Investigating portfolio optimisation using GA

1.Introduction

In this report, I will use GA(genetic algorithm) package and quantmod package to conduct analysis about portfolio optimisation.More specifically, Value-at-Risk. First, necessary libraries are imported.

```
library(quantmod)
library(GA)
```

2.Construction of a portfolio using GA package

2.1 Getting the stocks prices from 2019-01-01 to 2021-01-01.

I have chosen 8 stocks which either me or my acquaintance has ever invested in and they come from several different sectors. By selecting portfolio in this way, the report would fit purposes for this module as well as for shedding some light on real-world investment. I have chosen the time period from 2019 to 2021 so that I have a 2 years time span's data to feed into GA and then there is one year from 2021 till now to evaluate the portfolio on.

2.2 Forming the portfolio data fram and covariance matrix from which I calculate the risk and renaming the matrix with the abbreviations of the stocks.

```
myReData <- data.frame(as.xts(merge(dailyReturn(`600378.SS`),
                                    dailyReturn(`002405.SZ`),
                                    dailyReturn(`002230.SZ`),
                                    dailyReturn(`300081.SZ`),
                                    dailyReturn(`000795.SZ`),
                                    dailyReturn(`300459.SZ`),
                                    dailyReturn(`600660.SS`),
                                    dailyReturn(`002390.SZ`))))
colnames(myReData) <- c('hh','swtx','kdxf','hxdf','ylh','tmm','fybl','xbzy')
CovMatrix <-cov(myReData)
my_mean <- apply(myReData,2,mean)
```

2.3 Constrcution of evaluation function and setting parameters for GA.

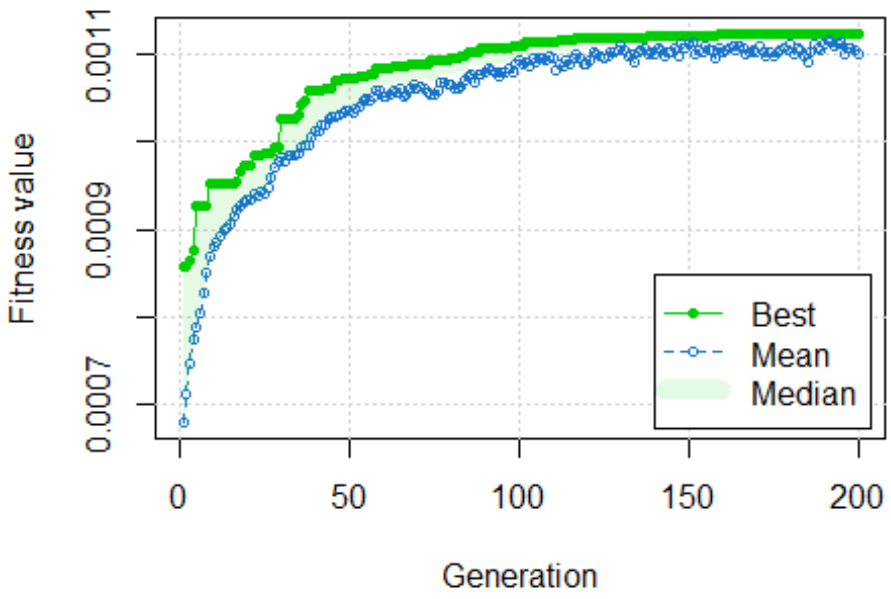
To construct the evaluation function, we need to take into consideration the return and risk at the same.

First we map the weights into the interval of [0,1] by dividing each weight with the total sum of weights and thus they represent the proportion of investment in each stock. Average daily return is easily required by calculating inner vector product between weights vector and return vector.

Risk value is obtained by conducting element-wise multiplication on covariance matrix of the stocks and the weight matrix of the stocks, which is built by using outer( ) function to perform exterior product on the pairs of weights, and summing all terms up.

