CS185 - Evolutionary Computing/AI for Finance
Assignment 1
03-03-2023
20227413

Part 1

Construction of a portfolio using the GA package To compose the portfolio, the assets have been selected at random from the FTSE Top 350. The stocks are Apple, American Express, British Airways, JD Sports and Fashion, Shell PLC, British Petroleum, Domino's Pizza, Britvic, J.D Wetherspoon, and Experian respectively.

```
myStocks <- c("AAPL", "AXP", "BA", "JD",  "SHEL", "BP", "DPZ", "BVIC.L", "JDW.L", "EXPN.L")
```

Obtaining financial data from 01-01-2020 - 01-01-2022.

```
getSymbols(myStocks, src="yahoo", from="2020-01-01", to="2022-01-01")
```

Creating a DF for the daily returns

```
myRetData <- data.frame(merge(dailyReturn(AAPL), dailyReturn(AXP), dailyReturn(BA), dailyReturn(JD), da
colnames(myRetData) <- myStocks
```

There are missing values in the data. The percentage of missing values is checked below:

```
(colMeans(is.na(myRetData)))*100
```

We can see that NA values total between 1.7% to 2.1% of the values. The missing values are replaced with 0.

```
myRetData[is.na(myRetData)] <- 0
```

Calculating the mean daily returns and annual returns.

```
meanDailyReturns <- apply(myRetData, 2, mean)
annualReturns <- (meanDailyReturns + 1)^252 - 1
sd(annualReturns)
```

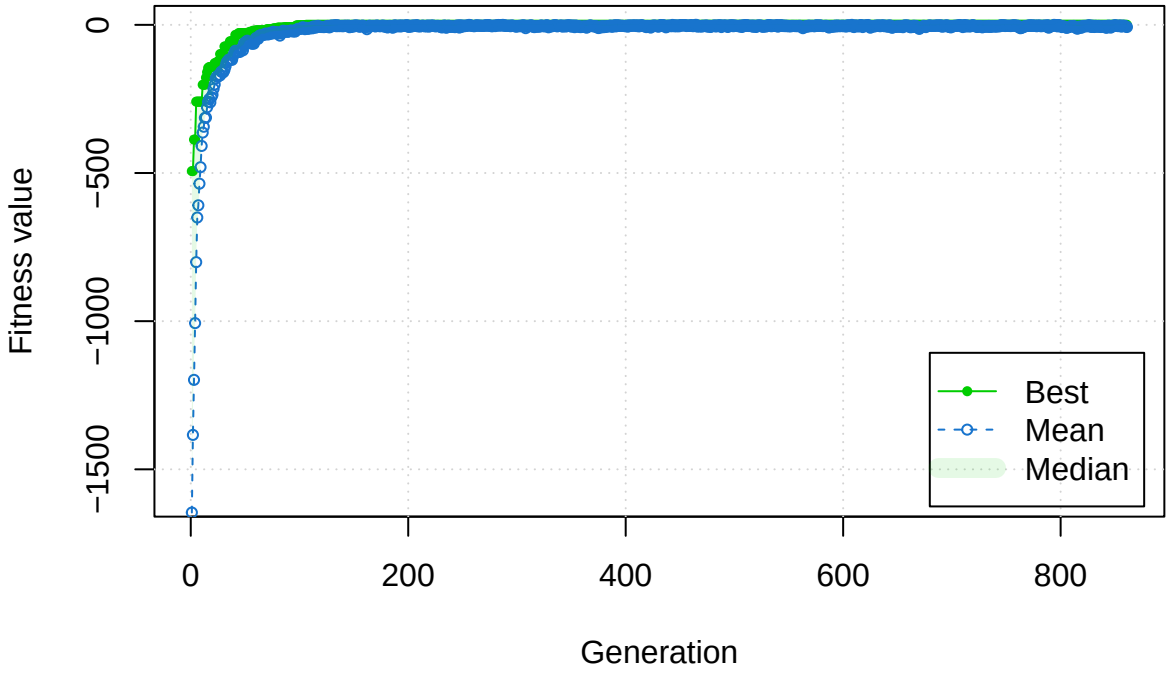
Using the GA package to identify an optimal set of weights
Having found the returns of the assets, we will now use the GA to identify the optimal weights for the portfolio.To do this, the Sharpe Ratio has been used to calculate the expected returns divided by the variance squared. A penalty function lowers the weight of the assets that aren't suitable to the multi-objective optimization goal.

