Portfolio Optimisation

Construction of a portfolio using the GA package

Loading Assets:

The assets selected for the initial portfolio are from a diverse range of industries during the financial trading period of *2021-2022*. Though markets may still be adjusting from the aftermath of COVID-19, using time periods **before** the pandemic may not give an accurate representation of current/future performance.

```
assets <- c("SMPL", "WING", "NFLX", "LLY", "ETSY",
            "CAT", "HAS", "SONY", "IBM", "LEVI")
```

Fitness Function

The Sharpe Ratio was the objective function selected as the GA's fitness function. The Sharpe Ratio is used as a method for measuring risk adjusted returns, meaning optimising a portfolio is achieved by simply calculating the ratio of returns over risk.

```
meanReturnsTrain <- apply(trainReturns,2,mean)
covMatTrain <- cov(trainReturns)
```

To extract returns and risk from the asset data, the mean and co-variance of the daily returns were used. Simultaneously maximising the returns whilst minimising the risk is classified as a multi-objective problem. However, due to limitations of the GA package, the alternative approach is to combine these into a single function.

```
getReturnsTrain <- function(x) {
  (sum(x * meanReturnsTrain)*252) # multiply by 252 to annualise returns
}

getRiskTrain <- function(x) {
  (sqrt(t(x) %*%(covMatTrain*252) %*% x)) # multiply by 252 to annualise risk
}

sharpe <- function(x) {
  return(getReturnsTrain(x)/getRiskTrain(x))
}
```

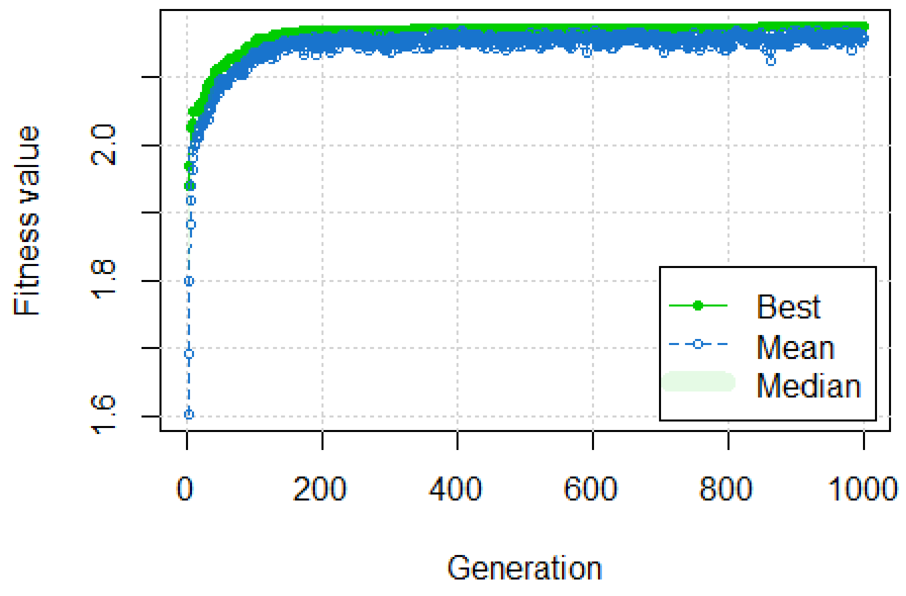
Genetic Algorithm Parameters

The parameters selected for the GA and their justifications are as follows:

```
lower <- c(rep(0,nStocks))
upper <- c(rep(1,nStocks))

GA <- ga(type="real-valued", fitness=sharpe, lower=lower,
         upper=upper, maxiter=1000, popSize=100, seed=69)
```

Param	Value	Reasoning
type	real-valued	continuous optimisation for finding weights
fitness	sharpe	combines the risk and returns
lower	>= 0	weights should be a postive number
upper	<= 1	individual weights cannot be greater than 1
maxiter	1000	based on a similar study by Chang et. al. (2009)
popSize	100	based on a similar study by Chang et. al. (2009)



Plot showing the Maximisation of the fitness function suggests that 1000 generations might be overkill as it starts to stablise around 400 iterations.

