Portfolio

Construction

Using Markowitz Optimization

ECONOMETRICS PROJECT

ROBERT ROMAN
ALBERTO PIO STIGLIANI
TOMMASO DE MARTINO
SAHAR SHIRAZI

TABLE OF CONTENTS

INTRODUCTION

I am an investment manager.
my task is to construct a portfolio for the clients.

My appproach is to use Markowitz Optimization
applying to 4 sectors of the economy.

My goal is to maximize returns while keeping
risk at an acceptable level

we use two approaches and create a final
optimized portfolio to identify our best stocks

Portfolio Details

How do we arrange the stocks?

Financial sector: MSCI WORLD FINANCIAL INDEX
TECHNOLOGY, ENERGY and HEALTH: YAHOO FINANCE

Finacial	Energy	Technology	Health
"JPMORGAN CHASE & CO",	"Exxon Mobil Corporation",	NVIDIA	Johnson & Johnson
"BERKSHIRE HATHAWAY",	"Chevron Corporation",	APPLE	Pfizer Inc.
"VISA",	"ConocoPhillips",	MICROSOFT	UnitedHealth Group Incorporated
"MASTERCARD",	"EOG Resources,",	AMAZON	AbbVie Inc.
"BANK OF AMERICA CORP",	"Williams Companies Inc.",	GOOGLE	Merck & Co., Inc.
"WELLS FARGO & CO",	"Enterprise Products Partners",	META	Thermo Fisher Scientific Inc.
"ROYAL BANK OF CANADA",	"ONEOK",	TESLA	Eli Lilly and Company
"HSBC HOLDINGS (GB)",	"Kinder Morgan",	SALESFORCE	Medtronic plc
"GOLDMAN SACHS GROUP",	"Energy Transfer",	AMD	Amgen Inc.
"COMMONWEALTH BANK OF AUS"	"Schlumberger"	ALIBABA	Gilead Sciences, Inc.

Methodology Overview

1

Monte Carlo Simulation

Generating random portfolios with varying weights across the stocks in order to calculate the expected return, volatility and shape ratio. Efficient Frontier is the result that shows the max return for a given level of risk.

Optimization Problem

Solving an Optimization Problem to find the Optimal Portfolio by using mathematical programming to find the single optimal portfolio that max the shape ratio based on defined constraints.

2

- **Apply both approaches to four key economic sectors: Financials, Energy, Technology, and Healthcare.**
- **We'll identify the 3 stocks with the highest weights in the optimized portfolios from each sector. These stocks represent the most significant contributors to each sector's portfolio performance.**

Theoretical Background

Markowitz Theory

Known as the efficient frontier theory, is a model that helps investors build a portfolio that maximizes expected return for each level of risk, or minimize risk for a given return.

MAIN GOAL: build an investment portfolio that optimizes the relationship between risk and yield.

We are assuming that investors are rational and want to maximize returns and minimize risk

The risk of a portfolio can be reduced through diversification

Key metrics

EXPECTED RETURN:
Calculated as the weighted average of the expected returns of the assets. This gives us an estimate of the potential profit from the portfolio.

$$E(R_p) = \sum_{i=1}^{n} w_i E(R_i)$$

SHARPE RATIO:
Measures the risk-adjusted return of a portfolio.
It measures how much "extra" return you get for each unit of risk. The higher the Sharpe Ratio, the better the portfolio, because you are getting more return for the risk taken.