Having done so, we are able to build the evaluation function from what we obtained earlier.Since we want to maximise the return while minimise the risk, obviously this fitness value has a positive relationship with return and negative one with risk. So we put the expression of return over that of risk. The original risk value could be negative or positive. Simply placing the risk value at the denominator would result in a non-monotone relationship as the fitness value goes to infinity when risk approaches 0 from positive axis and goes to minus infinity when approaching 0 from negative axis. In order to fix the this problem, we deliberately construct which map the risk value onto the interval of [0,1]. Sigmoid function does this job perfectly.

I use lsSelection to select parent after comparing the performance among the candidates. BlxCrossover and nrsMutation are chosen for performance reasons after some comparisons. The crossover rate is set to 0.6 due to what I have read in other papers about GA saying that a high crossover rate would result in sub-optimal solutions. Mutation rate is set to 0.05 as it is commonly thought to perform well, not too high so that the population change irregularly and slow down convergence speed nor too low that it could not escape local optima. Population size is set to 100 which I consider a moderate amount and max iteration is set to 200 as I found sometimes the best fitness could still improve after 150 iterations but seldom does so after 200 iterations.



```
## [1] 96.68359615  0.14481040  0.02684495  0.03764882  0.18759119  0.0
0977737
## [7]  0.17071530  0.04943923

## [1] 0.9935584777 0.0014881284 0.0002758692 0.0003868940 0.0019277605
## [6] 0.0001004761 0.0017543373 0.0005080569
```

To make the examination easier, I scale the return, risk and fitness values in the following section by 10^4 instead of scaling them in the building of evaluation function just to avoid expressing the preference of return or risk in the first place.

The solution's return and risk values are 22.4820544 and 9.9069063 respectively.

3. Evaluation on future data

3.1 Generating other possible portfolios and get data ready for exploration.

The data used for evaluation are from 2021-01-02 to 2022-01-01. We deliberately generate a balanced portfolio, whose weights are all equal for each stock, and a random portfolio, whose weights are randomly produced.

```
## [1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125

## [1] 0.1161972 0.1003521 0.1390845 0.1320423 0.0721831 0.1496479 0.16
54930
## [8] 0.1250000

## [1] 0.9935584777 0.0014881284 0.0002758692 0.0003868940 0.0019277605
## [6] 0.0001004761 0.0017543373 0.0005080569
```

The weights are generated in the order of balanced method, randomly chosen method and evolution method.

3.2 Performance comparison between the 3 portfolios on future data.

3.2.1 On future data

The GA generated portfolio's return and risk are 39.4665219 and 11.3649762 respectively.

The balanced portfolio's return and risk are 19.3231923 and 2.4442566 respectively.

The random portfolio's return and risk are 18.8118616 and 2.5303323 respectively.

We can see that GA generated portfolio has the highest return as well as the highest risk. The balanced portfolio performs quite good. Its return is the second among the 3 sets of portfolio while its risk is the lowest. The randomly generated portfolio's return is the lowest, while its risk is the second.

If we input those values into the fitness function, the results for each portfolio is 19.7220475, 9.6604154 and 9.4047408 respectively.

The results show that GA portfolio outperforms the rest in the future data. In the meantime, the balanced portfolio performs better than random portfolio. So the evolved portfolio will definitely do the investors good on future stock investment. However, there is one question left to be answered, that is whether the balanced portfolio could outperform the random portfolio in the past data? This question is easy to answer by computing the return, risk and fitness value on the past data.

3.2.2 On past data

The return,risk and fitness value of evolved portfolio on past data are 22.4820544, 9.9069063 and 11.235459 respectively.

The return,risk and fitness value of balanced portfolio on past data are 13.7976083, 3.7708024 and 6.8975034 respectively.

The return,risk and fitness value of random portfolio on past data are 13.5855895, 3.7867598 and 6.7915086 respectively.

3.2.3 Some observations on the performances.

We can see a rather similar pattern of behaviours from the 3 portfolios on past data as we had seen on the future data. The evolved portfolio still does the best on return while takes the highest risk as well. Whereas the balanced portfolio returns slightly higher than random portfolio and takes the lowest risk. The random portfolio returns the least and takes the medium risk.