Analysis of Evolved Weights

The weights are required to be normalised so that they do not sum to > 1. The ratio of each weight relative will remain the same so results should not be affected.

```
##        SMPL   WING   NFLX    LLY   ETSY    CAT    HAS   SONY    IBM   LEVI
## [1,] 0.1117 0.1146 0.0194 0.3959 0.0054 0.0175 0.0432 0.1424 0.0855 0.0644
```

It is clear that the LLY stock outperforms the majority of the other stocks in the training data. The projected returns and risk from applying the GA's derived weights can also be calculated. Though these values may not be useful on their own, maximising the Sharpe ratio leads to a more optimal investment, hence a higher return and a lower risk value is desirable.

```
## Expected Returns from evolved GA weights: 0.3674

## Expected Risk from evolved GA weights: 0.1689

## Sharpe Ratio from evolved GA weights: 2.1753
```

Thus the Sharpe ratio can be simply calculated by dividing the weighted return by the weighted risk to determine a baseline to which other portfolios can be compared to. Typically a Sharpe Ratio value greater than 1 is considered ideal so 2.1753 is considered very good in this case.

Evaluation of the portfolio on "future" data

Using the optimal weights evolved from running the GA, the expected future return and risk can be calculated by defining a similar function to what was done with train data.

```
testReturns = lapply(assets, function(sym) {
  dailyReturn(na.omit(getSymbols(sym, from="2022-01-01", to="2023-01-01",
  auto.assign=FALSE)))
})
```

This time, the daily returns from the following year *(2022-2023)* were used as test data to get the corresponding mean daily returns and covariance.

```
meanReturnsTest <- apply(testReturns,2,mean)
covMatTest <- cov(testReturns)

getReturnsTest <- function(x) {(sum(x * meanReturnsTest)*252)}
getRiskTest <- function(x) {(sqrt(t(x) %*%(covMatTest*252) %*% x))}
```

After calculating the weighted returns and risk, it is immediately noticeable that performance for future data is significantly worse.

Applying the Sharpe Ratio, the value is 0.0568 for the test data compared to 2.1753 for the train data. The initial concern of volatile markets after the effect of COVID-19 is a possible explanation behind such a drastic drop. Perhaps utilising training data over the course of 1 year was too short sighted - so expanding the period of the training data over multiple years i.e. 2018-2022 could improve predictions on future data.

```
## Expected Returns using Test Data: 0.0136

## Expected Risk using Test Data: 0.2393

## Sharpe Ratio using Test Data: 0.0568
```

That being said, this test data can still provide a very valuable benchmark when performance of the GA portfolio is compared with other portfolios such as one with equally balanced weights or weights generated randomly. These comparison will make it clearer whether the evolved weights were indeed suboptimal or whether the assets just had a poor performance during the year picked for the test data.

Evaluation of Performance across different Portfolios

A balanced portfolio is one consisting of equal weightings (0.1 in this case):

```r
balanced_weights <- rep(0.1, length(assets))
```

A Random portfolio can be achieved by Random Search Algorithm. The RSA approach involves running a number of searches with randomised weights and take the best results from the results. The "best" result in this case can be defined as either: weights which yield the minimum amount of *risk*, maximum amount of *returns* or highest *Sharpe Ratio*.

Comparison on Train Data

Train Set Performance	GA Evolved	RsA Maximise Return	RsA Minimise Risk	Balanced
Returns	0.3674	0.3524	0.2414	0.2665
Risk	0.1689	0.1857	0.1386	0.1596
Sharpe	2.1753	1.8977	1.7417	1.6698

Firstly, it is worth noting that the risk from the **balanced portfolio** is lower than that of the **GA** (0.1596 vs. 0.1689). Though initially surprising, it makes sense as the balanced portfolio can be considered as more "safe" by equally spreading the risk across all assets but it spreads itself too thin as evident by the worst performing Sharpe ratio when compared to the other portfolios.

The best performing **RSA** in *maximising returns* did indeed yield better returns than both **GA** and **balanced portfolios**. Similarly when considering the **RSA** with *minimal risk*, the expected risk is also better than both **GA** and **balanced portfolios**. Despite this, the GA evolved portfolio has the highest Sharpe ratio at 2.1753 and therefore can be considered the most optimal as the Sharpe Ratio considers both risk and return.