$$\text{Sharpe Ratio} = \frac{R_p - R_f}{\sigma_p}$$

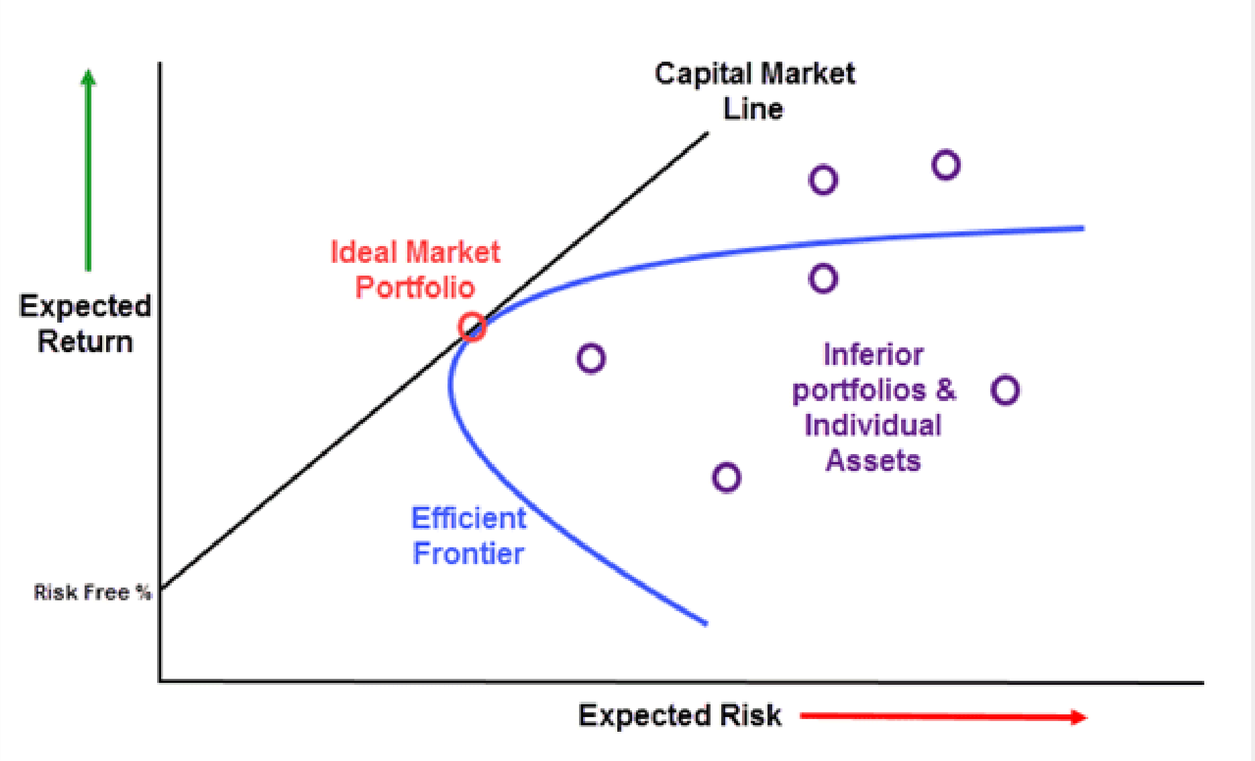
VOLATILITY or RISK:
Measured as the standard deviation of portfolio returns, representing uncertainty or fluctuations.

$$\sigma_p = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \text{Cov}(R_i, R_j)}$$

Efficient Frontier

Markowitz introduced the concept of the efficient frontier, a graph representing optimal portfolios.

- **Each point on the efficient frontier represents a portfolio that offers the maximum possible return for a given level of risk or the minimum risk for a certain return.**



- **However, not all portfolios on the efficient frontier have the maximum Sharpe Ratio. Only one point on the frontier has this maximum value, and that is the tangent portfolio.**
- **The portfolio with the highest Sharpe Ratio is a specific point on this frontier, the most desirable for a rational investor.**

CODES

MONTE CARLO SIMULATIONS

```
#MONTE CARLO SIMULATION

# set seed for reproducibility
np.random.seed(1)

# number of simulations
n = 100_000

# Initialize variables to store portfolio weights, returns, volatility, and Sharpe ratios
port_weights_ = np.zeros(shape=(n,len(data_.columns)))
port_volatility_ = np.zeros(n)
port_sr_ = np.zeros(n)
port_return_ = np.zeros(n)
nume_securities_ = len(data_.columns)

# Log Returns
log_returns_ = np.log(data_ / data_.shift(1))
log_returns_ = log_returns_.dropna()

# Define the ticker for the risk-free rate (^IRX represents the 13-week Treasury Bill yield)
ticker_rf = "^IRX"

# Download historical data for the risk-free rate from Yahoo Finance
rf_data = yf.download(ticker_rf, start=start_date, end=end_date)

# Calculate the most recent risk-free rate in decimal form (last adjusted close value, divided by 100)
rf = rf_data['Adj Close'].iloc[-1].values[0] / 100

# generation of random portfolios with random normalized weights (sum to 1)
for i in range(n):
    # Weight each security
    weights_ = np.random.random(len(data_.columns))
    # normalize it, so that sum is one
    weights_ /= np.sum(weights_)
    #store the weights
    port_weights_[i,:] = weights_

    # Calculation of the annualized expected return
    exp_ret_ = log_returns_.mean().dot(weights_) * 252
    port_return_[i] = exp_ret_fs

    # Annualized expected volatility
    exp_vol_ = np.sqrt(weights_.T.dot(252 * log_returns_.cov().dot(weights_)))
    port_volatility_[i] = exp_vol_

    # Sharpe Ratio
    sr_ = (exp_ret_ - rf) / exp_vol_
    port_sr_[i] = sr_
```

- After downloading the data and cleaning them from the NA values, we set the seed to ensure the reproducibility of the results, then we opted to generate 100,000 random portfolios to explore a wide range of potential portfolio allocations.

