Genetic Algorithm Approach for Portfolio Optimization

Mihail Mihaylov

Reg. No: 201888925

Introduction

This assignment involves optimising a portfolio by calculating the risk and the return values from a selected list of assets and determining the best way to invest in those assets by taking into consideration the risk and the return values of the assets in the portfolio. This has been done by implementing a multi-objective genetic algorithm using R studio. Since the available package in the IDE does not support multi-objective fitness function it is necessary that the fitness function is combined into a single fitness function and the solutions are evaluated using that. This report contains detailed information on the assets included in the portfolio as well as the details over the implementation of the fitness function and the genetic algorithm itself as well as the way the solutions are presented. Another part of this assignment involves testing the developed genetic algorithm solutions against future data to see how the model performs on unseen data and compare the results from the training data against other algorithms such as the Random Search Algorithm.

Choice of assets

The initial list of assets contained only 4 assets. Those assets were used to create the initial model and get the genetic algorithm running. Those assets were – AKER, BLDP, GOOG and AAPL. Those are the tickers for Aker Solutions, Ballard Power Systems, Google and Apple. After the genetic algorithm was set and run, there were 2 portfolios considered. One of them was of 10 different European banks including Royal Bank of Scotland, HSBC, Barclays, Lloyds, Deutsche Bank, Banco Bilbao, Santander, Credit Suisse Group, UBS Group and ING. The data collected for those banks was the weekly returns from 1st January 2018 to the 1st of January 2019. After running the genetic algorithm, it was determined that this is not a very good choice of assets so another portfolio would need to be used to get a good return for a low risk. The next portfolio considered was for Tech Companies with positive weekly returns for the same time period (1st January 2018 to 1st January 2019). The tech companies included in the portfolio are 21Vianet Group Inc (VNET), 3D Systems Corporation (DDD), 51Job Inc. (JOBS), Alphabet Inc (GOOG), 8x8 Inc. (EGHT), Aaron's Inc. (AAN), Acacia Communications (ACIA), ACI Worldwide (ACIW), Adobe (ADBE) and Advanced Micro Devices (AMD). Using these assets in a portfolio gave much better results and provided a nice example of the pareto plot which can be seen in the Results section of this report.

The data for the weekly returns for this portfolio optimization was collected using a package called "quantmod". The source of the data is "yahoo".

Details of the GA

After the weekly returns for those assets were collected and stored into a matrix, using a method called colMeans in RStudio the mean for each of the assets return over that year was calculated and using the cov method a covariance matrix was created.

weekly.returns	weekly.returns.1	weekly.returns.2	weekly.returns.3	weekly.returns.4	weekly.returns.5
0.0052070517	0.0080056128	0.0024850419	0.0004626212	0.0058884827	0.0018702594
weekly.returns.6	weekly.returns.7	weekly.returns.8	weekly.returns.9	weekly.returns.10	
0.0042573152	0.0048528733	0.0217737238	0.0055168432	0.0143870242	

Figure 1: Mean Weekly Returns for the Tech Companies Portfolio.

From there 4 functions were implemented. One was for calculating the return for each solution, which was represented by a vector of weights which sum up to 1. This was done by multiplying that vector by the returns vector calculated before and summing up the values in that vector. Another one was for calculating the risk for each solution. This was achieved by multiplying the covariance matrix by the vector solution then transposing the resulting matrix and multiplying it by the vector solution again. Doing that ensured that each covariance was multiplied by the weight of the assets involved in that covariance. After that the risk value was calculated by summing up all the values in the matrix.

	weekly.returns	weekly.returns.1	weekly.returns.2	weekly.returns.3
weekly.returns	0.008639306	0.0016878452	0.0014567328	0.0003674570
weekly.returns.1	0.001687845	0.0116821333	0.0003824922	0.0013071281
weekly.returns.2	0.001456733	0.0003824922	0.0043031178	0.0009434057
weekly.returns.3	0.000367457	0.0013071281	0.0009434057	0.0014034381

Figure 2: An example of a covariance matrix. Each weekly.return represents an asset.