Genetic algorithms are an end-to-end machine learning model based on the rules of natural evolution and selection. For an algorithm to be considered genetic, three features must be implemented:

- The model must be able to select from a population of chromosomes that can be chosen as solutions to a given problem.
- Ability to choose chromosomes competitively based on the fitness of said chromosomes, and how well they achieve the goal (minimizing risk whilst maximizing return)

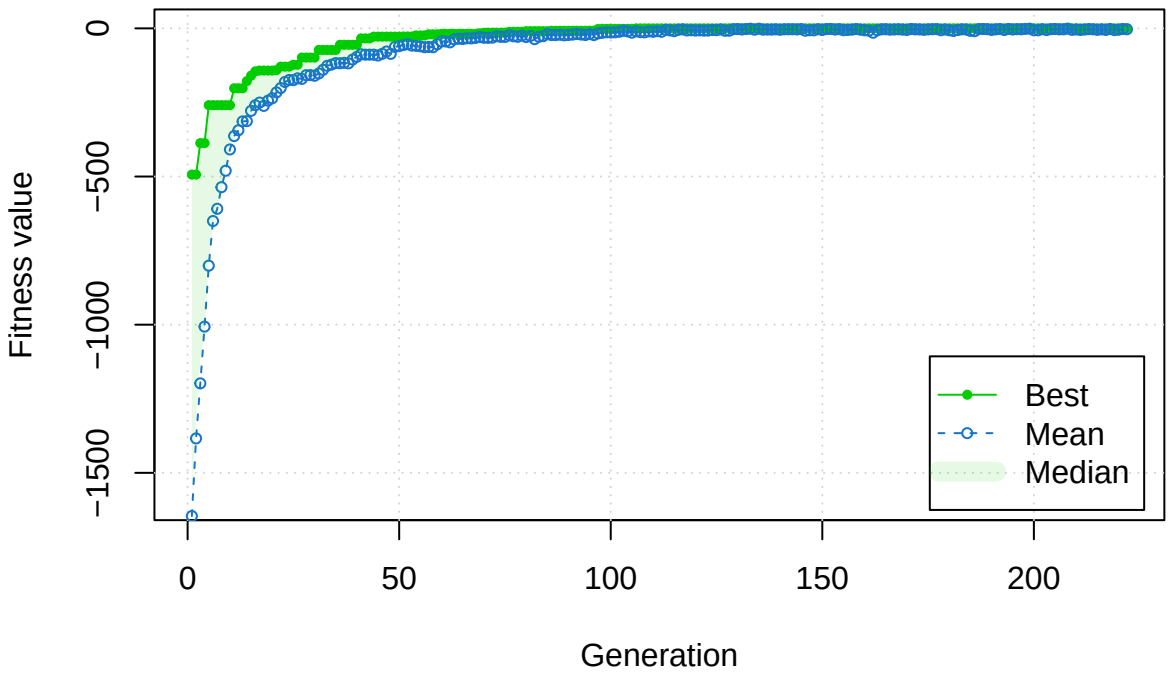
- Ability to alter the chromosomes genetic components in order to create new chromosomes for further testing.

The output below combines the above defined functions for risk, return, and weights with the penalty function in a genetic algorithm.