Comparison on Test Data

Test Set Performance	GA Evolved	RsA Maximise Return	RsA Minimise Risk	Balanced
Returns	0.0136	-0.0361	-0.0742	-0.1522
Risk	0.2393	0.3002	0.2697	0.3034
Sharpe	0.0568	-0.1203	-0.2751	-0.5016

For the test data, there are a number of key differences. Namely, the **GA** actually provided a solution that was actually most *risk-adverse* - even when compared to that of the **RSA**. More importantly, confirmation of the poor performance of the GA on test data was seen all across the other portfolios.

In fact, the only weights which managed to achieve a positive return was the **GA evolved portfolio**. The other portfolios all ended up with a loss - with the **balanced portfolio** performing the worst. This suggests evenly distributed weights in concept may be the safer approach but in practice, it could end up backfiring.

Evaluation of portfolios with differently balanced risk and return

The current GA is based off the assumption that the weightings of risk vs. return should be balanced. To explore the relationship between risk vs. reward, the fitness function can be modified to adjust the preference of either factors and emulate a multi-objective approach.

Weighted Fitness Function

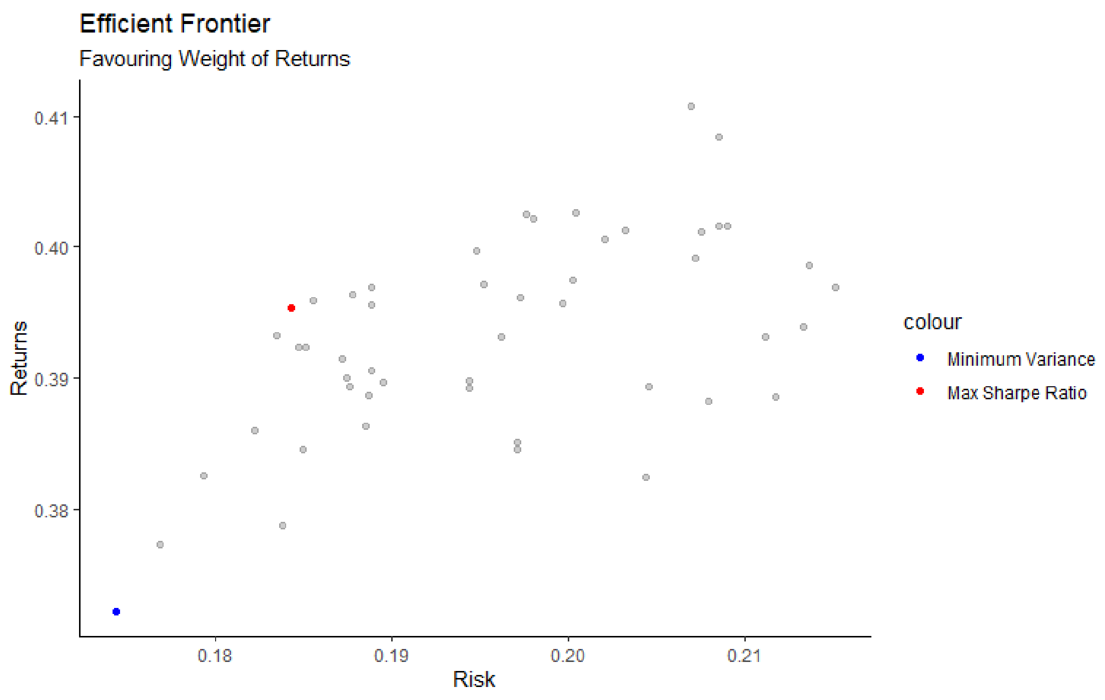
Instead of simply calculating the Sharpe ratio, the fitness function can take in a pair of parameters to apply a skew to the return or risk before calculating the sharpe ratio.

```
fitSkew <- function(x, s_ret, s_risk) {
  x <- x/sum(x)
  fit <-(s_ret+getReturnsTrain(x))/(s_risk+getRiskTrain(x))
  return(fit)
}
```

The GA was slightly altered from the initial implementation to account for the extra parameters needed to adjust the weights of returns and risk. It was wrapped by a function to run it 50 times, automatically incrementing the skew applied to model a more risk adverse priority or a return maximisation preference.

