WARWICK BUSINESS SCHOOL
THE UNIVERSITY OF WARWICK

Masters Programmes: Assignment Cover Sheet

Student Number:	**5583665**
Module Code:	**IB99L0**
Module Title:	**Financial Analytics**
Submission Deadline:	**13-06-2024**
Date Submitted:	**13-06-2024**
Word Count:	**1672**
Number of Pages:	**10**
Have you used Artificial Intelligence (AI) in any part of this assignment?	**Yes (ChatGPT for coding in ques 3,4,5 and chart in appendix)**

Table of Contents

Introduction

This report aims to analyse various out-of-sample performance strategies, where input parameters have been estimated using historical data (over various window sizes). Specifically, it evaluates the performance of the Mean-Variance model against Minimum Variance as well as the naïve portfolio. Comparison will be conducted over three distinct window sizes: 60 months, 120 months, and 180 months, providing insights into the effectiveness of each portfolio strategy across different periods.

Choice of data set

Explanation of Fama/French 5 Factors	
Factor	**Explanation**
Mkt-RF (Market Risk Premium)	The excess return of the market over the risk-free rate.
SMB (Small Minus Big)	The size premium - captures the excess returns of small-cap stocks over large-cap stocks.
HML (High Minus Low)	The value premium - represents the excess returns of high book-to-market stocks over low book-to-market stocks.
RMW (Robust Minus Weak)	Captures the excess returns of stocks with robust profitability over those with weak profitability.
CMA (Conservative Minus Aggressive)	Reflects the excess returns of firms with conservative investment strategies over those with aggressive investments.

Table 1 - Explanation of Fama/French 5 Factors

By comparing multiple datasets from Ken French's data library (comparison table available in appendix – section 1), it was identified that the Fama/French 5 Factors (2x3) U.S. Research Returns dataset was particularly well-suited. The dataset demonstrated significant explanatory power via its 5 factors. Offering insights into various metrics (available in Table 1) to evaluate investment opportunities and manage risk based on multiple dimensions of portfolio characteristics (FasterCapital 2024). Additionally, the dataset extended across multiple market cycles, covering periods from July 1963 to April 2024. Therefore, providing an extensive historical perspective for robust analysis concerning asset returns across various market conditions. Given that all provided datasets utilize the U.S. Treasury bill as a benchmark, the U.S. Research Returns dataset was deemed most appropriate. In essence, the dataset's comprehensive coverage of market returns offers a strong foundation for estimating return means and covariance matrices, making it particularly suitable for mean-variance and minimum-variance optimization analyses.

Estimation window

To capture the dynamics of these factors over time, three estimation windows at every possible time t, from previous M periods (months) were used to estimate the rolling means and covariances - 5 years (60 months), 10 years (120 months), and 15 years (180 months) (Chen,2019). These windows balance the need for robust statistical estimates and responsiveness to market changes whilst signifying how historical data lengths affect strategy success.

- **M1 - 60 Months (5 Years) -** A 5-year window allows for a balanced perspective between short-to-medium-term market conditions. It captures multiple business cycles and offers a sufficient number of observations to compute reliable statistical measures. The window provides a fair preview of market dynamics (economic conditions and market behaviours) allowing for timely adjustments in portfolio strategies whilst considering market performance and volatility. It is mainly helpful for identifying and responding to rapid market changes. However, it may also be more susceptible to short-term noise (Chen,2019).
- **M2 - 120 Months (10 Years) -** A 10-year window provides a broader view of the market, capturing medium-to-long-term trends and reducing the impact of short-term fluctuations. This window is particularly useful for understanding underlying market trends and the persistence of factors such as size and value premiums. By including a longer period, this window smooths out short-term volatility and noise, leading to more stable and reliable statistical measures. However, the window could create an opportunity cost as medium-term focus might overlook short-term opportunities, potentially leading to missed gains in more volatile markets (Dimitrios, 2022).
- **M3 - 180 Months (15 Years) -** The 15-year window captures a long-term historical perspective and covers multiple complete economic cycles. This facilitates to analyse performance across various phases (including growth, recession, and recovery). Furthermore, it identifies structural changes, ensuring that the estimates are robust and less volatile, thus offering more stable and reliable statistical measures. This window is ideal, as it identifies recurring patterns and linkages in the data, maintaining relevance to current market conditions while offering historical context.

