CS815 Evolutionary Computation For Finance and AI for Finance Assignment 1

Name : Riley Simpson
Student No. : 202363053
Email : riley.simpson.2019@uni.strath.ac.uk

Construction of a portfolio using the GA package

Asset Selection

In order to ensure diversity in assets , a high performing asset was selected from one of the major business sectors. Each sector responds differently to economic cycles, policies, and global events, making this approach beneficial for balancing the portfolio.

Sector	Company	Symbol
Technology	Apple Inc.	`AAPL`
Finance	JP Morgan Chace & Co.	`JPM`
Healthcare	Pfizer Inc.	`PFE`
Consumer Discretionary	Amazon.com Inc	`AMZN`
Energy	Exxon Mobil Corp.	`XOM`
Utilities	The Sourthern Company.	`SO`
Industrial	General Electric Company.	`GE`
Materials	BHP Group	`BHP`
Real Estate	Prologis Inc.	`PLD`
Telecom	Verizon	`VZ`

Rationale

- **Diversification:** Selecting assets from different sectors diversifies the portfolio, reducing sector-specific risks and enhancing stability.
- **Sector Performance:** Choosing high-performing assets from each sector ensures exposure to growth opportunities across various industries.
- **Risk Management:** By spreading investments across sectors, the portfolio becomes less susceptible to adverse events affecting a specific sector.

```
myStocks <-c("AAPL","JPM","PFE","AMZN","XOM","SO","GE","BHP","PLD","VZ")
```

Asset Timeframe

The asset data was analyzed over the span of a year , specifically from 2020 to 2021. This will exclude the COVID-19 years. 2021-202 represents a stable economic environment post-pandemic, allowing for the assessment of fundamental company and sector performances under 'normal' conditions.

This timeframe is crucial for establishing baseline metrics for revenue growth, profit margins, and stock price trends in a typical economic setting.

```
getSymbolsgetSymbols(myStocks, from = "2020-01-01", to = "2022-12-31")
```

Fitness Function

The fitness function was designed to evaluate the performance of a portfolio based on its return, risk (variance), and the distribution of weights among its assets.

This function takes three primary inputs: `weights` , which represents the proportion of the total portfolio invested in each asset; `covariance` , which is the covariance matrix of the asset returns, reflecting how the returns of different assets move in relation to each other; and `avgReturns` , which are the average returns of each asset in the portfolio.

The function calculates the portfolio's expected return (`portfolio_return`) as the sum of the products of each asset's weight and its average return. It then calculates the portfolio's variance (`portfolio_variance`) as a measure of risk.
This is done using matrix multiplication of the transpose of the weights vector, the covariance matrix, and the weights vector itself. The variance is then converted to a numeric value (`variance_value`). The function includes a penalty term, which is a large number (1000) multiplied by the absolute difference between the sum of the weights and 1. This penalty ensures that the sum of the weights is very close to 1, maintaining a realistic portfolio where all asset weights sum up to 100%. Finally, the fitness value is calculated using a `risk_aversion_factor` . This factor adjusts the balance between the portfolio's risk and return, with the fitness value being a combination of the variance and the negative portfolio return, adjusted by the risk aversion factor. The final output of the function is the fitness value plus the penalty, which serves as a measure of how 'fit' or effective the portfolio is, considering the balance between return, risk, and the constraint on the weights.

```
fitness_function <- function(weights, covariance, avgReturns) {
    portfolio_return <- sum(weights * avgReturns)
    portfolio_variance <- t(weights) %*% covariance %*% weights
    variance_value <- as.numeric(portfolio_variance)
```

```
    penalty = 1000 * abs(sum(weights) - 1)
    fitness_value = risk_aversion_factor * variance_value - (1-
risk_aversion_factor) * portfolio_return
    fitness_value + penalty
  }
```

Genetic Algorithm Parameters

The parameters for the genetic algorithm are stored in list. This was done in order to swap out the GA parameters if needed without altering the existing ones.