Prioritising Maximising Returns

The first case to analyse is adjusting the weight to favour maximising projected returns. The resultant weights from each run of the GA are utilised to calculate the returns/risk which are then used to plot an efficient frontier:



The max projected returns from this experiment: 0.4108 outperformed every other portfolio including the original GA portfolio which had a return of: 0.3674. However, the risk associated from this portfolio: 0.2069 is significantly higher than the risk of the original GA and even the RSA.

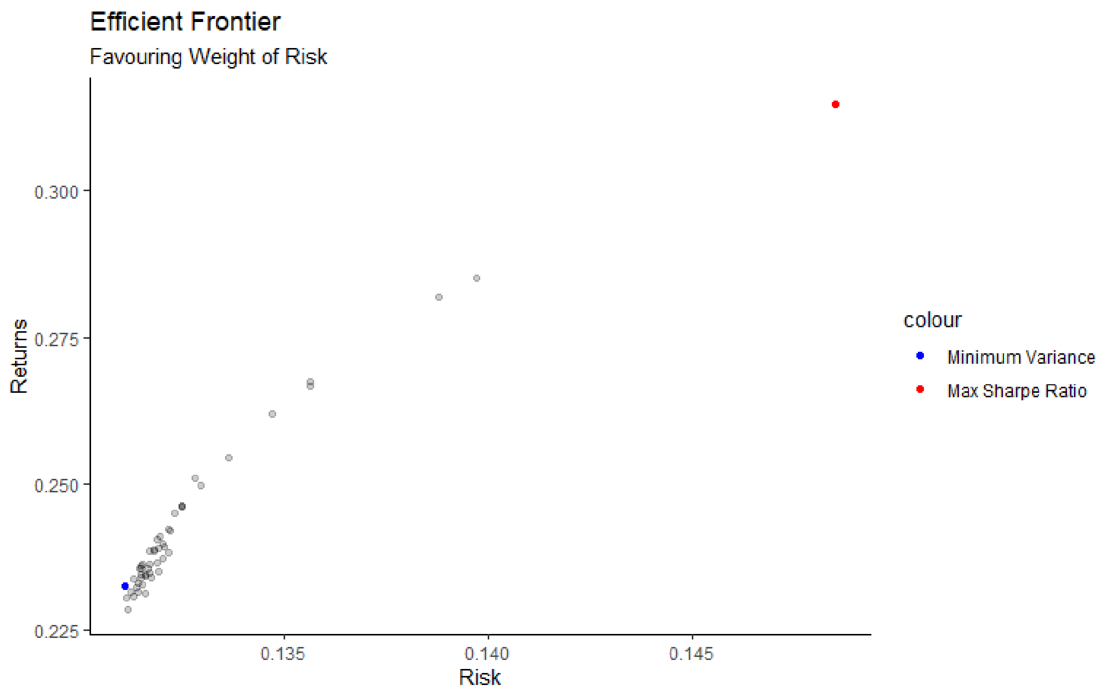
The Sharpe Ratio value lies on the Pareto front and was identified to be: 2.1445. When comparing to the other portfolios, the new GA still outperforms the RSA portfolios: [1.8977, 1.7417] and the balanced portfolio: 1.6698. The most optimal approach is still the original GA: 2.1753.

It can be concluded that the original balanced weighting of the risk:return split is more optimal, but if one prioritise returns over risk, then adjusting the weighting to a certain degree will still reap an optimal solution.

However, care needs to be taken to preserve an acceptable Sharpe Ratio - especially in the cases where increasing the weighting can lead to a negative Sharpe Ratio. In particular, the same experiment was ran across the test set and it was found to be more sensitive to adjustments to return. This is due to inflating the already high risk as a consequence of marginally improving returns.

Prioritising Minimising Risk

The effects of minimising risk can be explored in a similar approach to maximising returns. This time, a skew is added to the denominator of the fitness function to inflate the weightings assigned to the risk. As before the plot of all risk-return trade offs can be analysed:



Results of minimising risks exhibit a similar relationship to that of maximising returns - a positive correlation. Whereas the points in the first rebalance of risk:return weighting were more erratic and formed a front, the effects of decreasing risk seem to exhibit a more linear relationship between risk vs. returns.