It looks like our evaluation function cares less about the risk and devotes itself fully to generating the greatest return. In the following section we are going to verify our hypothesis and if our hypothesis stands, then we are to tune our evaluation function so that it totally favours risk or return and thus result in showing different behaviours of weighting the stocks.

4. Different balances of return and risk

4.1 Analysis on the behaviour of our evaluation function

In the previous section we saw that our evaluation function seems to prefer heavily on return and totally disregards the risk. Let's have a closer look at the numbers as it enables us to feel it more intuitively.

The data we are looking at comes from the past(2019-2021).The following table shows the corresponding returns, risks and weights from the 8 chosen stocks.

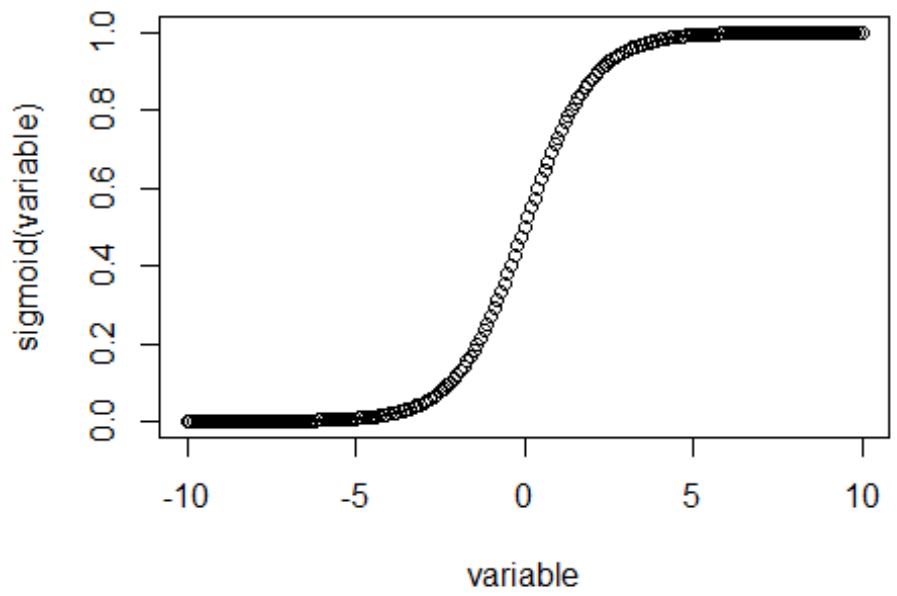
```
## [1] "Return     Risk     weight percentage"

##   pretty_mean pretty_va  weight_per
## 1   2.2533056 0.9997353 99.35584777
## 2   1.3691276 1.0290443  0.14881284
## 3   1.3850390 0.6851059  0.02758692
## 4   0.6727778 1.2725331  0.03868940
```

```
## 5   1.3570887 0.9638465  0.19277605
## 6   0.3176434 1.6735096  0.01004761
## 7   1.7776468 0.4810098  0.17543373
## 8   1.9054577 0.8019283  0.05080569
```

From the table, it is clear that our evolved solution weights heavily on stock 1. The percentage is over 99 which is dominating. Looking at the return and risk data of stock 1, it draws our attention that the return of stock 1 is the highest and it has a medium risk.

So far it is intuitively verified that our evaluation function focuses hugely on return as it attributes great weight on the stock with the highest return. As for the risk, we are still not sure how it is related to our function. Obviously this does not look like a good portfolio as it is so imbalanced. In order to tackle this problem,we are going to tweak our function in the following section to reach the effect of extremely return-focused and risk-focused respectively, and function lies somewhere in between who takes into account both risk and return.

Looking first at the sigmoid function we used earlier.