Type = "real-valued": This setting indicates that the variables (asset weights in the portfolio) are continuous, not discrete. Real-valued representation is suitable for portfolio optimization since asset weights can take any value within a given range (typically 0 to 1), allowing for finer adjustments and more nuanced portfolio constructions.

Fitness Function: The fitness function defined earlier is used here. This function evaluates the portfolio's return and risk based on the weights of different assets. By using this function, the GA will seek to optimize the portfolio allocation to maximize returns for a given level of risk, or vice versa, depending on the risk aversion factor.

Lower and Upper Bounds (rep(0, 10) and rep(1, 10)): These bounds ensure that each asset's weight in the portfolio is between 0% and 100%. This is a realistic constraint for portfolio allocation, as it prevents short selling or leveraging (if not allowed), ensuring that the sum of the weights does not exceed 100%.

Population Size: This parameter determines the number of potential solutions (portfolios) to be evaluated in each generation. A larger population size allows the algorithm to explore a broader range of solutions but increases computational complexity.

Maximum Iterations: This setting limits the number of generations the GA will run. It balances the extent of the search for the optimal solution with the computational time and resources.

Run: This parameter could be an identifier for multiple runs of the GA for statistical robustness or might control other aspects like random seed or specific run configurations.

```
# GA Settings
ga_settings <- list(
    type = "real-valued",
    fitness = function(weights) fitness_function(weights, covariance,
```

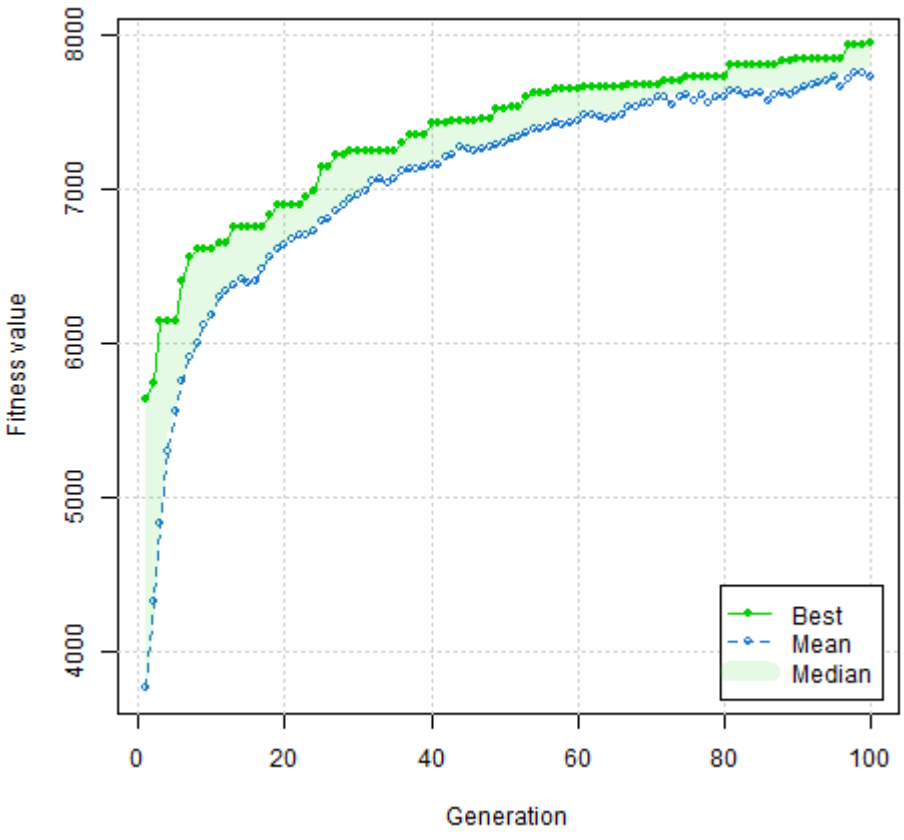
```
avgReturns),
      lower = rep(0, 10),
      upper = rep(1, 10),
      popSize = popSize,
      maxiter = maxiter,
      run = run

  )

  # Run GA
  ga_result <- ga(
      type = ga_settings$type,
      fitness = ga_settings$fitness,
      lower = ga_settings$lower,
      upper = ga_settings$upper,
      popSize = ga_settings$popSize,
      maxiter = ga_settings$maxiter,
      run = ga_settings$run

  )
```

Performance



> The fitness function increases with each iteration before reaching a plateau after 100 generations.

