Liam Devlin

Portfolio Optimisation Report

Portfolio Optimisation describes the problem of optimising a set of assets (financial investments) such that the maximum return is achieved through the lowest amount of risk, making this a multi-objective problem. As such, no one fittest solution can be deduced as increases in return inevitably will lead to higher risk and vice versa. The individual we wish to generate takes the form of a vector of weightings, equal in length to the investors number of assets, which represents the financial investment that should be made to each company.

Chapter 1 - Overview of the problem

The task for this assignment was to find an optimised set of portfolio investment weightings which would minimise risk and simultaneously maximise the total returns from the portfolio. The data was obtained from yahoo finance and took the form of a vector of weekly returns for each asset in the portfolio, over the period of 4 years for training and 1 year for testing.

Figure 1.1 Return Formula

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

As can be seen from Figure 1.1, the total return is simply the sum of the weekly returns from each asset in the portfolio, multiplied by its weight. This is the metric we wish to maximise in the GA, however as previously stated, the risk is equally important to minimise.

Figure 1.2 Risk Formula

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij} w_i w_j$$

From Figure 1.2, we can see that the overall risk of a portfolio can be expressed as the sum of the covariance matrix, where *i* is the row of the matrix and *j* is the column. To obtain the covariance matrix, we examine the covariance between all assets in the portfolio(noting that cov(A,B) does not necessarily equate to cov(B,A)) meaning we must calculate the covariance for each "cell" of the matrix. The rows and column of a given covariance matrix are identical representing an asset in the portfolio. For example the value at exampleCovMat[1,2] would be the covariance between Asset 1 and Asset 2. After we have obtained the completed covariance matrix, we sum it to obtain the final "risk" value.

The final risk value represents a certain 'stability' of a portfolio, with lower risk values meaning your portfolio is diverse and each asset tends to make money at different times of the year. As is the case for most businesses, the income stream is not necessarily consistent the entire year, covariance also takes into consideration the overall "trend" of two companies returns. Take for example a swimming pool reseller and a coal trader, the likelihood of a low covariance value is high, due to the fact that swimming pools are likely to be bought more often at the beginning of a year to prepare for summer, while the opposite is true for coal, often being bought in bulk near winter to prepare for the cold.

Having prepared our performance metrics, collected data and built a model, a comparison was to be made against "future" data to evaluate the accuracy and feasibility of a given evolved portfolio. Furthermore, a comparison against random solutions will be performed as well as an "even-split" portfolio, in which each asset is assigned the same weight.

Chapter 2: Implementation

The following packages were used during the development of the final R script.

- **quantmod :** Used to retrieve the financial data from Yahoo.
- **GA :** The main Genetic Algorithms package that provides all required functionality (Mutation, Selection, Crossover etc.)
- **SpatialEpi :** Used for normalizing weight vectors.

Choice of Assets

Careful asset selection was imperative to the outcome of this system as poor assets will give poor results no matter the complexity of the system. A good principle to adhere to for selecting assets is to make sure the portfolio (of assets) is "wide" or covers a range of different business areas. The chosen asset selection (15 assets) focussed primarily on tech(Nvidia, Intel, Qualcomm and Google) , however assets were also selected from fashion (Crocs, Urban Outfitters, KBS) , health (Magellan, Vertex, AKER), food retailers (Costco, Starbucks and Chipotle) and entertainment sectors (Take Two and Johnson Outdoors). Within each sector a conscious decision was made to vary the type of business such that they were not competing (e.g. Costco does not compete with starbucks which does not compete with chipotle.)