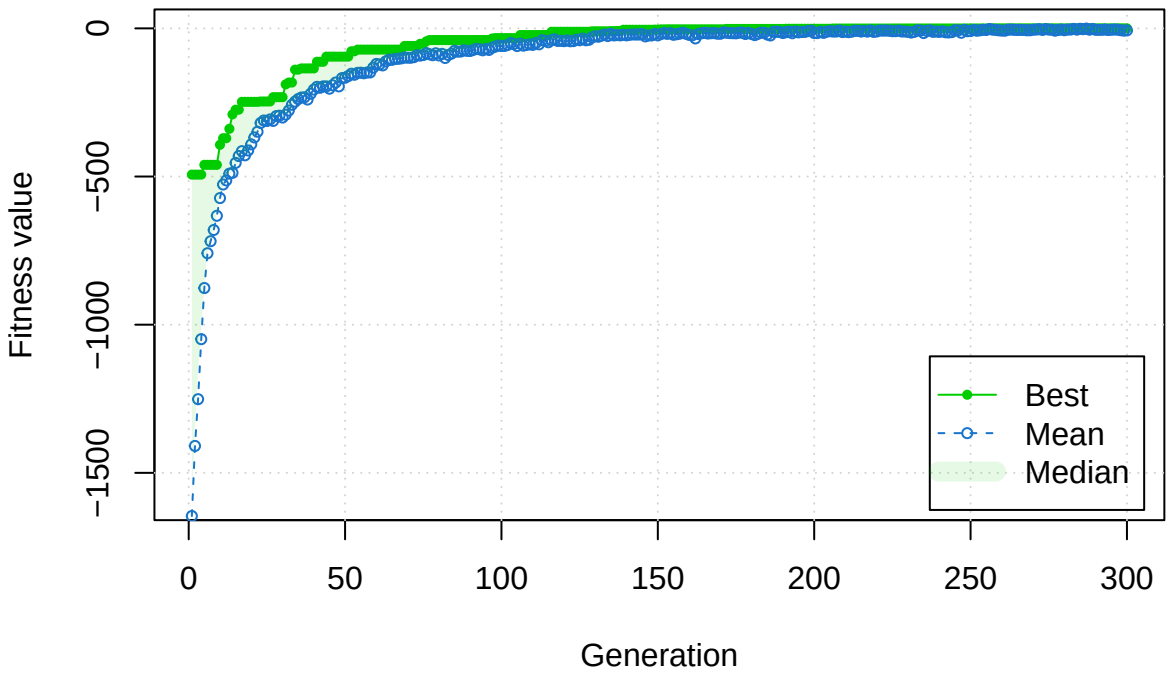


The above plot demonstrates the fitness value of GA1. The plot reaches a plateau well before the specified number of iterations. The following parameters are implemented into the second version of the GA:

maxiter=50000
run=50



GA2 still demonstrates premature convergence. The 'elitism' parameter is introduced below, as well as reduced 'maxiter' and 'run' parameters:



Storing the optimal weights into a vector:

```
optimalWeights = as.vector(summary(GA3)$solution)
```

Observations

GA3 shows that with reduced 'maxiter' and 'run' values and the introduced 'Elitism' parameter doesn't better the plateau and continues to cycle through between 50-100 generations to find the optimal set of weights. It's possible that further adjusting the GA's crossover and mutation parameters may further better the result.

Evaluation of the portfolio on unforeseen future data Splitting the stocks into a train and test set.
2020-01-01 - 2021-01-01 - Train set
2021-01-02 - 2022-01-01 - Test Set

Splitting the data into a test and train set using the above dates.

```
testStocks <- c("AAPL", "AXP", "BA", "JD",  "SHEL", "BP", "DPZ", "BVIC.L", "JDW.L", "EXPN.L")
getSymbols(testStocks, src="yahoo", from="2021-01-02", to="2022-01-01")
testRetData <- data.frame(merge(dailyReturn(AAPL),dailyReturn(AXP),dailyReturn(BA),dailyReturn(JD), dai
colnames(testRetData) <- testStocks
testRetData[is.na(testRetData)] <- 0
meanTestData <- apply(testRetData, 2, mean)
```