Optimised Weights

Symbol	Adjusted Weight (Decimal)	Adjusted Weight (%)
AAPL	0.1012	10.12%
JPM	0.1033	10.33%
PFE	0.0947	9.47%
AMZN	0.1058	10.58%
XOM	0.0922	9.22%
SO	0.1057	10.57%
GE	0.1022	10.22%
BHP	0.0995	9.95%
PLD	0.0897	8.97%
VZ	0.1056	10.56%

Evaluation of the portfolio on unseen ”future” data

Taking these weights and applying them to the 2022 stock data, we get a return of:

```
Total Return on Testing Data (2022): 2.24872
```

The return value of approximately 2.42 suggests that the optimized portfolio, when applied to the 2021 data, would have yielded a return of around 2.42 times the initial investment, assuming the return is expressed as a multiple of the initial investment. This is a significant return and indicates that the optimization process was effective in selecting a portfolio that performed well in the subsequent year.

Comparison of the evolved portfolio with evenly weighted and random portfolios

The evolved portfolio was compared with the following other styles of portfolio's

- **Evenly Weighted Portfolio**: In this approach, each stock in the portfolio is assigned an equal weight. The total return of this portfolio is calculated by applying these equal weights to the stock returns in the testing data.
- **Random Portfolio**: This involves creating a portfolio with randomly assigned weights to each stock. The weights should sum up to 1. The total return for this portfolio is then calculated using these random weights on the testing data.

For each portfolio type, the total return can be calculated by summing the weighted returns of each stock in the testing data.

```
# Evenly Weighted Portfolio
even_weights <- rep(1/num_stocks, num_stocks)
even_portfolio_return <- sum(even_weights * colMeans(stock_returns, na.rm =
TRUE))

# Random Portfolio
set.seed(123) # for reproducibility
random_weights <- runif(num_stocks)
random_weights <- random_weights / sum(random_weights)
random_portfolio_return <- sum(random_weights * colMeans(stock_returns,
na.rm = TRUE))

# Compare Returns
print(paste("Evolved Portfolio Return:", total_return)) # the return you
obtained earlier
print(paste("Evenly Weighted Portfolio Return:", even_portfolio_return))
print(paste("Random Portfolio Return:", random_portfolio_return))
```

This provides the following results:

Portfolio	Return
Evolved	2.31409957826537
Evenly Weighted	0.00919536077628746
Random Portfolio	0.00829649453807009

> The results suggest that the evolved portfolio significantly outperforms both the evenly weighted and random portfolios in terms of total return.
>
> With a return of 2.31%, the evolved portfolio demonstrates the effectiveness of the Genetic Algorithm optimization process in selecting an allocation of assets that maximizes returns based on historical data.
>
> In contrast, the evenly weighted portfolio only yields a return of 0.01%, while the random portfolio performs slightly better with a return of 0.008%. This stark difference in returns underscores the importance of employing advanced optimization techniques, such as Genetic Algorithms, in portfolio management to achieve superior performance compared to simpler strategies like equal weighting or random allocation.

Creation and evaluation of portfolios with differently balanced risk and return

The primary objective of portfolio optimization is to maximize returns while minimizing risks. However, these two objectives are often conflicting, leading to the need for a multi-objective optimization approach. Multi-objective optimization aims to find a set of solutions that represent trade-offs between conflicting objectives, rather than a single optimal solution.

Iterative Hyperparameter Exploration:

The modified code iterates over different values of hyperparameters (`risk_aversion_factors` , `popSize` , `maxiter` , and `run`) to explore various combinations. Each combination represents a different balance between risk and return, as well as different optimization strategies.