Development:

The first task for this program was to collect the weekly returns of each asset over the period of 4 years (1-1-2014 : 1-1-2018), with the hope that a larger amount of historical data will improve the model's ability to predict more accurately in the future. This data was collected through use of the quantmod package. We first define the assets (portfolio) we wish to use as a vector of their trading symbols (e.g. Google = "GOOG"). Next we use the function getSymbols to load these assets into the quantmod package, providing the source we wish to collect the data from, as well as a period of time. Next we collect the weekly returns of our assets using the quantmod function weeklyReturn, which is then combined into a Dataframe object "myRetData".

```
# Load data
myStocks<-c("QCOM", "INTC", "NVDA", "GOOG",
            "TTWO", "AKER", "JOUT", "CMG",
            "KBSF", "COST", "SBUX", "CROX",
            "URBN", "MGLN", "VRTX")

getSymbols(myStocks, src="yahoo", from="2018-01-01", to="2019-01-01")

# Load stocks from yahoo into dataframe
myRetData <- data.frame(as.xts(merge(weeklyReturn(QCOM), weeklyReturn(INTC), weeklyReturn(NVDA), weeklyReturn(GOOG),
                                     weeklyReturn(TTWO), weeklyReturn(AKER), weeklyReturn(JOUT), weeklyReturn(CMG),
                                     weeklyReturn(KBSF), weeklyReturn(COST), weeklyReturn(SBUX), weeklyReturn(CROX),
                                     weeklyReturn(URBN), weeklyReturn(MGLN), weeklyReturn(VRTX))))
```

Figure 2.1 Getting weekly returns in R with quantmod

We next define an array of weighting for the GA. This is not to be confused with the weightings the GA will generate, but rather a weighting for the fitness function which determines how “dominant” one objective is over the other, for example a weight of 0.3 means that maximising returns would be weighted 30% to minimising risk which would be 70%.

Figure 2.2 Single Objective Formula

$$F(x) = w \cdot f_1(x) + (1 - w) \cdot f_2(x)$$

Risk function

Next we define a function that will calculate the risk value of a given portfolio. Firstly we create an empty matrix which has the same number of rows and column as there are assets in the portfolio (e.g. 3 assets = 3x3 matrix). Next we start a nested for loop to iterate through each element of the matrix, allowing us granular control and providing two indices which makes it extremely simple to calculate the covariance for each element of the matrix.

```
getRisk<-function(x){

  COV_MATRIX<-matrix(data=NA, nrow=dimensions, ncol=dimensions)

  # Create covariance matrix
  # i = row
  for(i in 1:dimensions){
    # j = column
    for(j in 1:dimensions){
      # take covariance of two assets, multiply this value by asset1's weight and asset2's weight
      COV_MATRIX[j,i]=cov(myRetData[myStocks[j]], myRetData[myStocks[i]]) * x[j] * x[i]
    }
  }

  return(sum(COV_MATRIX))
```

Figure 2.3 Getting weekly returns in R with quantmod

As can be seen from figure x.x, each element of the matrix is the covariance of the weekly returns between stock *j* with stock *i* multiplied by the weighting of stock *j* and stock *i*. Once the loop has completed and we have the full covariance matrix, we sum this to give the final risk value of a portfolio.

Returns Function

We must also define a function to calculate the average returns of a given portfolio. Firstly we create a vector of equal length to the number of assets in the portfolio, and populate this vector with the mean weekly return for each asset, multiplied by the given asset's weight in the portfolio. Finally, we sum this vector to get the mean weekly returns of our portfolio based on the current weighting.

```
getReturns<-function(x){

  # Find total return
  returnVec<-rep(0, dimensions)
  for(i in 1:dimensions){
    returnVec[i] <- mean(myRetData[myStocks[i]][[1]]) * x[i]
  }
  return(sum(returnVec))

}
```

Figure 2.4 Function to collect the total weekly returns from portfolio

Fitness Function

The fitness function is the backbone of our GA solution as it determines how individuals (asset weights) should evolve over time. Given that we already know that our fitness function must maximise returns whilst minimising risk, and considering that we have previously defined a function to calculate both risk and returns, the fitness function is relatively simple.

```
fitness<-function(x){

  # normalize vector such that is sums to 1.
  weights<-normalize(x)

  # risk = sum(COV_MATRIX)
  risk = getRisk(weights)

  # returns = sum of all weekly returns
  returns<-getReturns(weights)

  # Single objective forumla
  # f(x) = w . f1(x) + (1 - w).f2(x)
  final<-objectiveWeight %*% returns + (1 - objectiveWeight) %*% (1-risk)
  return(final)

}
```

