1.0 INTRODUCTION

In a previous report Genetic Algorithm was used to optimise some company stocks collected in a portfolio. The purpose was to investigate the performance of R based Genetic algorithm package Genalg in comparison to another package with good optimisation ability. It was observed that Genalg produce a portfolio performance almost similar to that of the efficient frontier. Likewise this study is about the use of Differential Evolution algorithm for stock portfolio optimisation. The optimisation abilities of R based Deoptim package will be investigated and compared to Genalg and Efficient frontier since Genalg had proven itself to have good portfolio optimisation abilities.

2.0. BACKGROUND TO THE PROBLEM

2.1.1. PROBLEM DEFINITION

Out of all the concepts of finance and investment, the concept of portfolio theory which involves the optimisation of a portfolio of assets has always been very important. It is a concept regarding investment decision which if not care is not taken can result in a huge loss. Therefore an investor or a fund manager is usually faced with the task of deciding on the best portfolio option to invest funds. The investor or manager needs a thorough comparative study to decide the best possible option. This comparative study provides the investors with a view of the expected returns and risks contained in a portfolio. The best possible option is the portfolio that whose return is maximum and the risk is minimum sometimes called the optimum portfolio. The problem of finding an optimum portfolio of a combination of assets can be dated back in time.

2.2. DIFFERENTIAL EVOLUTION

Differential Evolution is was put forward by Storn and Price in 1997. Storn and Price (1997) defined Differential Evolution as a search heuristic. Differential Evolution algorithm is one of several evolutionary techniques that makes use of biological concepts of mutation, selection and crossover on a parent population to over successive generations to evolve a set of best offspring that suit a particular problem (Holland, 1975). However unlike other evolutionary algorithms, Differential Evolution uses floating for the chromosomes of its

population instead of binary system and it only uses arithmetic operators excluding logical operators during mutation (Ardia et al, 2011). DE can be used to solve multi objective, non-linear and non-continous problems (Pant and Zaheer, 2015). A pseudo code of Differential Evolution gotten from the work of Pant et al is provided in the Appendix (Fig. 1.1.).

Differential Evolution as been applied to optimisation problems such as image processing, supply chain management, integer optimisation and share portfolio optimisation (Ansari et al, 2015; Jauhar et al, 2014; Korczak et al, 2000). Differential evolution algorithm is applied directly to portfolio optimization problems.

2.3. DEOPTIM

Differential Evolution is not easily applied directly to optimisation problems. A package known as DEoptim was developed by David Ardia et al in 2005 to solve the non-convex problems. DEoptim package has a main function known has DEoptim.

```
DEoptim (fn, lower, upper, DEoptim.control,NP,Trace,Itermax,CR)
```

- Fn: the objective function to be optimized. The argument of the objective function is a vector of the parameters to be optimised. The objective function returns a single scalar value.
- Lower and Upper: two vectors specifying the lower and the upper bound constraints on the parameters to be optimised.
- DEoptim.Control: a list of tuning parameters.
- NP: it defines the size of population members. The default is 50 member. It is advised by the creators to set NP to be at least 10 times the length of the parameter vector.
- Trace: it indicates whether to print the process or not. If its True the iteration process is printed.
- Itermax : the maximum iteration (population generation) allowed. if not set it defaults to 200.
- CR: crossover probability from interval [0, 1]. It defaults to 0.9.

```
DEoptim.control(VTR = -Inf, strategy = 2, bs = FALSE, NP = 50,
        itermax = 200, CR = 0.5, F = 0.8, trace = TRUE,
        initialpop = NULL, storepopfrom = itermax + 1,
  storepopfreq = 1, checkWinner = FALSE, avWinner = TRUE, p = 0.2)
```

Figure 2.2. a sample of DEoptim.control's tuning parameters

2.4. DATASET

The assets used for the study were gotten from Yahoo finance. The portfolio of interest in for the study is a collection of 40 assets of different forms. The Assets comprised of some stock index, company stocks, and funds. The mean returns of each asset over the period observed once per month between 2006-02-01 and 2015-12-01 is given below (figure 2.3.).

```
              WDC      WY    WHR     WMB    WEC     XEL
Mean Return 0.016 0.0079 0.015 0.0095 0.011 0.0096

               XRX  XLNX     ZION     MMM     ABT  ADBE
Mean Return 0.0032 0.009 2.4e-05 0.0099 0.0099 0.011

                AMD    AET     AFL    APD    ARG       AA
Mean Return -0.0083 0.012 0.0094 0.011 0.016 -0.0019

              AGN  ALTR     MO     AEP     AXP    AIG    AMGN
Mean Return 0.021 0.013 0.016 0.0085 0.0088 0.011 0.0099

               APC     ADI    AON      APA  AAPL    AMAT
Mean Return 0.0051 0.0076 0.011 0.00075 0.025 0.0051

               ADM      T ADSK       M     FTR      SO
Mean Return 0.0062 0.008 0.01 0.0086 0.0026 0.0071

                FE     SPY     EFA
Mean Return 0.0021 0.0066 0.0034
```