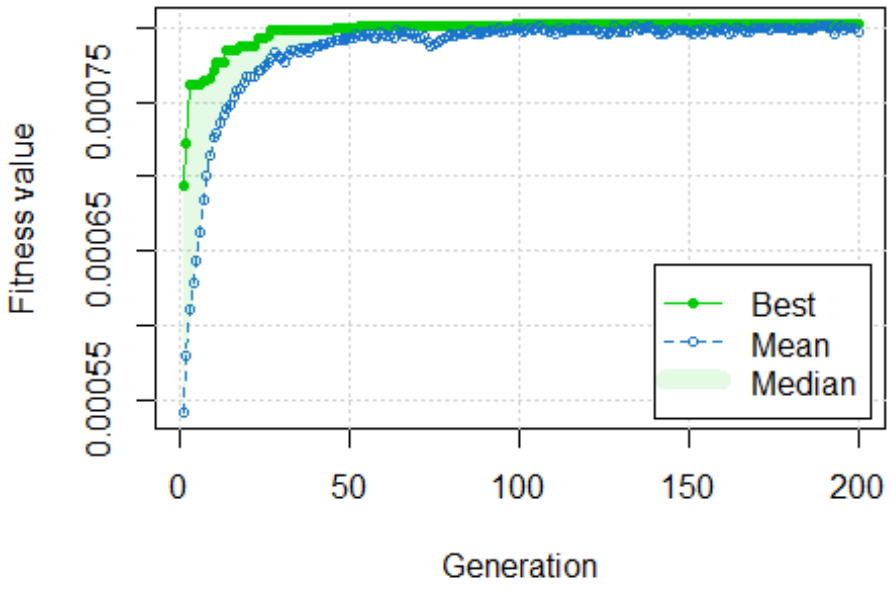
We can easily see that the derivative of sigmoid hits its maximum at 0. In fact, it can be easily calculated that the derivative of sigmoid function f(x) is f(x)(1-f(x)), which maximises at 0 with the value of 0.25. Simply consider return and risk value as variables, we can see that the evaluation function is linear to return but inverse quadratic to risk, which means that it is more sensitive to risk. This roughly gives us a hint that by tuning the coefficient of risk, for example simply increase the sum of

weighted covariance multiple times, the evaluation function would be more sensitive to risk and thus favours the less risky portfolios.

4.2 Tweaking the evaluation function.

4.2.1 Safe evaluation function(coefficient set as 1000)

```
set.seed((5))
safe_eval <- function(z)
  { weights <- z/sum(z)
  Target_mean <- sum(weights*my_mean)
  Risk <- 1000*sum(CovMatrix*outer(weights,weights))
  return(Target_mean/(1+exp(Risk)))

}
safe_GA <- ga(type = "real-valued", fitness = safe_eval,
          lower = c(0,0,0,0,0,0,0,0),
          upper = c(100,100,100,100,100,100,100,100),
          pcrossover = 0.6,
          pmutation = 0.05,
          popSize = 100,
          maxiter = 200
          )
plot(safe_GA)
```

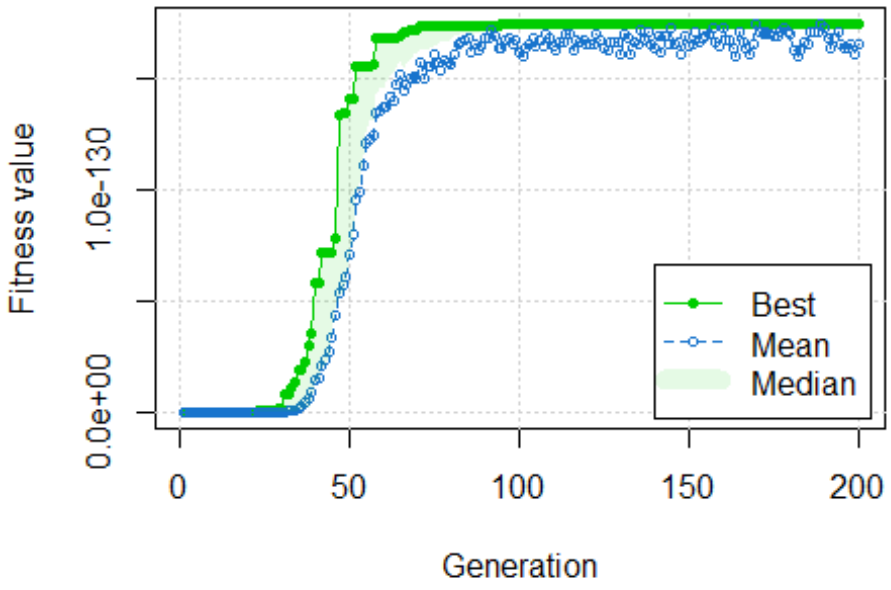



```
as.vector(safe_GA@solution)
```

```
## [1] 90.689333897  0.010995746  0.012902741  0.008696936  0.035628757
## [6]  0.058322407 91.849621077 67.026426463
```