Explanation with graphs available in appendix Section 2

Rolling values

In order to successfully construct a mean-variance optimized portfolio rolling means and covariances need to be considered. These are useful as they allow analysts to dynamically assess the changing correlations between asset returns (which is critical for risk management). To ensure estimates are not systematically biased we use the unbiased estimator method. These are calculated as:

Mean (Expected Return)	Covariance Matrix
$\mu = \frac{1}{M}\sum_{i=1}^{M} r_{t-i}$ • r_{t-i} is the return for period t-i	$\Sigma_t = \frac{1}{M-1}\sum_{i=1}^{M}(r_{t-i} - \mu_t)\,(r_{t-i} - \mu_t)^t$ • μ_t is the rolling mean at time t
This is the average return over the last M periods up to time t.	This measures the degree to which returns move together over the last M periods. The covariance matrix considers the deviations from the rolling mean for each period within the window.

(arazona.edu 2022)

In terms of code, the cov() method by default uses the unbiased estimator (dividing by N−1) where N is the number of observations :

```
rolling_means = df_months.rolling(window=M, min_periods=M).mean().dropna()
rolling_covariances = df_months.rolling(window=M, min_periods=M).cov(pairwise=True).dropna()
```

Mean-Variance Portfolio Optimization

Identifying Target Return - $\bar{r}$

The target expected return is a critical part of portfolio optimization. The metric represents the return level an investor aims to achieve over a specified period. Thus, guiding investment strategies to align with financial goals and risk tolerance. For this analysis, it was determined by averaging the calculated values of 2 determinants, the Capital Asset Pricing Model (CAPM) and the historical average returns.

Calculated as $E[r_i] = r_f + \beta_i(E[r_m - r_f])$ (WBS 2024) CAPM describes the relationship between systematic risk and expected return for assets, particularly stocks. Historical Average Returns, on the other hand, Calculate the historical average returns of assets or portfolios over a specific period and provide a benchmark for future returns.

Setting a target by combining the 2 ensures the return reflects both current market conditions and historical asset performance. Thereby enhancing the investment strategy's robustness.

The target return was calculated to be 0.38.

```
# Average Target

MTarget_return_final = (historical_portfolio_mean_return + MRKTexpected_return)/ 2
print(MTarget_return_final)
✓ 0.0s
0.38266879483675437
```

Optimisation Function

The object function aims to find the optimal weights for a portfolio that minimizes variance for a given target return. The portfolio variance is calculated using the following:

$$\text{Minimize } w^T \sum w$$

- w is the vector of portfolio weights – represented by 'weights' in code
- $\sum$ represents the estimated covariance matrix of asset returns - denoted as 'cov_matrix'

In the code, the optimisation was implemented as:

```
# Mean-Variance Optimization Function
def portfolio_optimization(mean_returns, cov_matrix, target_return):
    num_assets = len(mean_returns)
    args = (mean_returns, cov_matrix)

    def portfolio_variance(weights, mean_returns, cov_matrix):
        return np.dot(weights.T, np.dot(cov_matrix, weights))
```

Constraints

Target Expected Return:

$$\sum_{i=1}^{n} w_i \, \mu_i = \bar{r}$$

- μ is the estimated mean return vector - – represented by 'mean_returns' in code
- $\bar{r}$ is the target expected return – denoted as 'target_return'

```
{'type': 'eq', 'fun': lambda x: np.dot(x, mean_returns) - target_return}
```

Sum of Weights:

$$\sum_{i=1}^{n} w_i = 1$$

- The constraints ensure that the portfolio weights sum to 1 and that short-selling is not allowed (weights are non-negative).

```
{'type': 'eq', 'fun': lambda x: np.sum(x) - 1},
```

Bounds

Bounds ensure that the portfolio weights lie between 0 and 1. This is to avoid short selling (negative weights) and safeguards that no single asset is allocated more than 100% of the portfolio.

```
min_bounds = tuple((0, 1) for asset in range(min_num_assets))
```

Optimization for t+1

To incorporate t+1, the optimization problem is addressed at every possible time t. For each t, the data from the preceding M periods is utilized to estimate the necessary parameters (to calculate mean returns and the covariance matrix). These estimates determine the optimal portfolio weights w_i for that time. Once calculated w_i are then identified for time t, they are used to compute the portfolio's return in the subsequent investment period (from t to t+1). This process assesses the strategy's effectiveness in achieving the targeted return while minimizing risk.