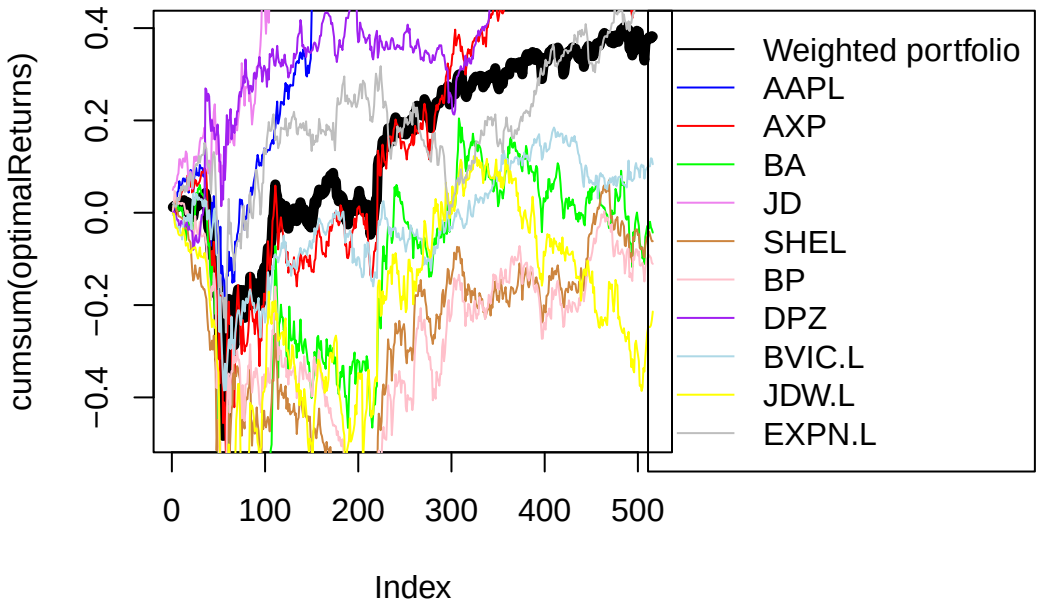
Calculating the performance of the weights based on the test and train data.

```
portReturnsTrain <- sum(optimalWeights*meanDailyReturns)
portReturnsTest <- sum(optimalWeights*meanTestData)
portReturnsTrain
portReturnsTest
optimalWeights <- sort(optimalWeights, decreasing=TRUE)
cbind(colnames(testRetData), optimalWeights)
```

```
optimalSD <- apply(testRetData, 1, function(x) sum(x * optimalWeights))
sd(optimalSD)
```

```
## [1] 0.01075124
```

Figure 1. Train Portfolio Returns



Observations

The train portfolio with the evolved weights shows a small positive increase in returns compared to the train portfolio (0.0006901542 > 0.0006573444), which suggests an aggressive portfolio may be worse performing. The assets with the most weightage are Apple, American Express and Boeing. JD Sports and Shell are also heavily weighted, which is suprising given their performance in the train data as seen in Figure 1. Similarly, BP and JDW have low weightings despite performing well in the train data. However, it is worth acknowledging that the nature of these assets have meant that trading performance was low during the first wave of COVID-19 restrictions (as seen in the 50 day mark in the index), and therefore, will impact the calculations made by the GA when deciding the optimal weights. It could be argued then, by comparing the optimal weights and Figure 1, that the the GA takes into account how quickly an asset recovers in terms of performance when deciding optimal weights.

The standard deviation of the returns on the test data is extremely low (0.01), indicating that the weights have minimized risk.

Therefore, it is reasonable to conclude that the test weights achieve the multi-objective goal of minimizing risk and maximizing return for this selection of assets. In the next section, different emphasis is given to risk and returns to recreate aggressive and conservative portfolios to futher determine if the weights pertain to the multiobjective goal.

Comparison of the evolved portfolio with balanced and random portfolios Having now found our optimal weights, a comparison will be made between a balanced portfolio using balanced weights, and a random portfolio using random weights. Different emphasis on risk and return will be explored and compared to the original optimal portfolio.

```
balancedWeights <- rep(0.1, 10)
balancedTrain <- sum(balancedWeights*meanDailyReturns)
balancedTest <- sum(balancedWeights*meanTestData)
balancedSD <- apply(testRetData, 1, function(x) sum(x * balancedWeights))

randWeights <- runif(n = length(myStocks))
randWeights <- randWeights/sum(randWeights)
sum(randWeights)
randomSD <- apply(testRetData, 1, function(x) sum(x * randWeights))

randTrain <- sum(randWeights*meanDailyReturns)
randTest <- sum(randWeights*meanTestData)

balancedTest
randTest
portReturnsTest

sd(balancedSD)
sd(randomSD)
sd(optimalSD)
```

```
cat("The returns of the balanced portfolio are", balancedTest, "with a SD of", sd(balancedSD),"\n")
```

```
## The returns of the balanced portfolio are 0.0006996564 with a SD of 0.01010663
```

```
cat("The returns of the random portfolio are", randTest, "with a SD of", sd(randomSD),"\n")
```

```
## The returns of the random portfolio are 0.0009800862 with a SD of 0.009389525
```

```
cat("The returns of the optimal portfolio are", portReturnsTest, "with a SD of", sd(optimalSD),"\n")
```

```
## The returns of the optimal portfolio are 0.0006901542 with a SD of 0.01075124
```