- We initialized variables to store portfolio weights, returns, volatility, and Sharpe ratios for each simulation.

- Then we computed the log returns applying the log function to the simple return. The log returns are more suitable for analysis, due to their mathematical and statistical properties, including time additivity

- The 3-Month Treasury Bill (^IRX) was used as the risk-free rate, a key parameter for the Sharpe ratio calculation.

- Finally, we implemented a for loop, using random.random to generate random non-negative values, which were then normalized to sum to one and stored as portfolio weights. Leveraging the log returns and the generated weights, we calculated the expected returns, expected volatility, and Sharpe ratio for each simulation.

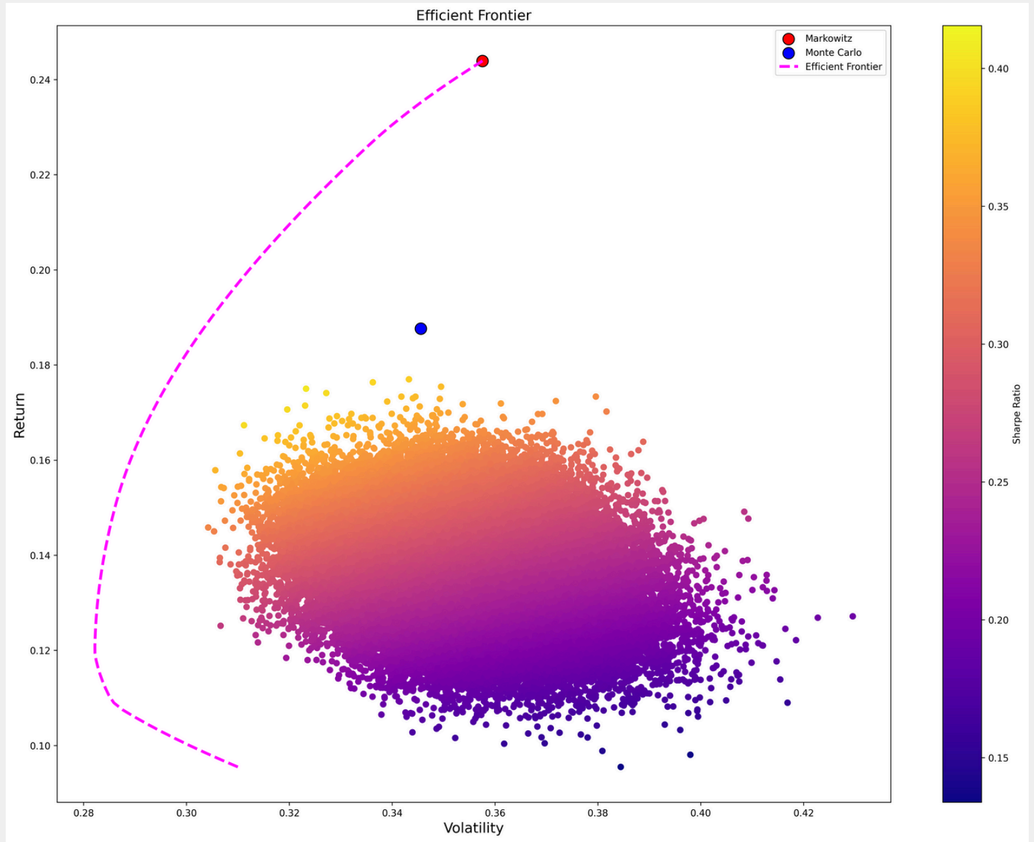
OPTIMIZATION PROBLEM

```python
# Annualized mean log return
log_returns_mean_ = log_returns_.mean()*252

# Annualized covariance matrix
cov_ = log_returns_.cov()*252

# Function to get return, volatility and Sharpe Ratio
def get_ret_vol_sr(weights_):
    weights_ = np.array(weights_)
    ret_ = log_returns_mean_.dot(weights_)
    vol_ = np.sqrt(weights_.T.dot(cov_.dot(weights_)))
    sr_ = (ret_ - rf) / vol_
    return np.array([ret_, vol_, sr_])


# Negative Sharpe ratio
def neg_sr(weights_):
    return get_ret_vol_sr(weights_)[-1] * -1


# check sum of weights
def check_sum(weights_):
    return np.sum(weights_) - 1


# Constraints for the optimization problem
cons_ = {'type':'eq','fun':check_sum} # type = type of constraint
                                      # eq = equality
                                      # fun = function

# Bounds on weights (NON-NEGATIVITY)
bounds_ = [(0, 1) for _ in range(10)]

# Initial guess for optimization to start with
init_guess_ = [1/10 for _ in range(10)]


# Call minimizer
opt_results_ = optimize.minimize(neg_sr, init_guess_,
                           constraints=cons_, bounds=bounds_, method='SLSQP')
```