Below is how it was implemented in the code:

```python
# Compute the return obtained by this portfolio in the investment period from t to t+1
next_period_return = np.dot(optimal_weights, df_months.iloc[t + M][['Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA', 'RF']].values)
portfolio_returns.append(next_period_return)
```

Output of optimal weights and expected returns per M period :

```
Optimal Weights for Window Size 60 Months:
Mkt-RF: 0.1191
SMB: 0.0000
HML: 0.0000
RMW: 0.2568
CMA: 0.0029
RF: 0.6212

Target Expected Return: 0.3800
Expected Return of Optimized Portfolio: 0.3800
```

```
Optimal Weights for Window Size 120 Months:
Mkt-RF: 0.2156
SMB: 0.0000
HML: 0.0000
RMW: 0.3275
CMA: 0.0000
RF: 0.4569

Target Expected Return: 0.3800
Expected Return of Optimized Portfolio: 0.3800
```

```
Optimal Weights for Window Size 180 Months:
Mkt-RF: 0.2246
SMB: 0.0000
HML: 0.0000
RMW: 0.2207
CMA: 0.0270
RF: 0.5276

Target Expected Return: 0.3800
Expected Return of Optimized Portfolio: 0.3800
```

Minimum-Variance Optimization

Minimum-variance optimization is a strategy that focuses on minimizing the overall risk (variance) of the portfolio. The model assigns weights to construct a portfolio that has the lowest possible variance (risk) for a given set of assets; without considering the expected returns of the individual assets. The approach is particularly attractive for risk-averse investors who prioritize stability and capital preservation.

Mathematically, the optimization problem is fairly similar to that of Mean-Variance optimisation. It is formulated by minimising $\sigma_p^2 = w^T \sum w$ subject to $\sum_{i=1}^{n} w_i = 1$. However, the difference lies in the target return constraint. The model minimises the portfolio variance without any constraint on the mean return. Thus, ignoring the need to optimise the trade-off between risk and return based on the specified target return or risk level.

Output of optimal weights and expected returns per M period :

```
Optimal Weights for Window Size 60 Months:
Mkt-RF: 0.0000
SMB: 0.0145
HML: 0.0000
RMW: 0.0141
CMA: 0.0111
RF: 0.9603
Expected Return of Optimized Portfolio: 0.1563
```

```
Optimal Weights for Window Size 120 Months:
Mkt-RF: 0.0000
SMB: 0.0101
HML: 0.0000
RMW: 0.0109
CMA: 0.0083
RF: 0.9707
Expected Return of Optimized Portfolio: 0.1035
```

```
Optimal Weights for Window Size 180 Months:
Mkt-RF: 0.0008
SMB: 0.0084
HML: 0.0000
RMW: 0.0073
CMA: 0.0094
RF: 0.9741
Expected Return of Optimized Portfolio: 0.0731
```

Naive 1/N Portfolio

Naive portfolio strategy (often referred to as the 1/N strategy), is an approach where total investment is allocated in equal proportions across all available assets. Rather than using established optimization functions (minimized or maximized), the model instead allocates equally distributed weight values across all assets. The weights are fixed and do not change based on market conditions or asset performance.

Equal weights Objective Function

- Each asset in the portfolio is assigned an equal weight of 1/N, where N is the sum of assets in the portfolio.

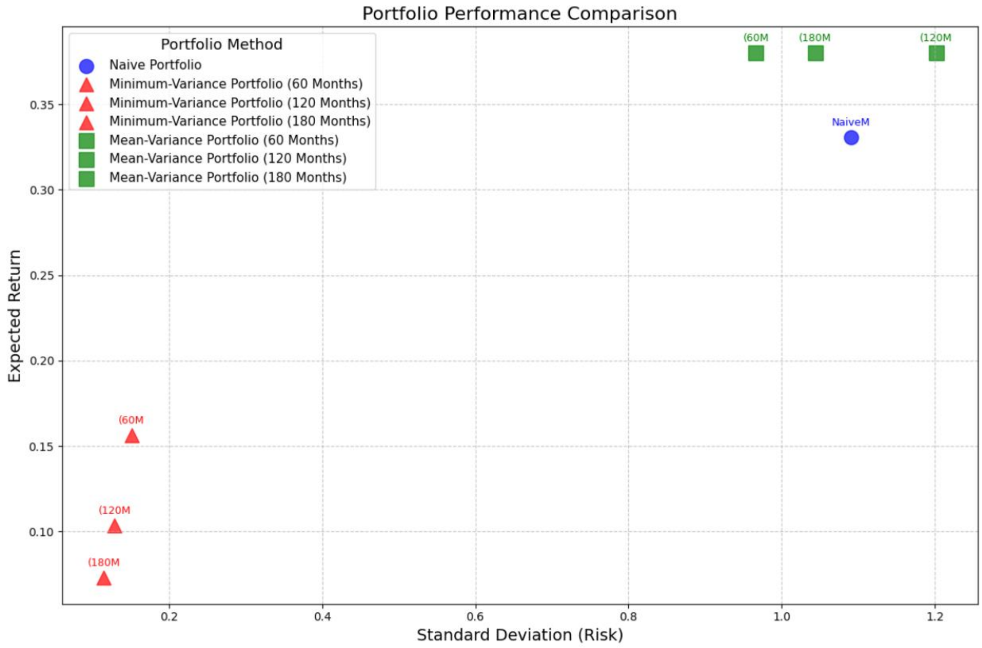
```
def naive_portfolio(mean_returns, cov_matrix, assets):
    num_assets = len(assets)
    equal_weights = np.array([1. / num_assets] * num_assets)
```