The minimum Variance label on the plot shows the solution which gave the lowest risk: 0.131102. Although it is the solution which results in lowest risk out of all portfolios, it is also the worst performing in terms of returns: 0.2325188 compared to GA: 0.1689, RSA: [0.2414, 0.3524] and balanced: 0.2665.

Moreover, the most optimal Sharpe Ratio in this case was actually where the weightings were most balanced. This suggests that risk actually has less influence on the overall optimality of the solutions than the returns. This observation starts to make sense when considering that the balanced portfolio had the worst performance in terms of being optimal, despite having an extremely low risk.

Once again, the risk:return trade-off is an important consideration since theoretically lower risk usually suggests a positive effect. However, the results here argues that reducing risk will be worse off in terms of finding an optimal solution, much like the previous attempt of inflating return.

Extreme Cases of Maximising Return and Minimising Risk

Another avenue to evaluate involves exploring either extremes of maximising returns and minimising risks.

Full Maximisation of Return

The new fitness function will simply be a maximisation problem as follows:

```r
fitMax <- function(x) {
  x <- x/sum(x)
  return(getReturnsTrain(x))
}
```

Full Minimisation of Risk

Similarly, this fitness function will be a minimisation problem, hence the -ve return:

```r
fitMin <- function(x) {
  x <- x/sum(x)
  return(-getRiskTrain(x))
}
```

Results against original GA

Train Set Performance	GA Balanced	GA Max Return	GA Min Risk
Returns	0.3674	0.5124	0.218
Risk	0.1689	0.2867	0.1305
Sharpe	2.1753	1.787	1.6703

Though the relative return and risk values are considered better than the balanced GA, the Sharpe Ratio scores once again highlights the importance of considering the return/risk trade off. Moreover, performance of the new GAs has deteriorated to the point where it is more comparable to the balanced portfolio.

Asset Selection

Preprocessing Asset Pool

For selection of assets from a larger pool, using the SP500 stocks would be a suitable starting point as these typically are valued as high performing stocks. A sample of 100 from the SP500 assets were taken but this process can be easily upscaled to using the entire pool of 500 if a more comprehensive solution is required.

A combination of the tidyquant and yfR packages were used to obtain the daily returns of the sampled assets as quantmod struggles to handle errors from processing missing stocks.

```
sp500 <- yf_index_composition("SP500")

## ✓ Got SP500 composition with 503 rows

set.seed(1)
assets <- sample(sp500$ticker, 100)
```

Representation of problem

A different approach to portfolio optimisation is required for selecting assets as the nature of the problem is more suited to using a binary representation. In particular, a study by Vaezi et. al. (2019) attempts to apply the knapsack problem in a portfolio selection process and thus will be the motivation for this project's approach.

The knapsack problem is a useful way of thinking about portfolio selection as they are both essentially searching for the ideal set from a larger pool. However, the criteria in which the algorithm determines a good asset from a bad one will still be ultimately based on Sharpe's Ratio. As Portfolio Optimisatio is the ultimate goal, balancing risk with returns is still vital in this selection stage. Thus, similar to the original approach, the mean daily returns and covariance matrix will be required:

```
meanReturns <- apply(dailyReturnsXTS,2,mean)
covMat <- cov(dailyReturnsXTS)
```

It appears to be agreed upon that a Sharpe Ratio falling below 1 is sub-optimal suggesting that investment is only worthwhile when it has a score of at least 1. Therefore, the selection criterion will return a 0 for a stock with a suboptimal Sharpe Ratio and a 1 for stocks with an acceptable score. The modified fitness function now looks like this:

```
sharpe <- function(x){
  return(getReturns(x)/getRisk(x))
}
```

```r
selection = function(x) {
  if (sharpe(x) <1) return(0)
  else return (sharpe(x))
}
```