- First, we calculate the annualized log returns and covariance matrix, which are essential inputs for the optimization problem.
- Next, we define a function that computes the portfolio's return, volatility, and Sharpe Ratio based on given weights, which are before converted into a NumPy array for calculation
- A second function is created to return the negative Sharpe Ratio, allowing us to maximize it indirectly using the scipy.optimize.minimize method, which minimizes the objective function.
- The third function ensures that the sum of the portfolio weights equals 1, maintaining the portfolio's total allocation constraint.
- We then set constraints to maintain this weight-sum condition and add bound to restrict weights to non-negative values.
- The optimization process begins with an initial guess that assumes an equal allocation across all assets.
- In the end, we use the SLSQP (Sequential Least Squares Programming) algorithm in the minimize function to minimize the negative Sharpe Ratio, ultimately determining the optimal portfolio metrics.

VISUALIZATION

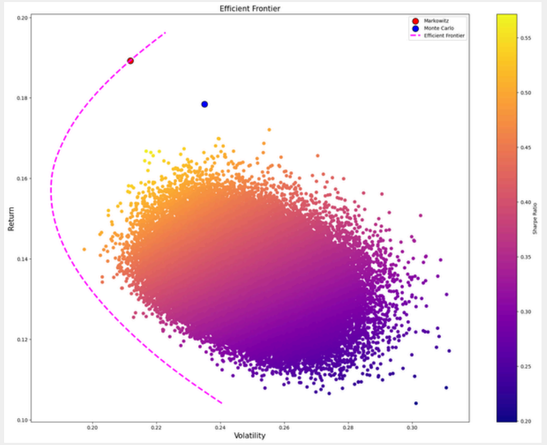
```
# Index of max Sharpe Ratio in Monte Carlo
max_sr_ = port_sr_.max()
ind_ = port_sr_.argmax()
# Return and Volatility at Max SR
max_sr_ret_ = port_return_[ind_]
max_sr_vol_ = port_volatility_[ind_]

# optimal_weights
optimal_weights_ = opt_results_.x
for st, i in sorted(zip(data_,optimal_weights_)):
    print(f'Stock {st} has weight {np.round(i*100,2)} %')

#Optimal portfolio metrics in Markowitz
optimal_ret_, optimal_vol_, optimal_sr_ = get_ret_vol_sr(optimal_weights_)

#Efficient Frontier
frontier_y_ = np.linspace(port_return_.min(), port_return_.max() * 1.3, 100)
frontier_vol_ = []

#funtion to minimize vol
def minimize_vol(weights_):
    return get_ret_vol_sr(weights_)[1]

for possible_ret in frontier_y_:
    cons_ = (
        {'type': 'eq', 'fun': check_sum},
        {'type': 'eq', 'fun': lambda w: get_ret_vol_sr(w)[0] - possible_ret})
    result_ =
optimize.minimize(minimize_vol,init_guess_,method='SLSQP',constraints=cons_,bounds=bounds_)
    frontier_vol_es.append(result_['fun'])


#Plot
plt.figure(figsize=(20, 15))
plt.scatter(port_volatility_, port_return_, c=port_sr_, cmap='plasma')
plt.colorbar(label='Sharpe Ratio')
plt.xlabel('Volatility', fontsize=15)
plt.ylabel('Return', fontsize=15)
plt.title('Efficient Frontier', fontsize=15)
plt.scatter(optimal_vol_, optimal_ret_, c='red', s=150, edgecolors='black', marker='o',
label='Markowitz')
plt.scatter(max_sr_vol_, max_sr_ret_, c='blue', s=150, edgecolors='black', marker='o', label='Monte
Carlo')
plt.plot(frontier_vol_, frontier_y_, c='magenta', ls='--', lw=3, label='Efficient Frontier')
plt.legend()
plt.show()
```