In essence, the approach is easy to understand and follows simple implementation. Furthermore, by spreading investments equally, the strategy avoids the biases that come from trying to predict which assets will perform best. However, in exchange for this simplistic view, the strategy forsakes key metrics such as individual risk, return characteristics, or correlations of the assets. Hence, the strategy may prove useful for initial exploratory analyses but may prove less effective in situations when effectively managing risk and returns are required.

Comparative Analysis of Portfolio Strategies

Calculated sample mean and variance of returns for each strategy and estimation window, yielding 7 realized pairs:

Method	Expected Return	Variance	Standard Deviation
Naive Portfolio	0.3305486968	1.1888805975	1.0903580134
Minimum-Variance Portfolio (60 Months)	0.1562986892	0.0229408052	0.1514622238
Minimum-Variance Portfolio (120 Months)	0.1034958762	0.0166340449	0.1289730393
Minimum-Variance Portfolio (180 Months)	0.0730886063	0.0132120676	0.1149437586
Mean-Variance Portfolio (60 Months)	0.3799999995	0.9340397913	0.9664573407
Mean-Variance Portfolio (120 Months)	0.3799999999	1.4443799738	1.2018236035
Mean-Variance Portfolio (180 Months)	0.38	1.089445539	1.0437650785



Mean-Variance Portfolio – Values of the model proved to be best overall with the Portfolio demonstrating consistently high expected returns across different time horizons. As can be seen on the graph, all points are located in the upper right quadrant. In particular, the 180-month window size achieved the highest expected return (0.38) with a relatively low standard deviation (1.04) in comparison to the shorter 120-month window (SD = 1.20). In essence, the consistent expected return across different horizons suggests robust performance with varying levels of risk.

Minimum-Variance Portfolio – Clustered at the lower left of the plot, with varying return levels. The portfolio provided the lowest expected returns but at the lowest risk (as reflected by the lower variance and standard deviation). The 60-month window offers the highest expected return (0.156) with a relatively low standard deviation (0.15). However, over longer time horizons the expected returns are reduced and matched by decreasing risk. As a result, these portfolios are ideal for conservative investment strategies or for risk-averse investors seeking to minimize risk.

Naive Portfolio – Positioned in between with one of the highest returns (0.33) and risk (1.09). In this scenario, it has proven suitable for investors with a high-risk tolerance. However, considering how the model was calculated (1/N) it could prove to be risky to use in a highly volatile environment.

Conclusion

By comparing portfolios, it can be identified that different window sizes play a pivotal role in the estimation of optimized portfolios. Furthermore, the choice of strategy for optimisation drastically effects expected returns. Therefore, investors should identify their risk tolerance and return expectations before selecting a portfolio strategy. In essence, the 180-month window size Mean-Variance Portfolio presents the highest expected return with moderate risk which makes it a compelling option for investors seeking higher returns. On the other hand, the Minimum Variance Portfolio provides the lowest risk, suitable for conservative investors. Lastly, the Naive Portfolio is suitable for investors with a high-risk tolerance who prefer a straightforward, non-optimized investment approach.

References

arazona.edu (2022) Unbiased estimation. Available at: https://www.math.arizona.edu/~jwatkins/N_unbiased.pdf (Accessed: 6 June 2024).

CFI Team (2022). Types of Financial Analysis. [online] Corporate Finance Institute. Available at: https://corporatefinanceinstitute.com/resources/accounting/types-of-financial-analysis/. (Accessed: 13 June 2024).

Chen, D. (2019) Fama french five factors, Fama French Five Factors - QuantConnect.com. Available at: https://www.quantconnect.com/research/15262/fama-french-five-factors/p1 (Accessed: 12 June 2024).