Figure 2.3 The mean return of each asset over 10 years

3.0 EXPERIMENTATION

3.1. ARCHITECTURE OF THE OPTIMISATION

The assets to be optimised were collected together as a vector. The monthly returns of the assets for 10 years from 2006 to 2016 were gotten from Yahoo Finance using the getReturns method of StockPortfolio package.

```
tickers < − c("WDC" ,"WY" , "WHR" , "WMB" , "WEC" , "XEL" ,
"XRX" , "XLNX" , "ZION" ,"MMM",  "ABT", "ADBE" , "AMD" ,
"AET" , "AFL" , "APD" , "ARG" ,"AA" , "AGN" , "ALTR" , "MO" ,
"AEP" , "AXP" , "AIG" , "AMGN" , "APC" ,"ADI" , "AON" ,"APA",
"AAPL" , "AMAT" ,"ADM" , "T", "ADSK", "M", "FTR", "SO", "FE",
"SPY",  "EFA")


getReturns (tickers, start="2006-01-01", end="2016- 03-31",
freq="monthly" )
```

Fig 3.1 The vector of assets

DEoptim function in the DEoptim package to evolve the set of weights needed to generate the optimal portfolio. Sharpe ratio was used in then objective function to select the best set of solutions in after every crossover stage. The objective function contains a function that normalises each evolved population solution. The optimal the weights of the assets in the portfolio.

```
out < − DEoptim(fn = obj, lower = rep(0, 40), upper = rep(1, 40),
DEoptim.control (NP = 400, itermax = 200, trace = TRUE,CR= 0.2))
```

```
obj < − function(x) {
          neww < − x/sum(x)
          portfoliomean < − dataset$Stockaverage %*% neww
          covmatrix < − cov(AssetsData$R)
          stdev< − sqrt(sum(neww * colSums((covmatrix * neww))))
          sharpe < − (portfoliomean / stdev)
          return(-sharpe)
        }
```

Figure 3.2. Objective function

4.0. RESULTS

4.1. RESULTS FROM DEOPTIM

After 200 hundred iterations the DEoptim method produced a solution with a sharpe ratio of -0.3774001 with a set of weights given in figure weights but the downside is that the sum of the weights is more than one. This is above the constraint of 1 placed on it.

```
>out$optim$bestval
  -0.3774001
```

```
> out$optim$bestmem
          par1        par2        par3        par4        par5
0.039200038 0.002942725 0.015127409 0.007571565 0.921140616
          par6        par7        par8        par9       par10
0.920085289 0.004903479 0.103240499 0.033927549 0.260410759
         par11       par12       par13       par14       par15
0.862704154 0.044015065 0.002359261 0.160219541 0.011942433
         par16       par17       par18       par19       par20
0.359085951 0.486379150 0.052921228 0.927981005 0.269501664
         par21       par22       par23       par24       par25
0.798667040 0.588131296 0.008937167 0.020283530 0.149590437
         par26       par27       par28       par29       par30
0.026028549 0.084592957 0.620672125 0.020747149 0.969333555
         par31       par32       par33       par34       par35
0.001338856 0.096326110 0.023995712 0.029945325 0.015800417
         par36       par37       par38       par39       par40
0.023562968 0.688076448 0.059187011 0.144852031 0.021012568
Sum of weights equals 9.876741
```

Figure 4.1 weights assigned to the assets in the portfolio

Since the weights do not sum up to one. The generated weights were normalised by dividing each weight by the total of the summing of all the weights.