Figure 2.5 Getting final weighted fitness value

First, it is important to note that each weighting vector generated by the GA will not be normalized as the method of generating each weight is simply to assign each element a random number between 0 and 1. As such the first step is to normalize the incoming weighting vector to sum to 1.

After we have retrieved the risk and return, we must use a formula to apply an objective weighting between risk and return as previously described. We first take the dot product of the weight and the first objective and the dot product of one minus the weight and one minus the risk. It is crucial that the the second objective of the formula is (1 - Risk) as the GA package tries to maximise its overall goal, and thus the closer the second value is to 1, the less overall risk we have.
We finally return the value produced from this formula as our final fitness score. To prepare the future data, we repeat the above steps again without running the GA, which allows us to compare our evolved solutions on unseen data, with a comparison against the other provided methods.

Chapter 3: Results

To test the results of the created system a comparison was made against a portfolio of equal investment weightings , as well as a random comparison. Furthermore, the system was tested against future data (2018-2019) to evaluate not only the evolved model's accuracy, but its feasibility as a real portfolio of investments.

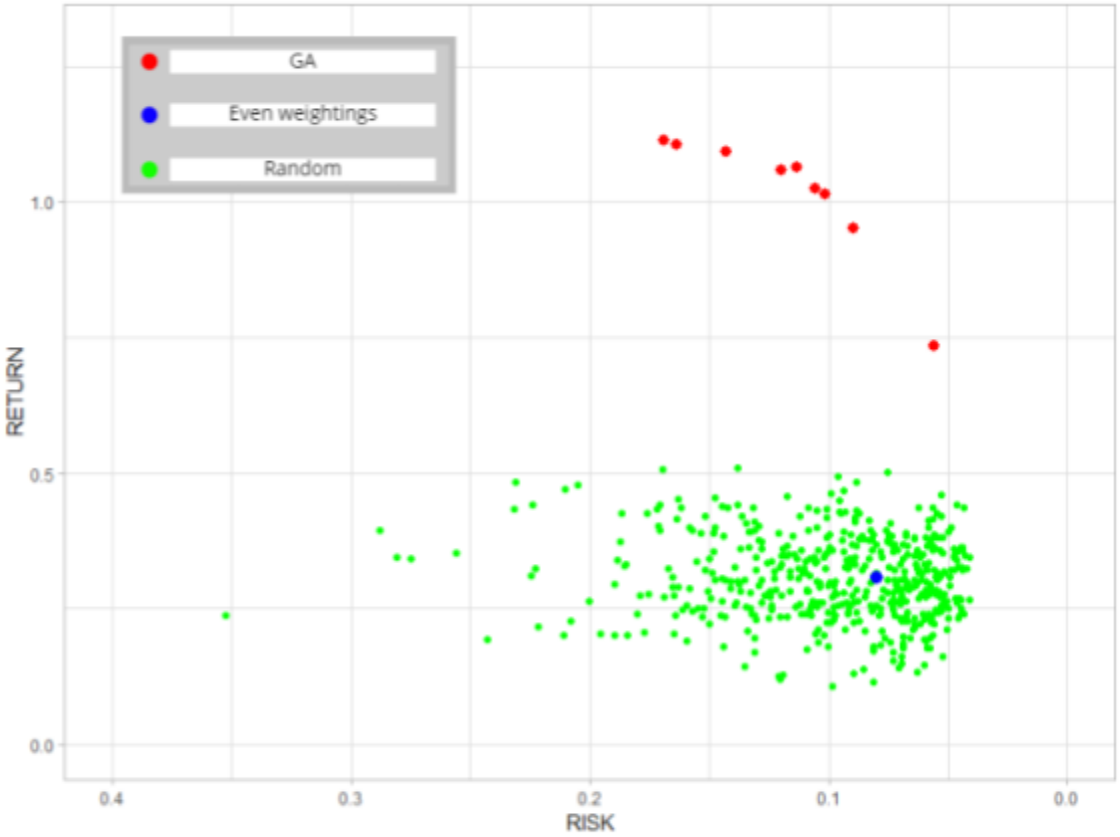

Figure 3.1 Results from training set

As can be seen from figure x.x, the GA out-performed random and even-weight splitting by a comfortable margin on the test set. The evolved solutions exhibit pareto optimality-like behaviour (with one outlier), in which no one solution is dominated by any other, which is the result we expect to see when tackling a multi-objective problem.