- In this section, we identify the portfolio with the maximum Sharpe ratio and compute the corresponding return and volatility for both the Monte Carlo simulations and Markowitz optimization.
- For the Monte Carlo we use the max() function to identify the highest Sharpe ratio, then, using the index of this maximum we extract the portfolio's associated return and volatility.
- For the Markowitz optimization, the above defined function is called with the optimal portfolio weights to calculate the optimal return, volatility and Sharpe ratio.
- Next, we use np.linspace() to generate 100 equally spaced values between the minimum and maximum returns of the random portfolios.
- For each target return, we minimize the portfolio's volatility using again the SLSQP algorithm in theminimize function, adding two constraints: the sum of portfolio weights must equal 1; the portfolio's expected return must match the target return.
- The results are visualized using a scatter plot where the x-axis represents portfolio volatility, the y-axis represents return, and the color represents the Sharpe ratio.
- The portfolio with the maximum Sharpe ratio from the Monte Carlo simulation is highlighted using a blue dot, while the Markowitz-optimized portfolio is marked with a red dot. The efficient frontier is plotted as a magenta dashed line. This plot provides a clear comparison of the Monte Carlo simulation results, the Markowitz optimization, and the efficient frontier.

Results by Sector

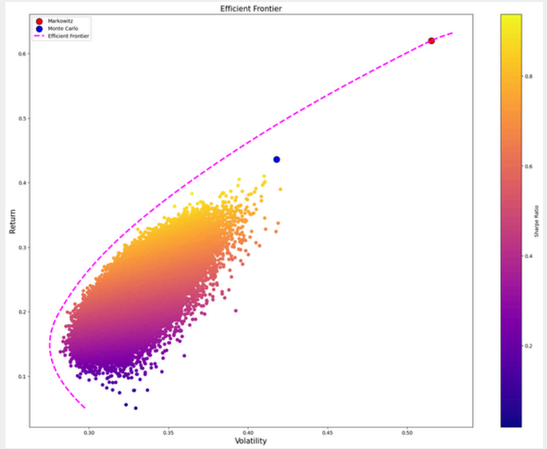
Financial Sector



SharpeRatio: 0.68

Stocks: JPM, BRK-B & CBA.AX

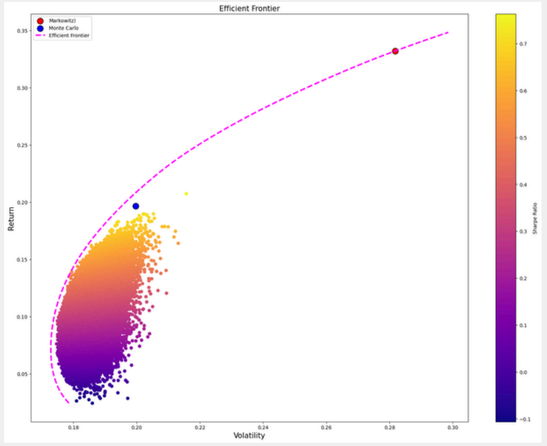
Technology Sector



SharpeRatio: 1.11

Stocks: NVDA & TSLA

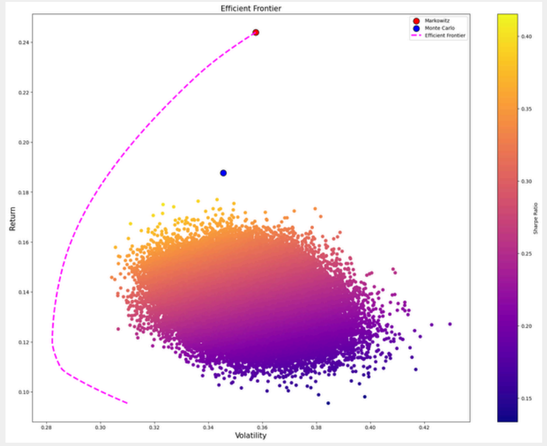
Health Sector



SharpeRatio: 1.02

Stocks: LLY & ABBV

Energy Sector

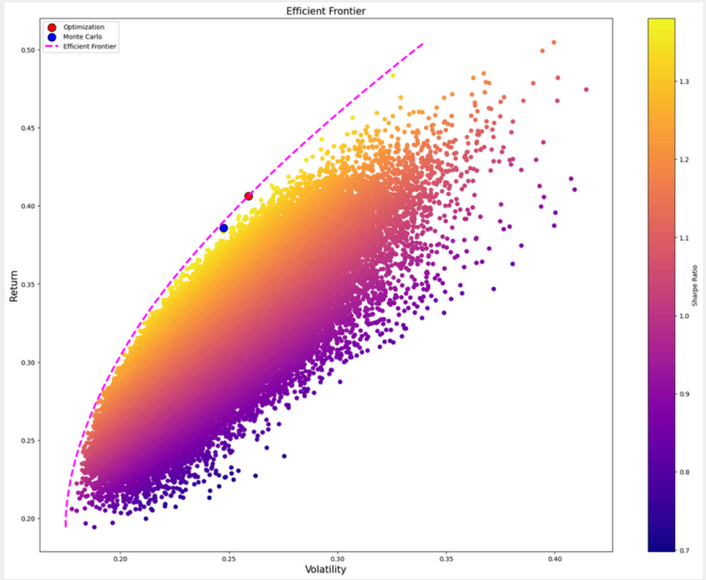


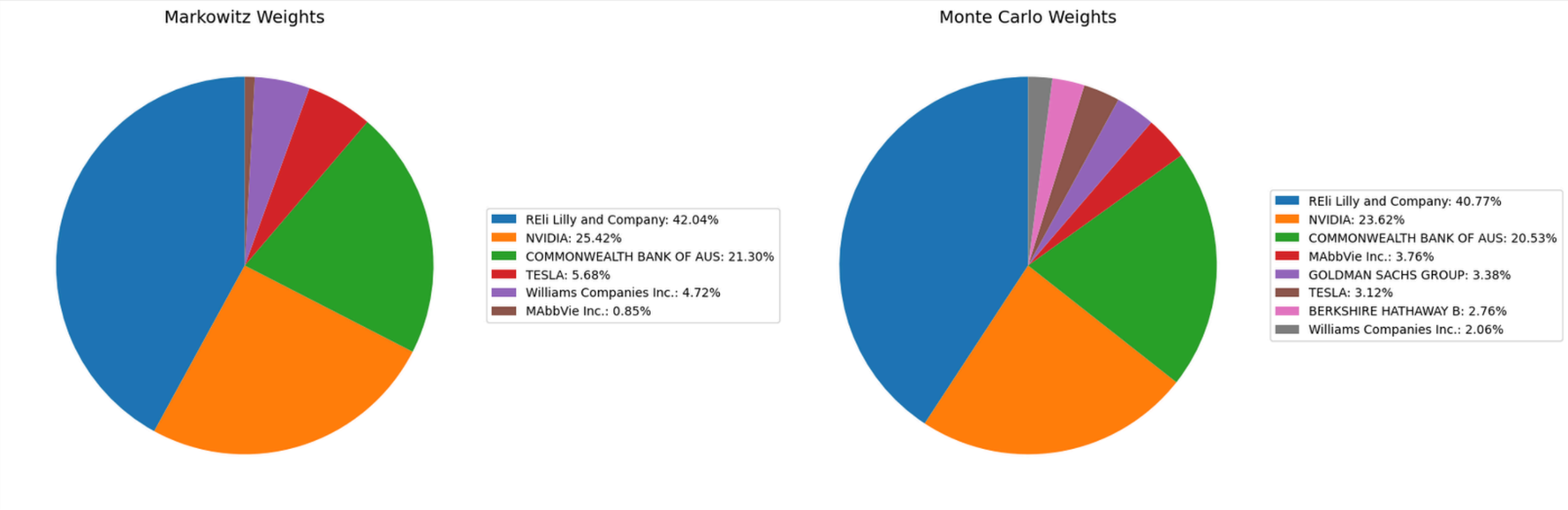
SharpeRatio: 0.55

Stocks: WMB

Final Portfolio Construction and Results:

```
For a given portfolio we have: (Using Markowitz - SciPy)

Return is : 0.40627639504115

Volatility is : 0.25896239653732983

SharpeRatio is : 1.3987605837237878


For a given portfolio we have: (Using Monte Carlo)

Return is : 0.38575239330114636

Volatility is : 0.247542089402651

SharpeRatio is : 1.3803809769386373
```





THANK YOU