> DEbest = out$optim$bestmem

> DEnormbest< − DEbest/sum(DEbest)

The new weights generated as a result of the normalization summed up to one. The new weights are provided below in figure 4.2. Given these weights, the investment is expected to yield return of 0.01294388 with portfolio risk of 0.03429751

```
> sum(DEnormbest)
[1] 1
```

> DEoptimalportfolio

DEoptimalmean	DEoptimaldev	DEoptimalsharpe
0.01294388	0.03429751	0.3774001

```
> DEnormbest
       par1         par2         par3         par4         par5
0.0039689245 0.0002979450 0.0015316195 0.0007666056 0.0932636232
       par6         par7         par8         par9        par10
0.0931567734 0.0004964673 0.0104528916 0.0034350957 0.0263660623
      par11        par12        par13        par14        par15
0.0873470496 0.0044564363 0.0002388704 0.0162219043 0.0012091472
      par16        par17        par18        par19        par20
0.0363567258 0.0492449046 0.0053581672 0.0939561987 0.0272864980
      par21        par22        par23        par24        par25
0.0808634214 0.0595471034 0.0009048700 0.0020536664 0.0151457290
      par26        par27        par28        par29        par30
0.0026353379 0.0085648657 0.0628417965 0.0021006069 0.0981430606
      par31        par32        par33        par34        par35
0.0001355565 0.0097528237 0.0024295173 0.0030319035 0.0015997603
      par36        par37        par38        par39        par40
0.0023857028 0.0696663478 0.0059925650 0.0146659750 0.0021274800
```

Figure 4.2. normalised weights of funds to invest in the portfolio

4.2. RESULTS FROM GENETIC ALGORITHM

4.2.1. Genalg Architecture

Similar to DEoptim the assets where collected in as a vector called tickers and the monthly returns for the assets where gotten from Yahoo finance using the same getReturns method from the StockPortfolio package. An average of the ten year data was derived using the summary syntax of R.

```
getReturns (tickers, start="2006-01-01", end="2016-03-31", freq="monthly" )
```

Sharpe ratio was used in the objective function to help with the task of maximising the return and minimising the portfolio risk generated by each set of evolved weights in the mutation process. Each set of weight offspring is normalised at every step of the evolution process in the objective function

```
evalFunc1 <- function(x) {
  neww <- x/sum(x)
  portfoliomean <- dataset$Stockaverage %*% neww
  covmatrix <- cov(AssetsData$R)
  stdev<- sqrt(sum(neww * colSums((covmatrix * neww))))
  sharpe <- (portfoliomean / stdev)
    return(-sharpe)
}
```

Figure 4.1 RBGA objective function

The main genalg function that performs the optimisation is known as RBGA. It accepts the objective function as an evaluation function in the mutation process as it acts as a constraint in the selection of evolved offsprings based on their performance in generating the sharpe ratio. The mutation chance was set to 0.01 meaning that only 1 percent of the best performing offsprings will be selected for the next mutation stage. The weight constraint was set to a minimum of 0.0 weight and a maximum of 1 for each asset. The population size and number of iteration to be performed by rbga were set to 300 and 150 respectively.

```
rbga (stringMin=c(rep(0.0,length(v))), stringMax=c(rep(1,length(v))), popSize = 300, iters = 150, mutationChance = 0.01,  evalFunc = evalFunc1, verbose=TRUE)
```

4.2.2. Genalg Result

The best weights evolved by rbga after 150 iterations summed up to 4.618473. two of these weights were even as high as 0.9 see figure 4.2 below. However after normalising the weights the sum of the new weights equalled one. The expected returns derived from the new weights was 0.01480029 while the portfolio risk was 0.03342009

```
> best <- getBest(GAmodel)
> sum(best)
[1] 4.618473
> normbest <- best/sum(best)

> sum(normbest)
[1] 1

> optimalportfolio

optimalmeanGA      optimaldevGA     optimalsharpeGA
  0.01480029           0.03342009          0.4428562
```

```
> best
          [1] 0.0003497405 0.0004653491 0.0082507152 0.0271446577
          [5] 0.9748212437 0.0315195093 0.0311471326 0.0149125187
          [9] 0.0103411078 0.0175474968 0.0834730300 0.0055596181
         [13] 0.0043652093 0.0127473976 0.0261734172 0.0055120930
         [17] 0.2021992868 0.0051863834 0.9881857899 0.0115346890
         [21] 0.9567352179 0.0131269026 0.0189908426 0.0010141998
         [25] 0.0600391971 0.0039156850 0.0012723685 0.1147499924
         [29] 0.0257151260 0.3933506748 0.0167653693 0.1281806957
         [33] 0.0142105434 0.0123871525 0.0022619888 0.0271343600
         [37] 0.3289520625 0.0062614635 0.0281590024 0.0038137583
```