Observations

The balanced portfolio yields much higher returns than the random portfolio ($0.0006996564 > 0.0006516931$), but still underperforms the optimal weights portfolio (0.0007020059). This is consistent with the findings in the prior sections, in that there is a noticeable difference in returns and the GA assigned weights. The random portfolio has the highest standard deviation($0.0112572 > 0.01075124 > 0.01075124$) and the lowest returns, making this the least optimal portfolio. The optimal portfolio has the highest returns and slightly

higher standard deviation indicating more volatility than the balanced portfolio but given the small difference in SD scores, the optimal portfolio may remain optimal.

Creation and evaluation of portfolios with differently balanced risk and return

In order to create GA's with differently balanced risks and returns, we need to modify the Sharpe ratio used in GA3. There are two aggressive and two conservative portfolios, and an equally balanced risk and return portfolio. The GA uses the optimal parameters found in Part 1 (GA3).
Aggressive = 90% risk, 10% return.

Aggressive = 75% risk, 25% return.

Balanced = 50% risk, 50% return

Conservative = 35% risk, 65% return

Conservative = 15% risk, 85% return.

```
cat("The returns of the 90% risk portfolio are", GA4Returns, "with a SD of", sd(GA4SD),"\n")
```

```
## The returns of the 90% risk portfolio are 0.0006187609 with a SD of 0.01804983
```

```
cat("The returns of the 75% risk portfolio are", GA5Returns , "with a SD of", sd(GA5SD),"\n")
```

```
## The returns of the 75% risk portfolio are 0.0007164968 with a SD of 0.01930101
```

```
cat("The returns of the 50% risk portfolio are", GA6Returns , "with a SD of", sd(GA6SD),"\n")
```

```
## The returns of the 50% risk portfolio are 0.0007351935 with a SD of 0.01909915
```

```
cat("The returns of the 35% risk portfolio are", GA7Returns , "with a SD of", sd(GA7SD),"\n")
```

```
## The returns of the 35% risk portfolio are 0.0008734264 with a SD of 0.01653106
```

```
cat("The returns of the 15% risk portfolio are", GA8Returns , "with a SD of", sd(GA8SD),"\n")
```

```
## The returns of the 15% risk portfolio are 0.0009144243 with a SD of 0.01579461
```

Observations
Unsurprisingly, the aggressive portfolios do not perform as well as the conservative portfolios, confirming the trend found in the optimal weights. Again, this is most likely due to the number of low performing stocks as observed in Figure 1. The difference in returns between the 50% and 75% risk portfolios is minimal (0.0007164968; 0.0007351935), meaning that increasing the portfolio risk of 25% creates a marginal increase in returns. In conclusion, the conservative portfolios work best with the current selection of assets. Similarly, the SD of the returns steadily decreases with less risk, indicating that a conservative portfolio with less risk is more stable in the long term. In concluysion, it would appear thgat the 35% risk portfolio is most suitable for the multi-objective goal.

Part 2 - Using GAs to select the assets In addition to the 10 previously selected assets, 40 more assets were pooled for the GA to determine the the most profitable assets from the total pool of 50, and if the previous returns could be improved upon.

```
myStocks2 <-c("AMZN", "GOOGL", "MSFT", "TSLA", "META", "KO","PEP", "JPM", "BAC", "GE","JNJ", "MRK", "PF
getSymbols(myStocks2, src="yahoo", from="2020-01-01", to="2022-01-01")
```

```
myRetData2 <- data.frame(merge(dailyReturn(AMZN), dailyReturn (GOOGL), dailyReturn(MSFT), dailyReturn(T

colnames(myRetData2) <- myStocks2
meanDailyReturns2 <- apply(myRetData2, 2, mean)
```

Modifying the fitness function previously defined in Part 1.