We observe that this time GA places a considerable weight on stock 8, and stock 1 remains to be dominated among the stocks. Recall from the previous table, we know stock 8 has the lowest risk value. This indicated that our tweak is working and changing the evaluation function so that it now takes into consideration the risk. Let's experiment more by adding more weight to risk in the evaluation function and see what solution it would produce.

4.2.2 Even Safer evaluation function(coefficient set as 1000000)




```
## [1] 1.363527e+01 4.206816e-06 3.444519e+01 7.629005e+00 4.161732e+01
## [6] 5.212914e+00 9.808645e+01 4.784201e+01
```

This time it is more obvious that the solution favours stock 7 the most. This is because it has the lowest variance of 0.48 and by increasing the coefficient of 'Risk', now the evaluation function favours a lot more stock 7 than previously. So the hypothesis has been proven intuitively. The detailed derivative is provided below.

$$\frac{\partial f}{\partial w_i} = \frac{(x_i(1 + exp(\Sigma\Sigma w_i w_j \sigma_{ij}) - C * exp(C * \Sigma\Sigma w_i w_j \sigma_{ij}) * \Sigma w_i x_i * (C * \Sigma w_j \sigma_{ij})}{\left(1 + exp(C * \Sigma\Sigma w_i w_j \sigma_{ij})\right)^2}$$

where$x_i$is the i-th value of the mean daily return for stock i,$w_i$is the weight for that stock,

σ_{ij} is the covariance of stock i and stock j, C is the constant with which we tune.

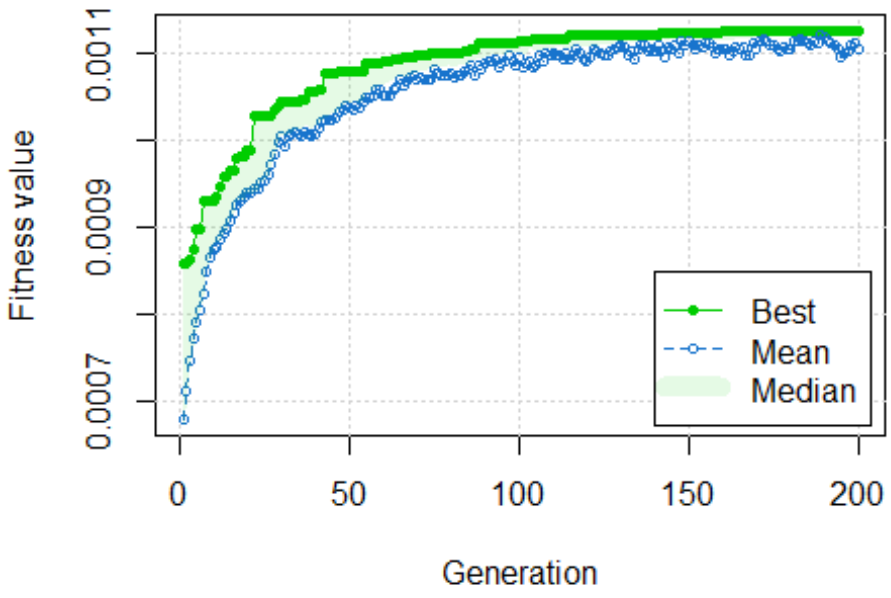
As we can see, the 'safer' GA behaves like the step function, which is very commonly seen when we tune the weight of neural nets extremely high. Also this risk-averse evaluation function converges slower than risky ones, which fits well with our intuitive.