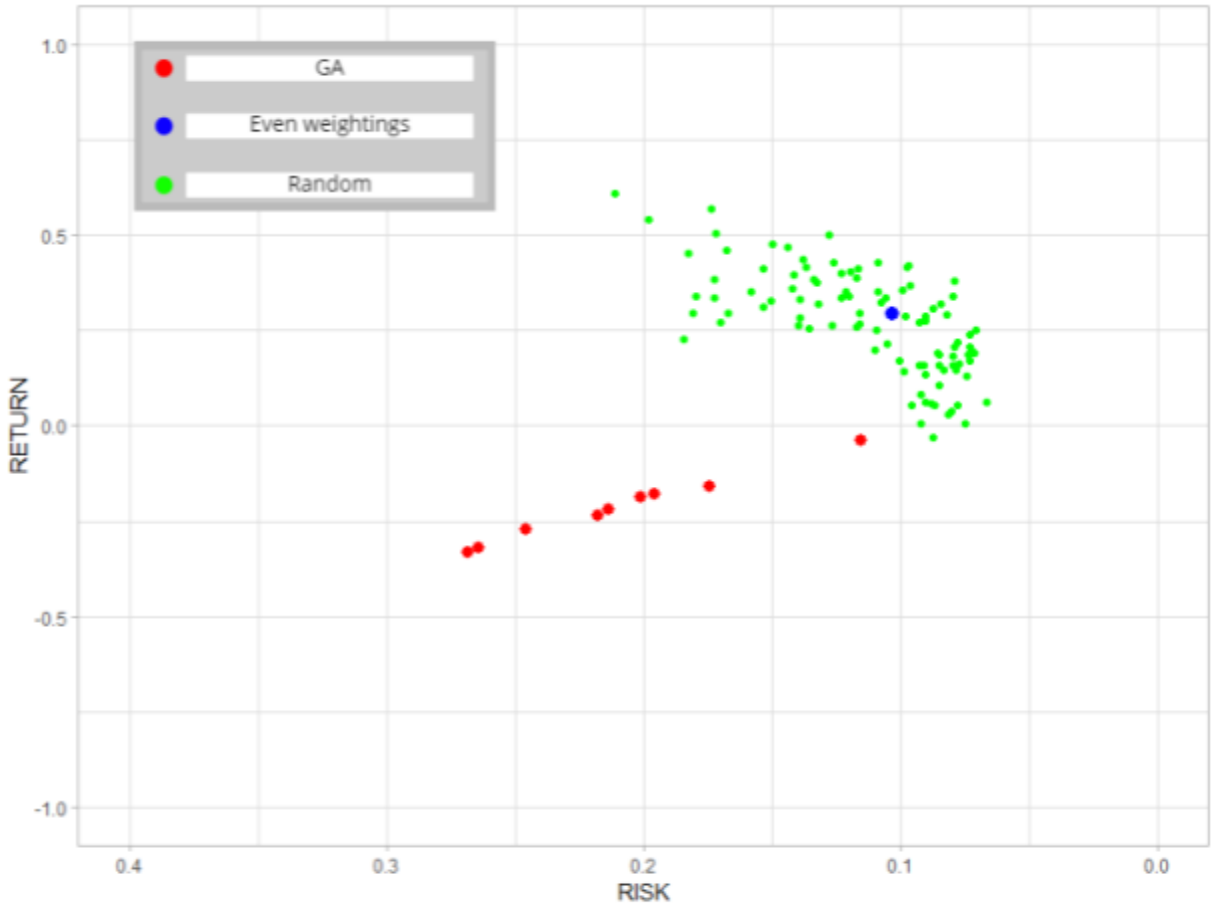

Figure 3.2 Results from test set

The evolved solutions were relatively high in risk compared to those offered by the random solution, however in turn the returns of each portfolio was significantly higher than any random or evenly weighted portfolio. From these results, it would be most easy to recommend a portfolio with a roughly even splitting between risk and return importance. The graph shows that for solutions in the region of 0.1 risk score have a similar return to those at 0.2 risk. Equal weighting of portfolio assets provided similar results to the best random solutions, but again failed to match the returns possible for low risk values that the GA was able to provide.

With this being said, it is also important to compare the results of the proposed portfolio against future data, to access the accuracy of the generated model. Unfortunately, the model fails miserably with future data, being vastly outperformed by Random and Even split weightings. The likely reason the GA produced such underwhelming scores is due to poor asset selection and a lack of historical data. Taking longer to assess each area of the market and which companies are likely to succeed may help to increase the feasibility of the generated GA solutions.

Another potential issue is that the data was overfitted to the period that was sampled for training, which could be alleviated by investing in more established companies with longer financial records to draw from.