Dimitrios, S.(2022) Fama and French: The five-factor model revisited (2022) CFA Institute Enterprising Investor. Available at: https://blogs.cfainstitute.org/investor/2022/01/10/fama-and-french-the-five-factor-model-revisited/ (Accessed: 13 June 2024).

FasterCapital (2024) Fama french five factor model: An extended model for capital forecasting and Apt, FasterCapital. Available at: https://fastercapital.com/content/Fama-French-Five-Factor-Model--An-Extended-Model-for-Capital-Forecasting-and-APT.html (Accessed: 6 June 2024).

OneAdvanced. (n.d.). An introduction into financial analytics. [online] Available at: https://www.oneadvanced.com/news-and-opinion/an-introduction-into-financial-analytics/.(Accessed: 9 June 2024).

Stack Overflow (2024). Stack Overflow - Where Developers Learn, Share, & Build Careers. [online] Stack Overflow. Available at: https://stackoverflow.com/. (Accessed: 10 June 2024).

WBS (2024). WBS - Financial Analytics Capital Asset Pricing Model [online] Available at: https://my.wbs.ac.uk/$/$/$/event/cmsfile/t/item/i/1705621/v/1/f/0/n/Lecture_CAPM.pdf (Accessed: 10 June 2024).

Williams, E.E. and Dobelman, J.A. (2018). Quantitative financial analytics : the path to investment profits. New Jersey: World Scientific. (Accessed: 13 June 2024).

APPENDIX

Section 1 - Compairsion

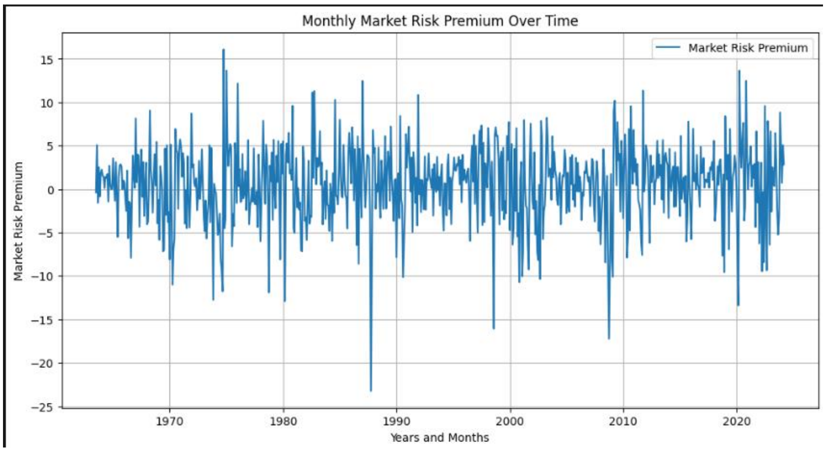
Dataset	Pros	Cons	Reason for Exclusion
Fama/French 3 Factors	- Includes Market, Size, and Value factors- Widely used and well-documented	- Lacks Profitability and Investment factors	- Limited factor coverage reduces the comprehensiveness of the analysis
Fama/French 3 Factors [Weekly/Daily]	- High-frequency data available- Suitable for detailed analysis	- Same as above: Lacks Profitability and Investment factors	- Limited factor coverage despite high frequency
Portfolios Formed on Size	- Focused on size-related returns- Detailed portfolio breakdowns	- Does not include other important factors like Value, Profitability, or Investment	- Too narrow in focus, limiting the scope of mean-variance analysis
Portfolios Formed on Book-to-Market	- Focused on value-related returns- Detailed portfolio breakdowns	- Does not include Size, Profitability, or Investment factors	- Too narrow in focus, limiting the scope of mean-variance analysis
Portfolios Formed on Operating Profitability	- Focused on profitability-related returns- Detailed portfolio breakdowns	- Does not include Size, Value, or Investment factors	- Too narrow in focus, limiting the scope of mean-variance analysis
Portfolios Formed on Investment	- Focused on investment-related returns- Detailed portfolio breakdowns	- Does not include Size, Value, or Profitability factors	- Too narrow in focus, limiting the scope of mean-variance analysis
6 Portfolios Formed on Size and Book-to-Market	- Includes Size and Value factors- Detailed portfolio breakdowns	- Lacks Profitability and Investment factors	- Limited factor coverage reduces the comprehensiveness of the analysis
25 Portfolios Formed on Size and Book-to-Market	- More granular size and value portfolios- Detailed portfolio breakdowns	- Lacks Profitability and Investment factors	- Limited factor coverage reduces the comprehensiveness of the analysis
100 Portfolios Formed on Size and Book-to-Market	- Highly granular size and value portfolios- Detailed portfolio breakdowns	- Lacks Profitability and Investment factors	- Limited factor coverage reduces the comprehensiveness of the analysis
6 Portfolios Formed on Size and Operating Profitability	- Includes Size and Profitability factors- Detailed portfolio	- Lacks Value and Investment factors	- Limited factor coverage reduces the comprehensiveness of the