Figure 4.2 rbga optimised weights

```
> normbest
          [1] 7.572644e-05 1.007582e-04 1.786460e-03 5.877410e-03
          [5] 2.110700e-01 6.824660e-03 6.744033e-03 3.228885e-03
          [9] 2.239075e-03 3.799415e-03 1.807373e-02 1.203778e-03
         [13] 9.451629e-04 2.760089e-03 5.667115e-03 1.193488e-03
         [17] 4.378055e-02 1.122965e-03 2.139637e-01 2.497511e-03
         [21] 2.071540e-01 2.842260e-03 4.111931e-03 2.195963e-04
         [25] 1.299979e-02 8.478311e-04 2.754955e-04 2.484587e-02
         [29] 5.567885e-03 8.516899e-02 3.630068e-03 2.775391e-02
         [33] 3.076892e-03 2.682088e-03 4.897698e-04 5.875180e-03
         [37] 7.122529e-02 1.355743e-03 6.097037e-03 8.257617e-04
```

Figure 4.3 normalised weights

4.3 RESULT FROM EFFICIENT FRONTIER

4.3.1. Efficient Frontier Architecture

4.3.2. Efficient Frontier Result

The results by efficient frontier produced an expected return of 0.01620446
Portfolio risk of 0.475403 and sharpe ratio of 0.03408574 as shown in figure 4.5. The weights generated are provided in figure 4.4.

```
> eff[eff$sharpe==max(eff$sharpe),]
```

Std.Dev	Exp.Return	sharpe
0.03408574	0.01620446	0.475403

```
> eff.optimal.point
                WDC              WY             WHR             WMB
-1.053015e-17 -2.384654e-17 3.758698e-18 -6.863246e-18
                WEC             XEL             XRX            XLNX
 0.2472219     1.079292e-16 1.304903e-17 -2.594726e-17
               ZION             MMM             ABT            ADBE
 8.641191e-19  1.036063e-16 -3.295998e-17 2.337646e-17
                AMD             AET             AFL             APD
 6.054292e-18 1.843338e-17 -1.153148e-17 1.158781e-16
                ARG              AA             AGN            ALTR
  0.05252853    -4.755821e-17 0.2979635 6.065262e-19
                 MO             AEP             AXP             AIG
 0.2640863     1.485743e-16 -4.118567e-17 1.904268e-18
               AMGN             APC             ADI             AON
 2.593225e-17    6.636552e-18 6.103134e-17 0.03425198
                APA            AAPL            AMAT             ADM
-1.033139e-17    0.05700632 5.091484e-17 0.04694146
                  T            ADSK               M             FTR
 5.498646e-19  5.71484e-18 1.691842e-17 9.019547e-17
                 SO              FE             SPY             EFA
  5.669574e-16     -6.678158e-17 -1.157532e-16   0
```