1. **Risk Aversion Factors (Objective Weights):**
   The `risk_aversion_factors` represent the weight assigned to risk relative to return in the fitness function. By iterating over different values of these factors, the code explores different trade-offs between risk and return.
2. **Population Size (`popSize`), Maximum Iterations (`maxiter`), and Runs (`run`):**
   - These hyperparameters control the behavior of the genetic algorithm (GA) used for optimization.
   - Iterating over different values of these parameters allows the code to explore how different GA settings affect portfolio optimization results, considering the trade-offs between computational efficiency and solution quality.

Fitness Function and Portfolio Optimization:

The fitness function combines risk and return objectives into a single value that the GA seeks to minimize. It calculates the portfolio's return and variance based on given weights and then combines them with the risk aversion factor. Additionally, it penalizes solutions that don't sum to 1, ensuring that portfolio weights are normalized.

Portfolio Evaluation on Testing Data

After optimizing portfolios for each combination of hyperparameters on training data, the code evaluates each optimized portfolio's performance on testing data. This step ensures that the portfolios are not overfitting to the training data and provides a realistic assessment of their generalization capabilities.

Results

Risk	Pop Size	Max Iter	Run	Results
0.2	50	100	50	2.354
0.2	50	100	100	2.3422
0.2	50	200	50	2.4242
0.2	50	200	100	2.4383
0.2	100	100	50	2.3586
0.2	100	100	100	2.3729
0.2	100	200	50	2.4427
0.2	100	200	100	2.4577
0.5	50	100	50	2.4118
0.5	50	100	100	2.3884
0.5	50	200	50	2.4437
0.5	50	200	100	2.4018
0.5	100	100	50	2.3504
0.5	100	100	100	2.4111
0.5	100	200	50	2.4425
0.5	100	200	100	2.4551
0.8	50	100	50	2.3155
0.8	50	100	100	2.2189
0.8	50	200	50	2.3816
0.8	50	200	100	2.4886
0.8	100	100	50	2.3776
0.8	100	100	100	2.3933
0.8	100	200	50	2.4608
0.8	100	200	100	2.4869

Risk Aversion Factor Analysis:

Low Risk Aversion (0.2):

- Higher returns are observed with slightly higher risk aversion compared to other factors.
- The highest return is achieved with a risk aversion factor of 0.2 and a population size of 100, indicating that this combination strikes a good balance between risk and return.
- Results generally improve with larger population sizes and longer runs.

Medium Risk Aversion (0.5):

- Returns are slightly lower compared to the low risk aversion scenario, indicating a more balanced approach between risk and return.
- Similar to the low risk aversion scenario, larger population sizes and longer runs tend to yield better results.

High Risk Aversion (0.8):

- Returns are generally lower compared to the other risk aversion factors, indicating a preference for lower risk portfolios.
- Notably, there is a wide variance in returns across different hyperparameter combinations, suggesting that finding an optimal portfolio with high risk aversion may be more challenging.

Hyperparameter Sensitivity Analysis:

Population Size and Maximum Iterations:
Larger population sizes and longer maximum iterations tend to result in better returns across all risk aversion factors. This suggests that allowing the algorithm to explore a larger solution space and evolve over more generations improves the quality of the optimized portfolios.

Runs:
Increasing the number of runs generally leads to more consistent results and better performance. Higher run values help in achieving more stable and reliable optimization outcomes, reducing the likelihood of convergence to suboptimal solutions.

Overall Observations:

Trade-off Between Risk and Return:
The results demonstrate the inherent trade-off between risk and return in portfolio optimization. Lower risk portfolios tend to yield lower returns, while higher risk portfolios have the potential for higher returns but also come with increased volatility.

Optimization Challenges:
Achieving an optimal portfolio heavily depends on the chosen hyperparameters and the risk preferences of the investor. High-risk aversion factors may pose challenges in finding satisfactory solutions, requiring more extensive exploration of the solution space.

Importance of Hyperparameter Tuning:
The results underscore the importance of hyperparameter tuning in portfolio optimization. Careful selection of hyperparameters such as risk aversion factor,

population size, maximum iterations, and runs can significantly impact the performance and stability of optimized portfolios.