Since the solution vectors, which were given as an input to those 2 methods, had to sum up to 1 there was a need for a method to scale down the weight vectors. This was done by a method which divides each value from the vector solution by the sum of the vector solution. Once the 2 fitness functions for risk and return were implemented, it was necessary to implement a fitness function which would take into account both and return a single fitness value, because the GA package implemented in RStudio does not support multi-object optimisation. This fitness function had to be implemented in such a way that it would make the GA maximise the return value and minimise the risk value in order to get a better solution. Since the ga method in the GA package is trying to maximise the fitness function the following formula was used:

*fitness <- (w*maxReturn(x))+((1-w)*(-minRisk(x)))*

In this formula the w variable is the ratio between risk and return thus giving the possibility to calculate different solutions whether the focus of the GA is to find a higher return or a lower risk solution. The x value is a solution vector provided by the genetic algorithm. And maxReturn() and minRisk() are the 2 methods calculating the return and the risk values respectively. In order for the genetic algorithm to try and minimize the risk value that value was negated.

The genetic algorithm implemented uses a method called ga in the GA package. This required a few key variables in order to work properly. First it required a type of solution it needs to look for. In this case the type is real-valued. Another requirement is a fitness function which was already explained above. In order to make sure the ga created proper chromosomes it required a lower and upper limits which in this case are represented by vectors of 0s and 1s respectively. In order to make sure that the solutions are the same size as the number of stocks the vectors for upper and lower limit were created to be the same size as the number of stocks in the portfolio. For this assignment the values for crossover and mutation were left by default which is 0.8 and 0.1 respectively. The number of iterations was set to 200, the size of the population to 250 and the run value was set to 50. The latter means that if the genetic algorithm doesn't find a different maximum fitness value solution in 50 iterations it should stop and assume that this is the best value. Other additional useful arguments which were given to the ga method were the parallel and monitor arguments. Installing 2 packages called parallel and doParallel and setting the parallel value to TRUE

allowed the ga to run on multiple cores thus ensuring a faster runtime. The monitor value being set to TRUE was used to print the mean and maximum fitness function values at each iteration of the ga.

Both the multi-objective fitness function and the ga were put into another function which using the built-in function mapply and a sequence from 0.2 to 1 in increments of 0.05 allowed for the running of the genetic algorithm 17 times with different risk/return ratios. The value 0.2 means that the genetic algorithm would try to find a solution which was favouring the risk value and the higher it goes it begins to favour the return value more and the risk value less. The last run of the genetic algorithm was with a weight 1 which meant that only the return value is taken into consideration when running. After running the algorithm 17 times the function mapply would return a list with the 17 best solutions from each run in the form of a list of vectors. Using the split function in RStudio those were combined into a matrix. This matrix was then used to calculate the risk and return for each solution and plot them against each other, thus creating a Paretto plot. More details about the results are described in the results section of this report.

Results on the evolved solution

This section would include details of the results obtained from the genetic algorithms. After obtaining the list of vector solutions from the numerous runs of the genetic algorithm with different risk/return ratios that list was converted to a matrix. However those solutions are not scaled down to 1 like they are supposed to be so using a built-in function called lapply and giving it the matrix of solutions and the function for scaling the solutions as arguments it provided another matrix of scaled down solutions. Then using the built-in function sapply twice and giving it the scaled matrix of solutions as well as the functions for calculating the return and the risk it returned two vectors containing the return and the risk values for those solutions. After that those vectors were plotted against each other to provide a Paretto plot of the solutions. The achieved plot which can be seen in figure 3 below, was used to compare the different solutions, thus determining the best one according to the acceptable risk that an investor would be willing to take to maximise their return.