Figure 4.4 weight from efficient frontier

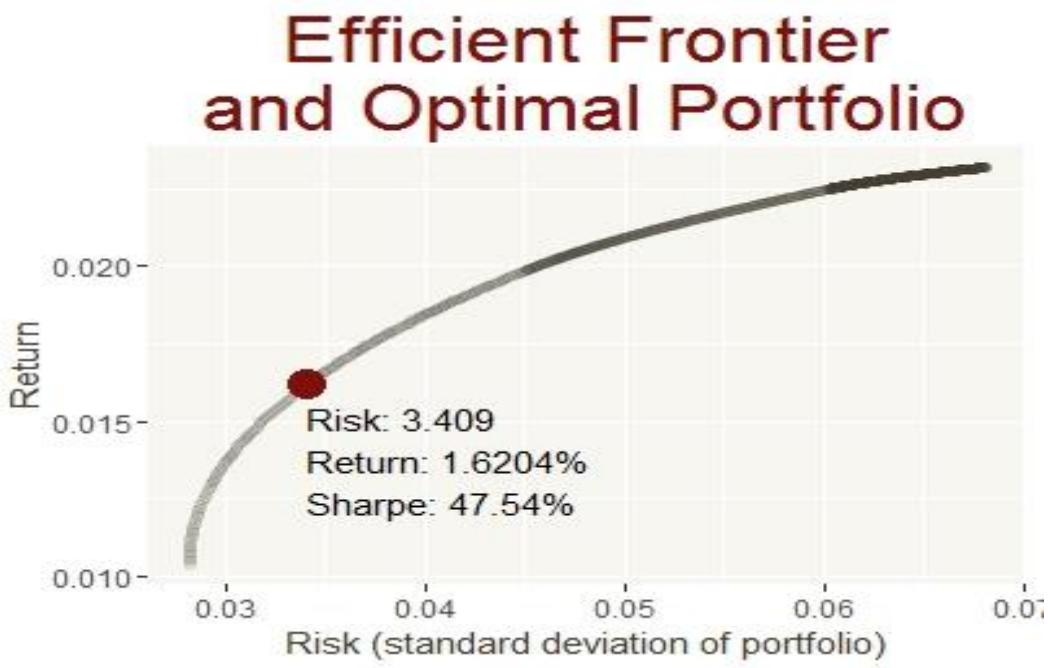

Figure 4.5 graph of the optimal portfolio

4.4. COMPARISON OF DEOPTIM RESULT WITH GENETIC ALGORITHM AND EFFICIENT FRONTIER

Comparing the performance of all three packages for their optimisation abilities, it was observed that the efficient frontier produced the highest expected returns with portfolio risk even lower than that produced by Deoptim. DEoptim generated the least expected return from its optimal weights or so called "best solution". In terms of minimisation of portfolio risk, Genalg package had the best performance producing portfolio risk as low as 0.03342009. unlike the weights produced by efficient frontier the weights given by Genalg and DEoptim still required normalisation to sum up to one.

DEoptim		Genalg		Efficient frontier	
return	risk	return	risk	return	risk
0.01294388	0.03429751	0.01480029	0.03342009	0.01620446	0.03408574

5.0 CONCLUSION

DEoptim was used to allocate weights to 40 assets in an investment portfolio. The even with the normalisation function included in the objective function, the total allocation of the resulting weights was still more than one which further required the result of allocated weights to be normalised. Even after normalisation, the result from the Deoptim allocation of weights did not produce a portfolio with performance that is good enough to compete with Genalg and Efficient frontier. However it can be seen that Differential evolution is very similar to genetic algorithm except that it does not consider its candidate solutions as chromosomes in form of binary strings but as vectors of floating points.

REFERENCES

Ansari, I.A., Pant, A., Ahn, C.W. (2016) Robust and false positive free watermarking in IWT domain using SVD and ABC. Engineering Applications of Artificial Intelligence. 49, 114-125

Ansari, I.A., Pant, A., Ahn, C.W. (2015) SVD based fragile watermarking scheme for tamper localization and self-recovery. International Journal of Machine Learning and Cybernetics. 1-15,10.1007/s13042- 015-0455-1

Ardia, D., Boudt, K., Carl, P., Mullen, K., & Peterson, B. G. (2011). Differential Evolution with DEoptim: An application to non-convex portfolio optimization.The R Journal, 3(1), 27-34.

Korczak, J., & Roger, P. (2000). Portfolio Optimization using Differential Evolution. Prace Naukowe Akademii Ekonomicznej we Wrocławiu, (855), 302-319.

J. H. Holland. Adaptation in Natural Artificial Systems. University of Michigan Press, 1975.

Jauhar, S.K., Pant, M.: Genetic algorithms, a nature-inspired tool: review of applications in supply chain management. In: Proceedings of Fourth International Conference on Soft Computing for Problem Solving, pp. 71–86. Springer India, Jan 2015

Jauhar, S.K., Pant, M., Abraham, A. (2014) A novel approach for sustainable supplier selection using differential evolution: a case on pulp and paper industry. In: Intelligent Data analysis and its Applications, vol. II, pp. 105–117. Springer International Publishing.

Storn, R. and Price, K.(1997) Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11:341–359.

APPENDIX

/* **Initialize parameters** */
Set mutation scale factor F ϵ [0, 2],
crossover parameter CRϵ (0,1),
population size NP, counter g =0
/* **Initialize a population** */

While stopping criteria
do g=g+1
for i=1 to NP do
/***Differential Mutation** */
Select randomly 3 individuals
X^g_{p1}, X^g_{p2}, $X^g_{p3} \epsilon$ p^g
$V_i = X^g_{p1} + FX (X^g_{p2} + X^g_{p3})$
p1,p2,p3 ϵ { 1,2,….NP}/{i}
/* **Probability Binomial Crossover** */
Generate randomly J_{rand} ϵ {1,2,…..,D}
For j=1 to D **do**
generate randomly rand (0,1)
if rand(0,1) $\leq$ CR or j=j_{rand}
$u1^{j,g} = x^j_i$
end
end for

/***Greedy Selection** */
if $f(u_i^g) < f(X^j g)$
else
$u1^{j,g} = x^g i$
end if
end for
end while

Figure 1.1. Pseudo code for DE