	breakdowns		analysis
25 Portfolios Formed on Size and Operating Profitability	- More granular size and profitability portfolios- Detailed portfolio breakdowns	- Lacks Value and Investment factors	- Limited factor coverage reduces the comprehensiveness of the analysis
100 Portfolios Formed on Size and Operating Profitability	- Highly granular size and profitability portfolios- Detailed portfolio breakdowns	- Lacks Value and Investment factors	- Limited factor coverage reduces the comprehensiveness of the analysis
6 Portfolios Formed on Size and Investment	- Includes Size and Investment factors- Detailed portfolio breakdowns	- Lacks Value and Profitability factors	- Limited factor coverage reduces the comprehensiveness of the analysis
25 Portfolios Formed on Size and Investment	- More granular size and investment portfolios - Detailed portfolio breakdowns	- Lacks Value and Profitability factors	- Limited factor coverage reduces the comprehensiveness of the analysis
100 Portfolios Formed on Size and Investment	- Highly granular size and investment portfolios - Detailed portfolio breakdowns	- Lacks Value and Profitability factors	- Limited factor coverage reduces the comprehensiveness of the analysis
25 Portfolios Formed on Book-to-Market and Operating Profitability	- Includes Value and Profitability factors - Detailed portfolio breakdowns	- Lacks Size and Investment factors	- Limited factor coverage reduces the comprehensiveness of the analysis
25 Portfolios Formed on Book-to-Market and Investment	- Includes Value and Investment factors - Detailed portfolio breakdowns	- Lacks Size and Profitability factors	- Limited factor coverage reduces the comprehensiveness of the analysis
25 Portfolios Formed on Operating Profitability and Investment	- Includes Profitability and Investment factors - Detailed portfolio breakdowns	- Lacks Size and Value factors	- Limited factor coverage reduces the comprehensiveness of the analysis
32 Portfolios Formed on Size, Book-to-Market, and Operating Profitability	- Includes Size, Value, and Profitability factors - Detailed portfolio breakdowns	- Lacks Investment factor	- Missing Investment factor, which is important for a comprehensive mean-variance analysis
32 Portfolios Formed on Size, Book-to-Market, and Investment	- Includes Size, Value, and Investment factors - Detailed portfolio breakdowns	- Lacks Profitability factor	- Missing Profitability factor, which is important for a comprehensive mean-variance analysis
32 Portfolios Formed on Size, Operating Profitability, and Investment	- Includes Size, Profitability, and Investment factors - Detailed portfolio breakdowns	- Lacks Value factor	- Missing Value factor, which is important for a comprehensive mean-variance analysis

Fama/French 5 Factors (2x3)	- Includes Market, Size, Value, Profitability, and Investment factors - Comprehensive coverage - Available in multiple frequencies - Historical archives for back-testing	- None	- Provides the most comprehensive factor coverage for a robust and multifactor mean-variance analysis

Section 2

Estimation windows were chosen by analysing the graphs

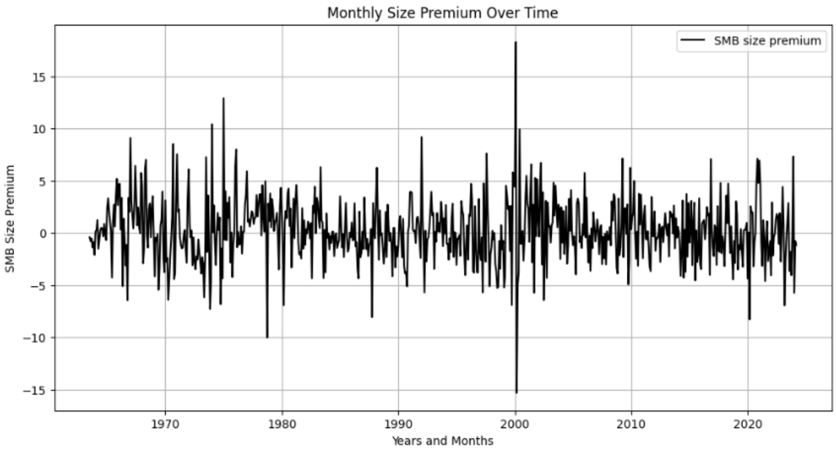


Graph 1 – monthly Market premium

60 Months (5 Years)

A 5-year window strikes a balance between short-term and long-term market conditions. It captures multiple business cycles and offers a sufficient number of observations to compute reliable statistical measures. The "Monthly Market Risk Premium Over Time" graph shows considerable volatility, which a 5-year window can smoothen to some extent, while still capturing recent market conditions.