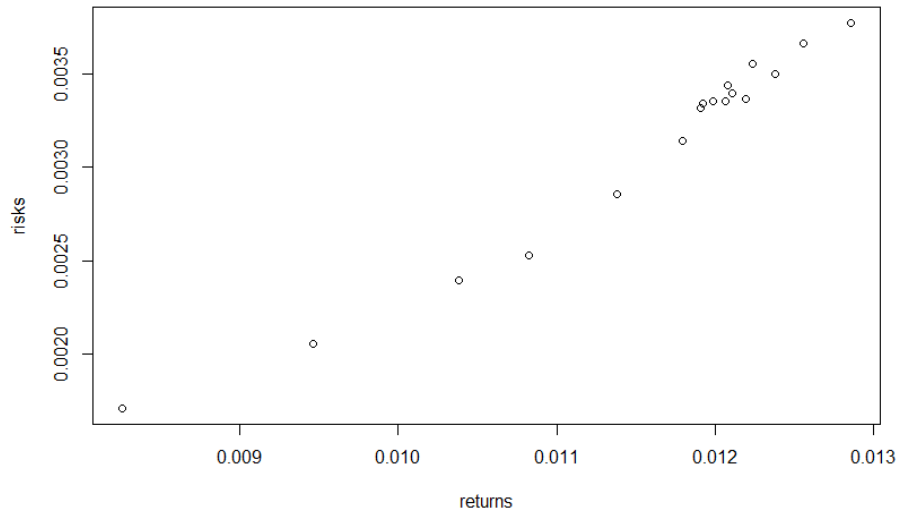

Figure 3: Sample Risk/Return Paretto Plot from a run of the Genetic Algorithm

As can be seen from the figure above when the return increases so does the risk. From this plot an investor could determine what risk they are willing to take for what return and figure out which solution would be the best. As can be seen the first 4 runs which prioritize minimal risk rather than high return have quite a steady return growth with not so high return growth. After that when the risk and return ratio is almost equal the risk starts growing at about the same rate as the return

and once the return gets a higher priority it reaches a point where it can no longer find a solution with a lot better return but no higher risk so the remaining solutions are very close to each other. During previous sample runs this genetic algorithm was able to reach a slightly higher return for a slightly lower risk thus proving that the algorithm does not provide the absolute best solutions every time and would require an increase of the number of iterations and the size of the population but that would greatly increase the run time.

A second sample run of the genetic algorithm for this portfolio was performed, which as can be seen from figure 4 below, provided similar values to the previous run but with a slightly higher return and a slightly more even increase of the risk values for each solution.

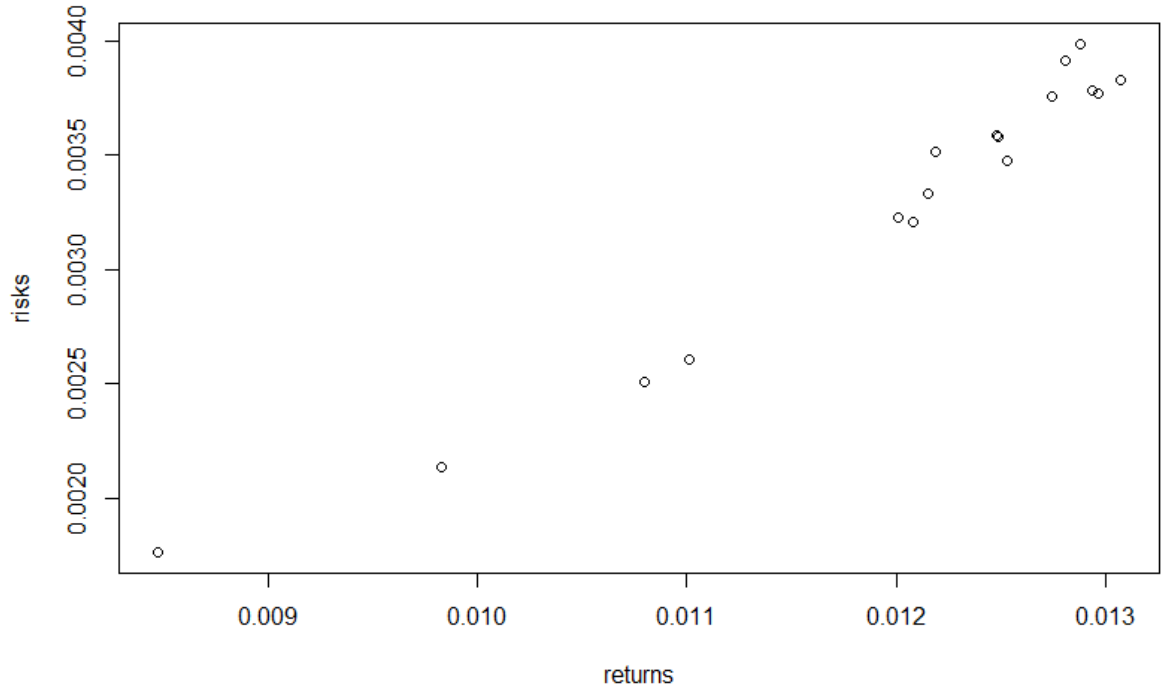

Figure 4 Sample Risk/Return Paretto Plot from a second run of the Genetic Algorithm

