Date == datetime(2011,7,1), 2:end};  % June 30 prices

price_0731 = P{P.Date == datetime(2011,8,1), 2:end};  % July 29/30
prices

% compute july log returns

r_july = log(price_0731 ./ price_0630)';  % returns as column vector
(n×1)

% obtain rf again

% Retrieve the actual U.S. risk-free rate for July 2011,

% using the 3-month Treasury Bill rate from FRED, which is

% a standard proxy in portfolio analysis.

c = fred;

rf_data = fetch(c, 'TB3MS', '07/01/2011', '07/31/2011');  % 3-month
T-bill

rf_july = rf_data.Data(end, 2) / 100;  % convert % to decimal

% define a function to evaluate portfolios

function [ret, var_, std_, sharpe] = eval_portfolio(w, r, rf, Q)

   ret = w' * r;

   var_ = w' * Q * w;

   std_ = sqrt(var_);

   if std_ > 0

       sharpe = (ret - rf) / std_;

   else

       sharpe = NaN;

   end

end
```

```
% evaluate portfolios

[ret_mvo, var_mvo, std_mvo, sr_mvo] = eval_portfolio(x_mvo, r_july,
rf_july, Q_annual);

[ret_rob90, var_rob90, std_rob90, sr_rob90] =
eval_portfolio(x_robust_90, r_july, rf_july, Q_annual);

[ret_rob95, var_rob95, std_rob95, sr_rob95] =
eval_portfolio(x_robust_95, r_july, rf_july, Q_annual);

[ret_rp,  var_rp,  std_rp,  sr_rp ] = eval_portfolio(x_risk_parity,
r_july, rf_july, Q_annual);

[ret_mkt, var_mkt, std_mkt, sr_mkt] = eval_portfolio(weights_mktcap,
r_july, rf_july, Q_annual);

% display results

strategy_names = {'MVO', 'Robust MVO 90%', 'Robust MVO 95%', 'Risk
Parity', 'Market Cap'};

results = table(strategy_names', ...

   [ret_mvo; ret_rob90; ret_rob95; ret_rp; ret_mkt], ...

   [var_mvo; var_rob90; var_rob95; var_rp; var_mkt], ...

   [std_mvo; std_rob90; std_rob95; std_rp; std_mkt], ...

   [sr_mvo; sr_rob90; sr_rob95; sr_rp; sr_mkt], ...

   'VariableNames', {'Strategy', 'Return', 'Var', 'StdDev',
'SharpeRatio'});

disp('Portfolio Evaluation for July 2011:')

disp(results)

%
```

Part b

Load data (Jan 2011 to Nov 2011)

```
prices = readtable('monthly_prices_jan_to_nov.csv');  % must include
dates from Jan 2011 to Dec 2011
```

```
T = readtable('price_and_shares_2011.csv');

prices.Date = datetime(prices.Date);            % ensure datetime format
```

Rolling window loop

```
test_months = 7:11;  % July to November

results = [];

c = fred;   % Create FRED connection once

prev_rf = 0.0004;  % Default in case of missing data

for m = test_months

   % Define estimation window: previous 6 months

   start_idx = m - 6;

   end_idx = m - 1;

   % Get estimation window log returns

   price_window = prices{start_idx:end_idx, 2:end};  % exclude date
column

   log_returns = diff(log(price_window));            % (5 x 20)

   mu = mean(log_returns);      % monthly mean return

   Q = cov(log_returns);        % monthly covariance

   mu_annual = mu * 12;

   Q_annual = Q * 12;

   % Market cap weights for this month

   % Use price at end of estimation window (i.e., end of month m - 1)

   price_latest = prices{end_idx, 2:end};           % 1×20 row vector

   shares = T.Shares_Outstanding;                   % 20×1 vector

   market_caps = price_latest .* shares';           % 1×20 .* 1×20
(element-wise)

   weights_mktcap = market_caps / sum(market_caps); % normalize to sum
to 1

   % Market return and variance
   r_mkt_series = log_returns * weights_mktcap';

   mean_mkt = mean(r_mkt_series) * 12;

   var_mkt = var(r_mkt_series) * 12;

   % Get risk-free rate for month m-1

   estimation_date = datestr(prices.Date(end_idx), 'mm/dd/yyyy');  %
e.g., '06/30/2011'

   start_rf = datestr(prices.Date(end_idx - 1), 'mm/dd/yyyy');

   end_rf = datestr(prices.Date(end_idx), 'mm/dd/yyyy');

   rf_data = fetch(c, 'TB3MS', start_rf, end_rf);

   if isempty(rf_data.Data)

       warning('Missing FRED data for %s. Using previous rate.',
estimation_date);

       if m > 7

           rf = prev_rf;

       else

           rf = 0.0004;  % fallback if July has no data

       end

   else

       rf = rf_data.Data(end, 2) / 100;  % convert % to decimal

       prev_rf = rf;

   end


   % compute lambda

   lambda_idzorek = (mean_mkt - rf) / var_mkt;

   % === Solve all 4 portfolios using previous code blocks ===

   % 1. MVO --> x_mvo

   % Assume:

   % - mu_annual: 1×n vector of annualized expected returns
   % - Q_annual:  n×n annualized covariance matrix

   % - lambda_idzorek: scalar

   % - n = number of assets

   n = length(mu_annual);

   e = ones(n, 1);

   mu_vec = mu_annual';

   % Define MVO objective function

   mvo_obj = @(x) lambda_idzorek * (x' * Q_annual * x) - mu_annual * x;


   % Constraints

   Aeq = ones(1, n);

   beq = 1;

   lb = zeros(n,1);

   ub = [];

   x0 = ones(n,1)/n;


   % fmincon solver

   opts = optimoptions('fmincon','Display','off','Algorithm','sqp');

   [x_mvo, fval] = fmincon(mvo_obj, x0, [], [], Aeq, beq, lb, ub, [],
opts);

   % 2. Robust MVO with CI=90% --> x_robust_90

   T_obs = size(log_returns, 1);            % Number of time periods
(e.g., 43 months)

   alpha = 0.90;                           % 90% confidence level

   epsilon_rob90 = sqrt(chi2inv(alpha, n));  % sqrt of chi-squared
critical value

   Theta = diag(diag(Q_annual)) / T_obs;

   % Set Up Optimization Problem

   robust_obj_90 = @(x) lambda_idzorek * (x' * Q_annual * x) ...
```

```
                    - mu_annual * x ...
                    + epsilon_rob90 * norm(sqrtm(Theta) * x, 2);
   % Define Constraints
   Aeq = ones(1, n);     % sum(x) = 1
   beq = 1;
   lb = zeros(n, 1);     % x >= 0
   ub = [];              % no upper bound
   x0 = ones(n, 1) / n;  % initial guess: equal weights
   % Solve Using fmincon
   opts = optimoptions('fmincon', 'Display', 'off', 'Algorithm', 'sqp');
   [x_robust_90, fval] = fmincon(robust_obj_90, x0, [], [], Aeq, beq,
lb, ub, [], opts);
   % 3. Robust MVO with CI=95% --> x_robust_95
   alpha = 0.95;                        % 95% confidence level
   epsilon_rob95 = sqrt(chi2inv(alpha, n));  % sqrt of chi-squared
critical value
```

```
   Theta = diag(diag(Q_annual)) / T_obs;
   % Set Up Optimization Problem
   robust_obj_95 = @(x) lambda_idzorek * (x' * Q_annual * x) ...
                    - mu_annual * x ...
                    + epsilon_rob95 * norm(sqrtm(Theta) * x, 2);
   % Solve Using fmincon
   [x_robust_95, fval] = fmincon(robust_obj_95, x0, [], [], Aeq, beq,
lb, ub, [], opts);
   % 4. Risk Parity --> x_risk_parity
   % define objective function
   % Q_annual: covariance matrix (n×n)
   n = size(Q_annual, 1);

   % Objective function: sum of squared deviations from θ

   risk_parity_obj = @(z) sum((z(1:n) .* (Q_annual * z(1:n)) -
z(end)).^2);

   % set constraints

   Aeq_rp = [ones(1, n), 0];     % sum(x) = 1

   beq_rp = 1;

   lb_rp = [zeros(n, 1); -Inf];  % x ≥ 0, θ unrestricted

   ub_rp = [];                   % no upper bound

   x0_rp = [ones(n, 1)/n; 0.01]; % initial guess for [x; θ]

   % solve Using fmincon

   [z_opt, fval] = fmincon(risk_parity_obj, x0_rp, [], [], Aeq_rp,
beq_rp, lb_rp, ub_rp, [], opts);

   x_risk_parity = z_opt(1:n);    % extract weights

   % 5. Market Cap --> weights_mktcap

   % the weights are already computed above

   weights_mktcap = weights_mktcap';

   % ==== Evaluate on next month ====

   price_start = prices{m, 2:end};

   price_end = prices{m+1, 2:end};

   r_test = log(price_end ./ price_start)';

   % Store performance

   [r1, v1, s1, sh1] = eval_portfolio(x_mvo, r_test, rf, Q_annual);

   [r2, v2, s2, sh2] = eval_portfolio(x_robust_90, r_test, rf,
Q_annual);

   [r3, v3, s3, sh3] = eval_portfolio(x_robust_95, r_test, rf,
Q_annual);

   [r4, v4, s4, sh4] = eval_portfolio(x_risk_parity, r_test, rf,
Q_annual);
```

```
   [r5, v5, s5, sh5] = eval_portfolio(weights_mktcap, r_test, rf,
Q_annual);

   results = [results;

       r1, v1, s1, sh1;

       r2, v2, s2, sh2;

       r3, v3, s3, sh3;

       r4, v4, s4, sh4;

       r5, v5, s5, sh5];

end
```

Display results

```
methods = repmat(["MVO"; "Robust MVO 90% CI"; "Robust MVO 95% CI"; "Risk
Parity"; "Market Cap"], 5, 1);

months = repelem(["July", "August", "September", "October",
"November"]', 5);

T_result = array2table(results, ...