To dive into this intuitive, we can see that C is very closely related to the covariance terms. By tuning on parameter C, the partial derivative of the function changes accordingly, and those stock that has a larger variance thus has a large derivative, which means the changing rate of their weight would impact on the fitness value greater. The fitness function would receive more feedback when we tune on the weight of the stock with larger variance.

Notice that our sigmoid function for risk has a positive symbol before the variable rather than normal sigmoid with a negative symbol. This is because we want to the negative covariance sum to stand for safe, so we want to map large negative covariance to 1, that's why I have changed the symbol here. The derivative would be a little different in terms of symbol. However that does not invalidate our findings about the effect of tuning C.

Since we now know that C decides whether the fitness function would prefer stock with large variance or not, we are totally entitled to change it as large as possible or as little as possible but no smaller than 0 as we do not wish to map positive covariance sum to negative, to see the impact.

4.2.3 Very risky evaluation function(coefficient set as 0)




```
## [1] 93.657376212  0.003335906  0.040244907  0.018351095  0.098653854
## [6]  0.010113900  0.125780705  0.003221354
```

We can see that when C is set to 0, the evolved portfolio weights almost all on stock 1, which shows complete favour of return and ignores risk.

Let's examine how all 6 portfolios perform by looking at their returns, risks and fitness values both on past data and future data.

4.2.4 Performance of 6 portfolio on past data

For simplicity, I only show the results here. I have multiplied 10^4 with them for easier inspection.

```
## [1] 3.632049e-01 4.403725e-05 5.167464e-05 3.483067e-05 1.426909e-04
## [6] 2.335775e-04 3.678518e-01 2.684365e-01
```

Safe portfolio performance: return: 19.8423974, risk: 3.8680778, fitness: 8.025972

Safer portfolio performance: return: 16.3893522, risk: 2.9236618, fitness: 8.1934782

Risky portfolio performance: return: 22.5079553, risk: 9.9507115, fitness: 11.2483784

Originally evolved portfolio performance: return: 22.4820544, risk: 9.9069063, fitness: 11.235459

Balanced portfolio performance: return: 13.7976083, risk: 3.7708024, fitness: 6.8975034

Random portfolio performance: return: 13.5855895, risk: 3.7867598, fitness: 6.7915086

4.2.5 Performance of 5 portfolio on future data

Similar results can be obtained on future data as well.

Safe portfolio performance: return: 14.6367711, risk: 4.3281258, fitness: 7.3168018

Safer portfolio performance: return: 10.0441727, risk: 2.9645559, fitness: 5.0213419

Risky portfolio performance: return: 39.5576818, risk: 11.427194, fitness: 19.7675401

Originally evolved portfolio performance: return: 39.4665219, risk: 11.3649762, fitness: 19.7220475

Balanced portfolio performance: return: 19.3231923, risk: 2.4442566, fitness: 9.6604154

Random portfolio performance: return: 18.8118616, risk: 2.5303323, fitness: 9.4047408

4.2.6 Some observations

Everything is well within our expectation that the riskier the portfolio is,the more it focuses on return and ignores risk,and vice versa. However there is one thing to note that the risky weights outperforms originally evolved weights on the initial fitness function by just a little on both past and future data.

5. Integrated GA approach for selecting assets and evolving the weights.

In this second part of the report, I am going to build a large pool of assets from which we select some good ones to work on. This part would involve using GA twice, one for selecting the assets, another for evolving the weights.

5.1 Building the pool of assets and pick 10 good ones among them.

For this part we randomly select 50 stocks from Shanghai Stock Exchange(SSE) and subsequently cherry pick some on the criteria that the chosen stocks would ideally give rather generous returns and bring less risks to the investors. First we scrape all the company code from the website
https://topforeignstocks.com/indices/components-of-the-shanghai-composite-

index/. We randomly pick 42 out of 1542 stocks and name them from 1 to 42 for simplicity and bring in the 8 stocks from part 1 to form a pool of 50.