In my opinion this portfolio of assets would provide a nice steady growth of the investment without a very high risk. This portfolio was chosen on the base of having no negative returns for the data it was trained on so in order to provide further insight on the quality of this portfolio a test run of the solutions with new data containing the weekly returns for 1 month was performed. More details can be seen in the future performance section.

Future performance of portfolio

In order to test the 2 sample runs solutions against "future" unseen data, after the solutions were obtained a new set of data was acquired for the same portfolio. The test data contains the weekly returns for 1 month from the 1st of January 2019 to the 1st of February 2019. After doing the mean return and the covariance matrix for this test data, the same 17 solutions from the 2 runs of the genetic algorithm were tried out on the test data to see how it would perform.

As can be seen from figure 5 below the solutions from the 1st run of the genetic algorithm provided higher return values for lower risk values on the test set rather than the training set.

However this does not provide the same “slope” for the different risk/return ratios and the results are scattered all over the plot.

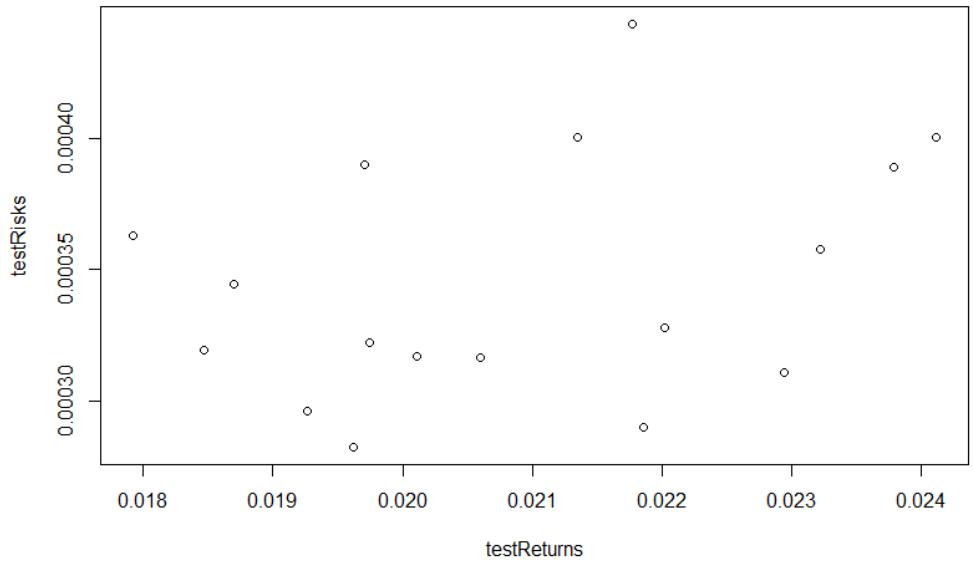

Figure 5: Risk/Return Plot on the test set from the first run of the Genetic Algorithm.

Running the solutions from the second run of the genetic algorithm against the test set showed a similar plot to the first one. However, in this plot it can be seen that there are 2 solutions with very high return values and very low risk values. The 2 runs of the GA on the test data confirmed my opinion for the portfolio. They both point to the portfolio being very good, providing high returns for low risk. Also those runs point that the genetic algorithm for portfolio optimization does not provide the best solution in all the cases and would require an increase in the number of iterations and the size of the population.