   'VariableNames', {'Return', 'Variance', 'StdDev', 'Sharpe'});

T_result.Method = methods;

T_result.Month = months;

T_result = movevars(T_result, {'Month', 'Method'}, 'Before', 'Return');

format shortG

disp(T_result)
```

Part c

Extract returns and Sharpe ratios

```
% Extract monthly returns (reshape to 5 portfolios x 5 months)

returns = reshape(results(:,1), [5, 5]);     % 5 portfolios × 5 months
(July-Nov)

stds    = reshape(results(:,3), [5, 5]);     % corresponding StdDevs
```

```
% Compute cumulative wealth over June-Nov

% Start from June 30 prices → apply July return first

W0 = ones(5, 1);                % Initial wealth = 1 for all portfolios

wealth = zeros(5, 6);

wealth(:,1) = W0;

for t = 1:5

   wealth(:,t+1) = wealth(:,t) .* (1 + returns(:,t));  % recursively
multiply

end

% Compute Sharpe ratios

sharpe = returns ./ stds;
```

Plot cumulative wealth

```
% Plot cumulative wealth from June to November

months = ["June", "July", "August", "September", "October", "November"];

methods = ["MVO", "Robust MVO 90%", "Robust MVO 95%", "Risk Parity",
"Market Cap"];

plot(wealth', '-o', 'LineWidth', 2)

legend(methods, 'Location', 'northwest')

xlabel('Month')

ylabel('Cumulative Wealth')

xticks(1:6)

xticklabels(months)

title('Cumulative Wealth from June to November 2011')

grid on
```

Plot Sharpe ratios

```
plot(sharpe', '-o', 'LineWidth', 2)

legend(methods, 'Location', 'northwest')

xlabel('Month')
```

```
ylabel('Sharpe Ratio (no rf)')

xticks(1:5)

xticklabels(["July", "August", "September", "October", "November"])

title('Monthly Sharpe Ratios (No Risk-Free Rate)')

grid on
```

Appendix - C

Portfolio Weight

```
            MVO Portfolio Weights
            _____________________

    F              1.2787
    CAT           -2.3102
    DIS            6.2093
    MCD           -2.6341
    KO            0.26781
    PEP            7.7859
    WMT          0.064815
    C              0.1239
    WFC           -5.5249
    JPM          -0.27272
    AAPL           2.5689
    IBM            10.703
    PFE          -0.21225
    JNJ           -7.0389
    XOM            1.0108
    COP           -7.9527
    ED           -0.83034
    T             0.27239
    VZ              1.663
    NEM           -4.1725
```

```
              Robust MVO Weights (ellipsoidal, 90%)
              _____________________________________

    F                      0.21046
    CAT                   -0.48048
    DIS                    0.47218
    MCD                     -0.281
    KO                      0.1399
    PEP                    0.83847
    WMT                    0.11128
    C                      0.83594
    WFC                   -0.64041
    JPM                   -0.14037
    AAPL                   0.14656
    IBM                     2.3092
    PFE                  -0.076363
    JNJ                    -1.0619
    XOM                   0.068733
    COP                   -0.76324
    ED                    -0.19051
    T                      0.10558
    VZ                     0.32539
    NEM                   -0.92939
```

```
            Robust MVO Weights (ellipsoidal, 95%)
            _____________________________________

    F                     0.1869
    CAT                 -0.41518
    DIS                  0.37303
    MCD                 -0.24307
    KO                   0.12842
    PEP                  0.71352
    WMT                 0.097129
    C                    0.76853
    WFC                 -0.50911
    JPM                 -0.11898
    AAPL                 0.12869
    IBM                   1.9892
    PFE                -0.061326
    JNJ                 -0.87527
    XOM                 0.044912
    COP                 -0.62815
    ED                  -0.16204
    T                   0.078995
    VZ                   0.29756
    NEM                 -0.79376
```

```
        Risk Parity Portfolio Weights
        _____________________________

F                 0.036936
CAT               0.013179
DIS               0.020277
MCD               0.030853
KO                0.036434
PEP                0.10024
WMT               0.016082
C                 0.061993
WFC               0.058343
JPM               0.034072
AAPL              0.054386
IBM               0.070394
PFE                0.05557
JNJ               0.059023
XOM               0.050083
COP               0.058521
ED                0.056296
T                 0.025291
VZ                0.088359
NEM               0.073662
```

```
            Market Cap Portfolio Weights
            ____________________________

    F                 0.023085
    CAT                0.02146
    DIS               0.030081
    MCD               0.025841
    KO                0.062069
    PEP               0.041388
    WMT               0.060743
    C                 0.033334
    WFC               0.039137
    JPM               0.048765
    AAPL              0.076743
    IBM               0.065331
    PFE               0.047626
    JNJ               0.068599
    XOM                0.15032
    COP               0.031014
    ED               0.0082216
    T                 0.073217
    VZ                0.067279
    NEM               0.025746
```

Appendix - D

Contribution Table

Name	Contribution
Sally (Shuman) Zhao	Primarily worked on the code, debugging, appendix, and proofread the report
Ellie (Yun) Wang	Primarily worked on the report, helped with coding, and debugging