Updated GA Parameters

Firstly, a custom population and mutation functions are defined so that exactly 10 assets are picked in the final solution. This is to ensure that this function is consistent with the portfolio optimisation pipeline defined earlier:

```r
customPopulation <- function(k){
  function(GA){
    m <- matrix(0, ncol = GA@nBits, nrow = GA@popSize)
    for(i in seq_len(GA@popSize))
      m[i, sample(GA@nBits, k)] <- 1
    m
  }
}

customMutation <- function(GA, parent){
  index <- which(GA@population[parent,] == 1)
  change <- sample(3, 1)
  index[sample(length(index), change)] <- sample(setdiff(seq_len(GA@nBits),
index), change)
  parent <- integer(GA@nBits)
  parent[index] <- 1
  parent
}
```

Tournament selection is used in this instance as a study by Owais et. al. (2020) confirms that it was the optimal approach for solving the binary knapsack problem. The study also recommends to use one point crossover but it only explored one point vs two point. The results of using one-point crossover does not ensure exactly 10 assets are picked. As uniform crossover ensures this constraint is maintained, it will be the function adopted.

```r
gaControl("binary" = list(selection = "gabin_tourSelection",
                          crossover = "gabin_uCrossover"))
```

It is also important to define the GA as a binary problem as shown:

```r
selectionGA= ga(type='binary', fitness=function(x){selection(x)},
               nBits=ncol(dailyReturnsXTS), names=symbols, maxiter=500,
               mutation=customMutation, population=customPopulation(10),
               popSize=100, keepBest=TRUE, seed=1)
```

Selected Assets

It is interesting to note that none of these assets existed in the original set of assets that were manually picked.

```
##  [1] "MRO"  "PANW" "IRM"  "UNH"  "WST"  "ANET" "HSY"  "NOC"  "CPT"  "CHD"
```

Evolve Weights for Selected Portfolio

To evaluate the competency of this algorithm, portfolio optimisation of the top 10 selected assets will be conducted. Similar to the original GA, an output of weights will be assigned to each of the 10 assets. This portfolio will then be compared to the GA evolved portfolio in the first part.

The same GA used for the original Portfolio Optimisation is used. The only difference in the fitness function is utilising the mean returns and covariance of the new assets:

```
newGA <- ga(type="real-valued", fitness=sharpeNew, lower=lower,
            upper=upper, maxiter=1000, popSize=100, seed=69)
```

The evolved weights of the new assets are as follows:

```
##         MRO   PANW    IRM    UNH    WST  ANET    HSY    NOC   CPT    CHD
## [1,] 0.0959 0.0494 0.0752 0.0899 0.1878 0.058 0.0049 0.0695 0.305 0.0645
```

Evaluation of the Selected Portfolio

A comparison of the new portfolio can be compared to the original portfolio. Immediately, the performance of the new portfolio appears to provide an even more optimal solution than that of the first. As the original GA portfolio already outperformed that of all other portfolio, it is surprising to see this one provided even better results. However, it is important to note that these are results on the train set.

Train Set Performance	GA Original	GA New
Returns	0.3674	0.5607
Risk	0.1689	0.129
Sharpe	2.1753	4.3475

Looking at the test set, it is evident that even this portfolio struggles to perform on unseen data and indicates some level of overfitting (as it performed significantly better in the train set). This may indicate that the algorithm may not be a suitable predictor of future performance, especially in more turbulent financial conditions.

Test Set Performance	GA Original	GA New
Returns	0.0136	-0.1722
Risk	0.2393	0.2353
Sharpe	0.0568	-0.7322

For future work, increasing the size of the train set (i.e. range of 10 years+) may be a potential direction in improving future predictions and allow this algorithm to adapt to significant market changes such as the 2021-2023 period explored in this work.

References

Chang, T.J., Yang, S.C. and Chang, K.J., 2009. Portfolio optimization problems in different risk measures using genetic algorithm. Expert Systems with applications, 36(7), pp.10529-10537.

Owais, W.B., Alkhazendar, I.W. and Saleh, D.M., 2020. Evaluating the impact of different types of crossover and selection methods on the convergence of 0/1 Knapsack using Genetic Algorithm. arXiv preprint arXiv:2010.03483.

Vaezi, F., Sadjadi, S.J. and Makui, A., 2019. A portfolio selection model based on the knapsack problem under uncertainty. PloS one, 14(5), p.e0213652.