5.2 Selecting stocks from the pool.

To cull the desired stocks, we define a criteria on which they are chosen. Overall, we want the stocks to give us the most award while they remain stable, so we use the ratio of the sum of all chosen stock's average daily returns and the sum of their covariances to decide their fitness. We use binary GA to maintain balanced weights at this stage as we do not wish to add any bias or coincidence to the portfolio. We also limit the number of chosen stocks to be at maximum 15.

5.3 Performance comparisons with evolved portfolio in part1.

The selected portfolio's performance is shown below: return : 30.1285903, risk: 680.2519843, fitness: 218.2817521

Recall from section 3.2.2 we got the following result: 'The return,risk and fitness value of evolved portfolio on past data are 22.4820544, 9.9069063 and 11.235459 respectively.'

Note that the risk of 15 stocks is dramatically great. This is because the covariance matrix is of different size from part1 and we have done sigmoid transformation on it as well. Thus the risk value offers no value for comparison. The fitness values could not be compared directly due to the same reason. The return is considerably higher than part1, which is reasonable and possible enough as the more stocks we have at hands, the larger chance we stand of being able to select those that offer higher returns. In this case the average return increases by almost 50% from 22 to 30 and this is only a balanced portfolio! let's try to optimise this portfolio to see how good result it can achieve. Since risk can't be directly compared, we focus solely on return in both scenarios where we prioritise return and risk, respectively.

5.4 Evolving portfolio from part2 to acquire risk-averse and profit-focused portfolios.

5.4.1 Safe portfolio

The selected stocks' mean returns and variances are shown for inspection and performance analysis.

```
## [1] "Return     Risk"

##    prett_mean  prett_va
## 1    2.456193 1.0429677
## 2    2.993239 0.9288429
## 3    6.100781 1.6834548
## 4    1.197824 0.9658506
## 5    3.780839 1.0874306
## 6    1.558894 1.5447475
## 7    1.661793 1.0177489
## 8    5.829659 1.3581345
```

```
## 9    1.020085 1.0319248
## 10   1.641235 1.0214442
## 11   6.402675 1.7658426
## 12   3.903159 0.9269616
## 13   1.068389 2.2164085
## 14   3.224579 0.7394170
## 15   2.353542 0.8067707

##  [1] 48.26889837  0.84820014 33.05858376 78.77110302 47.21836181  0.
01150352
##  [7] 10.16189166 46.75938439 51.91486116  0.01019861 23.90624016 31.
81996873
## [13]  0.01650251 75.94927772 99.52837933
```

Return: 30.9088596. This safe portfolio weighs the most on stock 15, who has the second lowest variance, yet still offers a higher return than balanced one. By just looking at the return from a safe portfolio, we are very optimistic about the return from an aggressive one!

5.4.2 Aggressive portfolio

```
##  [1]  0.45553543  0.56298391 76.97477187  0.12655815  0.46387940  0.
16742823
##  [7]  0.18760348  1.62381696  0.28062348  0.06183288 96.35361432  0.
31167364
## [13]  0.40953308  0.53591170  0.24426101
```

Return: 61.8494187. The result is of no surprise. This time the aggressive portfolio weighs the most on stock 11, whose mean return is 6.4, the highest of 15 stocks.

5.5 Final temporary conclusion.

From the last section, we can conclude that by choosing from a larger pool of stocks and increasing the size of portfolio, higher return could very possibly be received. The risk is somehow unable to be measured numerically here due to the structure of function. However by looking at the risk table, we can see that the highest variance is 2.21 of stock 13. Comparing with the variances of stocks from part1, whose highest is 1.67, although it looks riskier,in fact, stock 13, has not been placed considerable weights in both scenarios. Therefore it is safe to say that this portfolio of 15 stocks is not very likely to be riskier than the previous one.