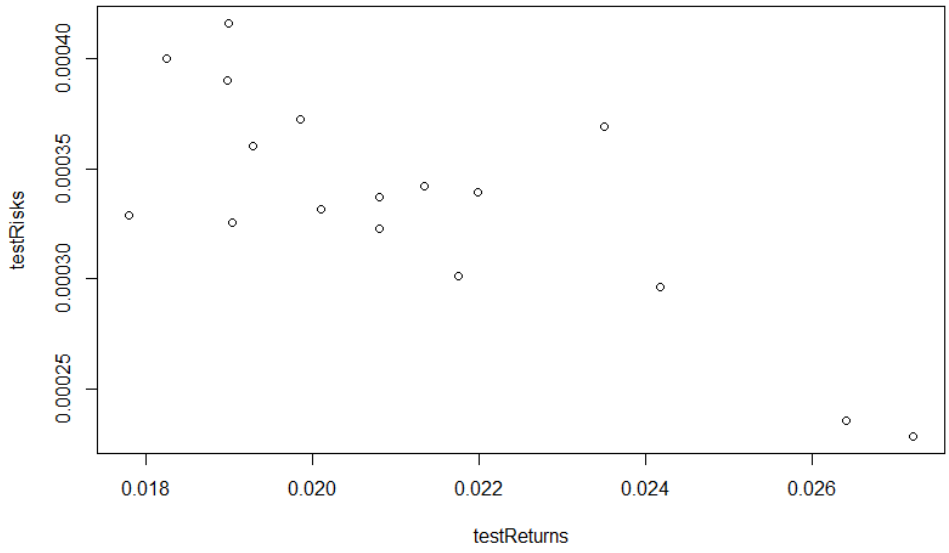

Figure 6: Risk/Return Plot on the test set from the second run of the Genetic Algorithm.

Analysis of the evolved portfolio(s)

In order to compare the Genetic Algorithm implemented against other algorithm a random search algorithm was created. The RSA involved creating a list with 10000 possible solutions and plotting their risk and return values. The plot for the RSA can be seen in figure 7 below. As can be seen from comparing the plot of the RSA with the plots of the genetic algorithms, the GA has provided better solutions with higher return and lower risk value.

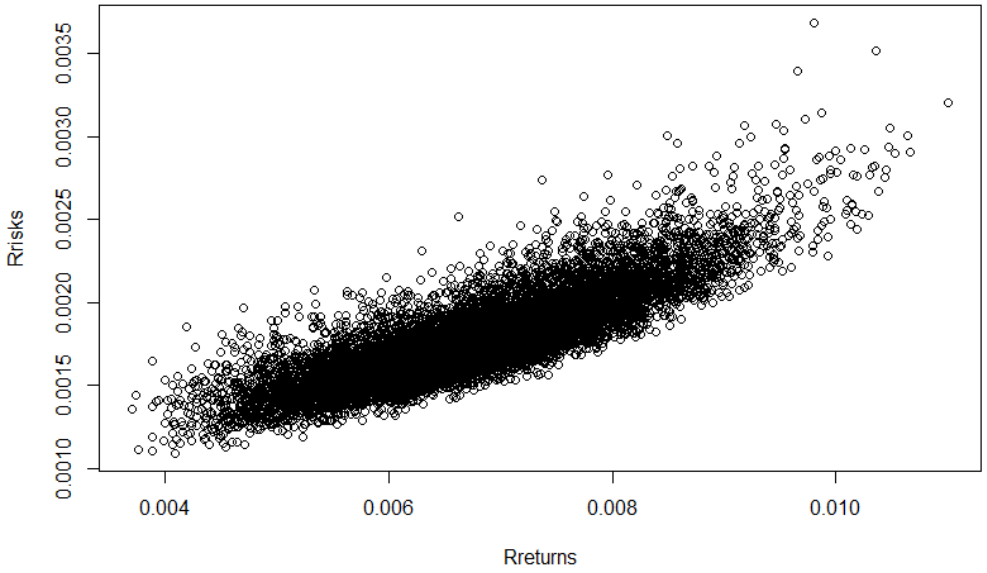

Figure 7: Random Search Algorithm plot of Risk/Return values.

In fact, the random search algorithm has provided very few solutions with return values as high as the lowest from the genetic algorithm but their risk values were a lot higher. From the RSA solutions, the one with the highest return value is:

0.057154399 0.264198040 0.055640646 0.015028301 0.040269812 0.104825074 0.008420580 0.024715679 0.230971662 0.003091302

This solution gave a return value of 0.011 and a risk value of 0.0032 compared to the solution with the highest return value of 0.013 and a risk value of 0.0038. This proves that the Genetic algorithm performs better in a portfolio optimization problem than the random search algorithm.