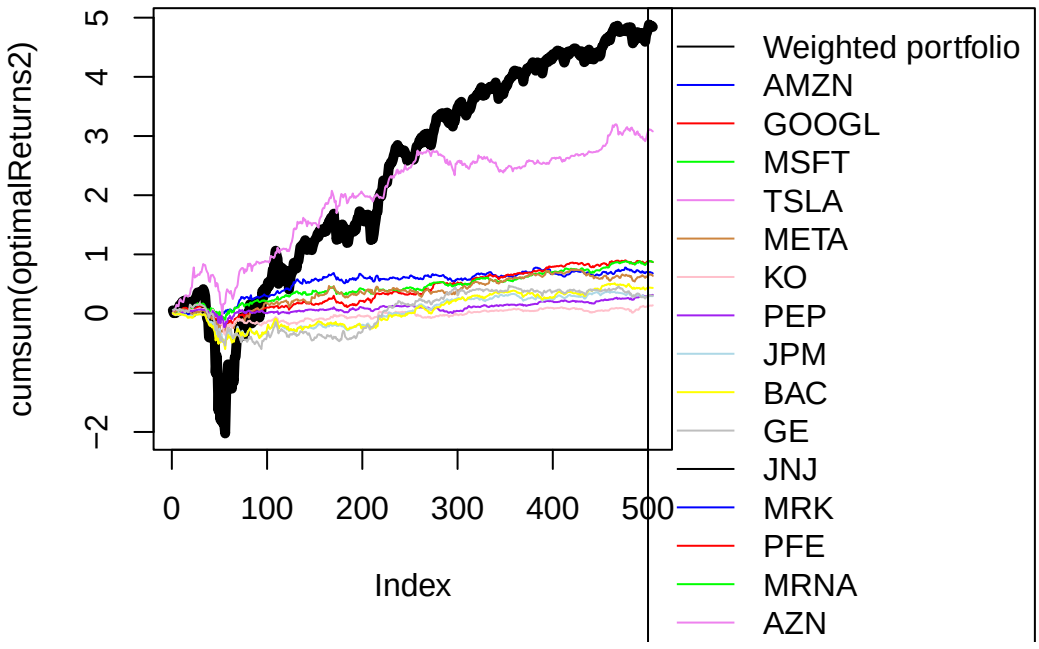
Processing the 50 new stocks through the GA.

```
GA2.1 <- ga(type = "real-valued", fitness = function(x){-objFunction2(x)}, lower = rep(0, ncol(myRetData
```

```
optimalWeights2 = as.vector(summary(GA2.1)$solution)
optimalWeights2
```

```
optimalWeights2DF <- data.frame(group = myStocks2, value = optimalWeights2)
optimalWeights2DF <- optimalWeights2DF[order(optimalWeights2DF$value, decreasing = TRUE), ]
optimalWeights2DF <- Reduce(rbind, by(optimalWeights2DF, optimalWeights2DF["group"], head, n = 3))
```

Figure 1. GA Assest Portfolio Returns



Final observations and summary In conclusion, using the GA to find the optimal weights of this portfolio proved sucessful. Finetuning the fitness function further may lead to more optimal weights. It

would appear from the results that the GA takes into account how quickly an assest recovers when evolving weights. A balanced portfolio has a slightly lower standard deviation than the optimal portfolio indicating more stability, but perhaps the optimal portfolio should still be used. Differing risk and return portfolios further reiterate that a conservative portfolio performs better.

References DD (2018) Portfolio Optimization in R, coding-finance. Available at: https://www.codingfinance.com/post/2018-05-31-portfolio-opt-in-r/ (Accessed: March 2, 2023).

Grefenstette, J.J. (1993) "Genetic algorithms and machine learning," Proceedings of the sixth annual conference on Computational learning theory - COLT '93 [Preprint]. Available at: https://doi.org/10.1145/168304.168305.

Malato, G. (2018) Portfolio optimization in R using a genetic algorithm, Medium. The Trading Scientist. Available at: https://medium.com/the-trading-scientist/portfolio-optimization-in-r-using-a-genetic-algorithm-8726ec985b6f (Accessed: March 2, 2023).