Graph Reference: The period around 1970-1980 displays high volatility (Graph 1), which a 5-year window can help mitigate while preserving enough historical context.

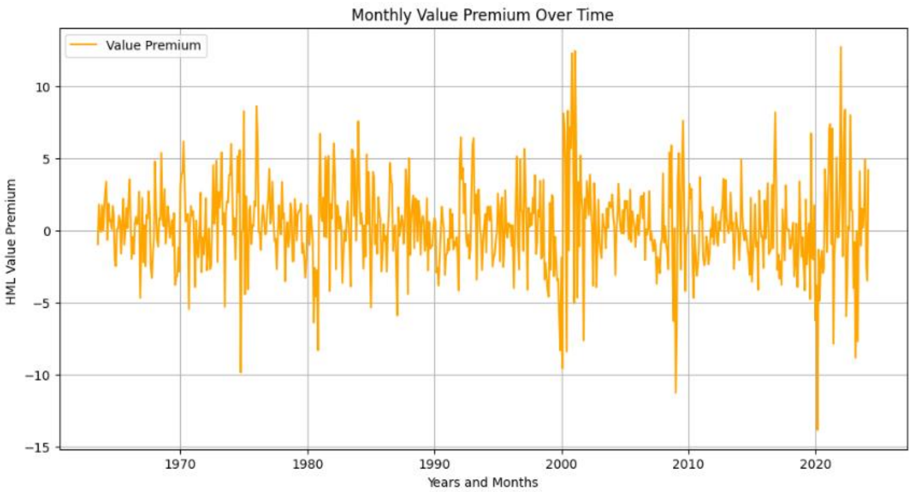


Graph 2- monthly size premium

120 Months (10 Years)

A 10-year window provides a broader view of the market, capturing long-term trends and reducing the impact of short-term fluctuations. This is particularly useful for understanding underlying market trends and the persistence of factors such as size and value premiums, as indicated in the "Monthly Size Premium Over Time" (Graph 2).

Graph Reference: The spikes in the size premium around the year 2000 (Graph 2) highlight significant market events. A 10-year window averages these spikes, providing a stable mean and covariance estimation.



Graph 3- Monthly value premium

180 Months (15 Years)

The 15-year window incorporates a long-term historical perspective, which is essential for understanding persistent market behaviour and factor trends. This window captures multiple market cycles and structural changes, ensuring that the estimates are robust against short-term noise.

Graph Reference: The "Monthly Value Premium Over Time" (Graph 3) shows periodic volatility across decades. A 15-year window effectively captures these long-term trends, as observed in the stable periods post-1980.

Proof of code from GPT

Question 3 – after multiple back and forth

At every time used in the previous step, compute the return obtained by the portfolio with weight 1/N for each of N available assets in the investment period from t to t+1.

Sure! We can compute the return for the portfolio with equal weights (1/N) for each available asset at each time step used in the previous optimization. Below is the code that includes this calculation:

python Copy code

```python
import pandas as pd
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt

# Load your data
file_path = 'c:\\Users\\Shariq\\OneDrive\\Desktop\\Warwick Business School\\Warwick Classe
df = pd.read_csv(file_path)

# Clean the Data
df['Years and Months'] = pd.to_datetime(df['Years and Months'], format='%Y%m')
df.set_index('Years and Months', inplace=True)
df.sort_index(ascending=True, inplace=True)

# Select only numeric data for rolling calculations
numeric_data = df.select_dtypes(include=[np.number])
numeric_data.columns = ['Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA', 'RF']

# Define multiple window sizes for rolling calculations
window_sizes = [60, 120, 180]  # Example windows in months

# Dictionary to store results for each window size
results = {}

for M in window_sizes:
    rolling_means = numeric_data.rolling(window=M, min_periods=M).mean().dropna()
    rolling_covariances = numeric_data.rolling(window=M, min_periods=M).cov(pairwise=True)

    results[M] = {
        'rolling_means': rolling_means,
        'rolling_covariances': rolling_covariances
    }

def portfolio_optimization(mean_returns, cov_matrix, target_return):
    num_assets = len(mean_returns)
    args = (mean_returns, cov_matrix)

    def portfolio_variance(weights, mean_returns, cov_matrix):
        return np.dot(weights.T, np.dot(cov_matrix, weights))
```

```python
    constraints = (
        {'type': 'eq', 'fun': lambda x: np.sum(x) - 1},
        {'type': 'eq', 'fun': lambda x: np.dot(x, mean_returns) - target_return}
    )
    bounds = tuple((0, 1) for asset in range(num_assets))
    result = minimize(portfolio_variance,
                      num_assets * [1. / num_assets],
                      args=args,
                      method='SLSQP',
                      bounds=bounds,
                      constraints=constraints)
    return result.x

# Example target return
target_return = 0.1  # Adjust based on your choice
optimization_results = {}
equal_weight_returns = {}

for M in window_sizes:
    rolling_means = results[M]['rolling_means']
    rolling_covariances = results[M]['rolling_covariances']

    # Aggregate the rolling means and covariances
    mean_returns = rolling_means.mean()
    cov_matrix = rolling_covariances.mean(level=0).cov()

    # Solve the optimization problem
    optimal_weights = portfolio_optimization(mean_returns, cov_matrix, target_return)

    portfolio_returns = []
    equal_weight_portfolio_returns = []
    N = len(mean_returns)
    equal_weights = np.full(N, 1/N)

    for t in range(len(rolling_means) - 1):
        # Compute the return obtained by the optimized portfolio in the investment period
        next_period_return = np.dot(optimal_weights, numeric_data.iloc[t + M][['Mkt-RF', '
        portfolio_returns.append(next_period_return)

        # Compute the return obtained by the equal weight portfolio in the investment peri
        equal_next_period_return = np.dot(equal_weights, numeric_data.iloc[t + M][['Mkt-RF
        equal_weight_portfolio_returns.append(equal_next_period_return)

    # Compute the portfolio's expected return and variance for the last period
    expected_return = np.dot(optimal_weights, mean_returns)
    portfolio_variance = np.dot(optimal_weights.T, np.dot(cov_matrix, optimal_weights))

    optimization_results[M] = {
```

Question 4– just removed the constraint, proof was on cursor but I do not have access to it as the software refreshed.

Question 5 – after multiple back and forth

To compute the naive portfolio return with equal weights (1/N for each asset) without using estimation windows, you can modify the code to directly calculate the mean and covariance for the entire dataset. Here's how to adjust the code accordingly:

python Copy code

```
import pandas as pd
import numpy as np

# Load your data
file_path = 'c:\\Users\\Shariq\\OneDrive\\Desktop\\Warwick Business School\\Warwick Classe
df_months_sorted = pd.read_csv(file_path)

# Convert 'Years and Months' to datetime if it's not already and set it as index
df_months_sorted['Years and Months'] = pd.to_datetime(df_months_sorted['Years and Months']
df_months_sorted.set_index('Years and Months', inplace=True)

# Ensure the DataFrame is sorted by date in ascending order
df_months_sorted.sort_index(ascending=True, inplace=True)

# Select only numeric data for calculations
numeric_data = df_months_sorted.select_dtypes(include=[np.number])
numeric_data.columns = ['Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA', 'RF']

# Define assets
assets = ['Mkt-RF', 'SMB', 'HML', 'RMW', 'CMA', 'RF']

# Calculate overall mean and covariance
mean_returns = numeric_data.mean()
cov_matrix = numeric_data.cov()

# Calculate naive portfolio
def naive_portfolio(mean_returns, cov_matrix, assets):
    num_assets = len(assets)
    equal_weights = np.array([1. / num_assets] * num_assets)

    expected_return = np.dot(equal_weights, mean_returns)
    portfolio_variance = np.dot(equal_weights.T, np.dot(cov_matrix, equal_weights))
    portfolio_std_dev = np.sqrt(portfolio_variance)

    return equal_weights, expected_return, portfolio_variance, portfolio_std_dev

# Calculate naive portfolio results
naive_equal_weights, naive_expected_return, naive_portfolio_variance, naive_portfolio_std_
    mean_returns[assets], cov_matrix.loc[assets, assets], assets
)

# Print the results
print("Equal Weights Portfolio:")
for asset, weight in zip(assets, naive_equal_weights):
```