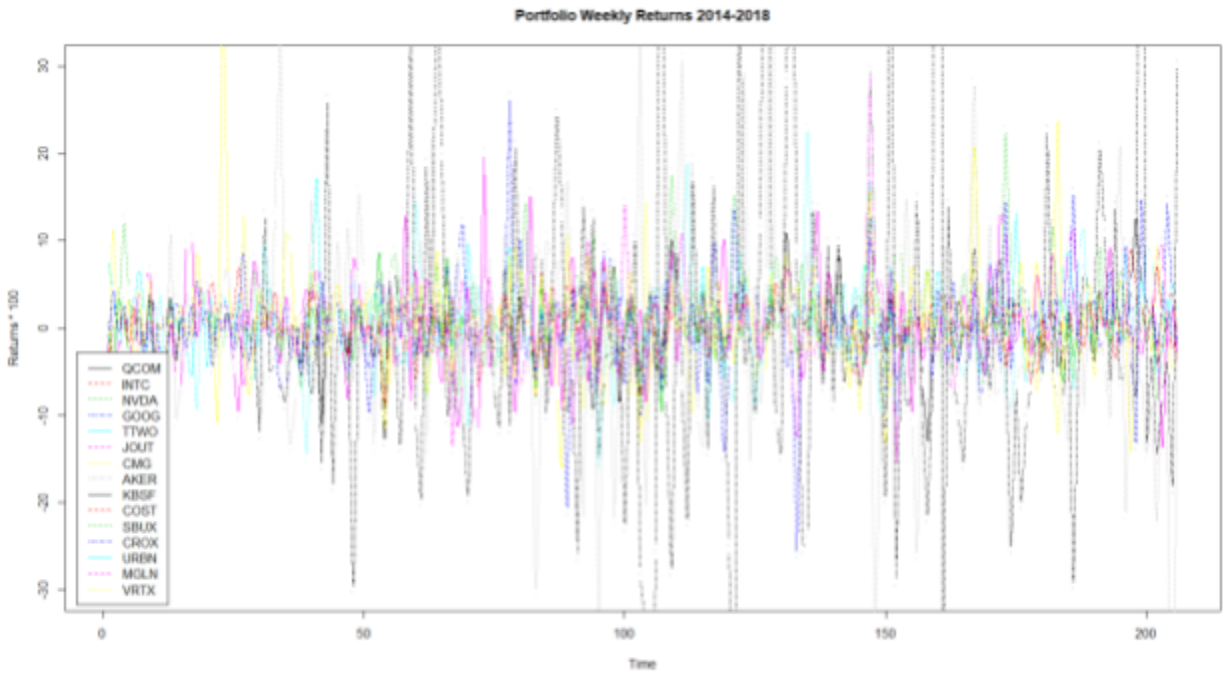

Figure 3.3 Weekly returns of assets from 01-01-2014 to 01-01-2018

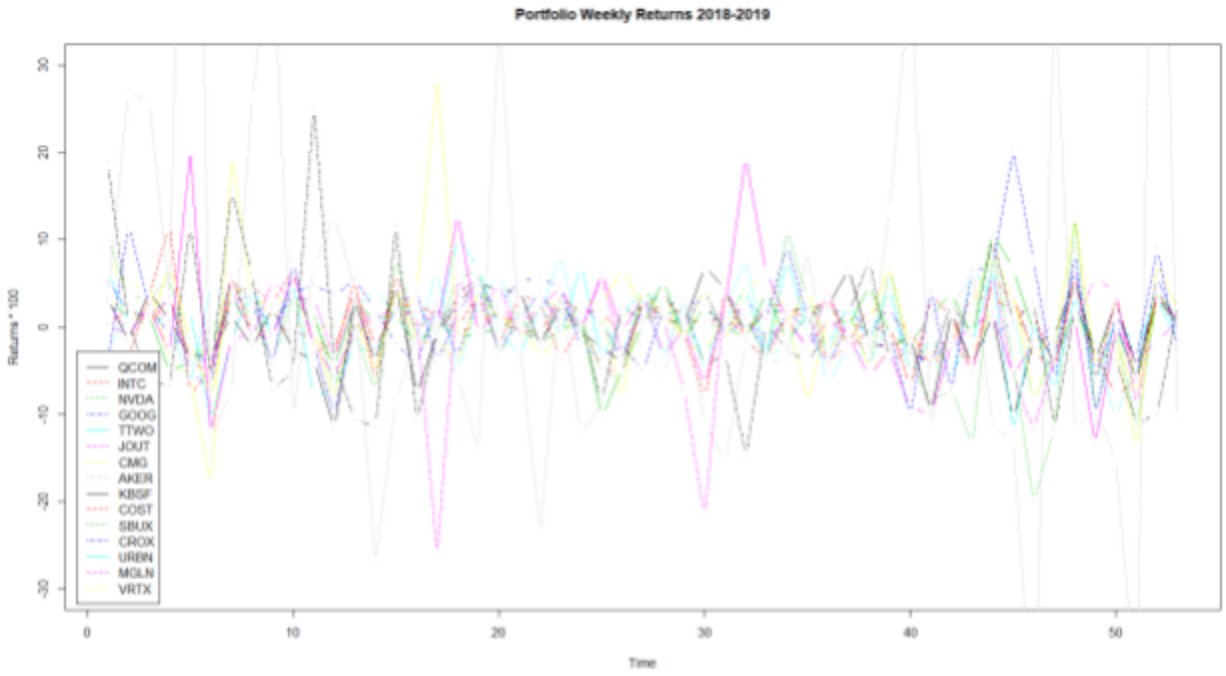

Figure 3.4 Weekly returns of assets from 01-01-2018 to 01-01-2019

Comparing the weekly returns between the training period (2014 - 2018) and the "future" period (2018 - 2019) we can see a large variation in overall returns between the training period and the test period. This could also be a major contributing factor as to why the predictions on the test data were so inaccurate, as can be seen from both plots the data is extremely erratic and does not move in an easily definable way. Again, a larger historical data-set would likely improve the results of the GA significantly.

Overall the generated GA solutions would not make suitable portfolios due to their terrible performance in the future, even when compared with random and equal - weighted investment.