# Portfolio Optimization Using GAs

## Construction of a Portfolio Using The GA Package

We start by importing the libraries that we need. We will import *quantmod* to get the stocks data, *GA* for evolutionary algorithms, and *xts* to handle time-series data.

```r
library(quantmod)
library(GA)
library(xts)
```

We need to select 10 stocks of companies in different sectors to invest in. We will select 2 companies from the technology, finance, retail, automobile, and oil and gas sectors because they are sectors that the UK is very active in. We select the data produced during the period from the beginning of 2019 to the end of 2022 as training data, and calculate the mean daily returns and variance.

```r
# Tickers of the different companies
tickers <- c("GOOG", "AAPL", "GS", "MS", "TSCO",
             "JSAIY", "TSLA", "VWAGY", "BP", "SHEL")

# Use the tickers to get the data from Yahoo Finance
train_data <- lapply(tickers, function (x) {
  na.omit(getSymbols(x, src = "yahoo", from = "2019-01-01",
                     to = "2022-12-31", auto.assign = FALSE))
})

# Adjust the dataset to be able to work with it
names(train_data) <- tickers
adjustedPrices <- lapply(train_data, Ad)
adjustedPrices <- do.call(merge, adjustedPrices)

# Get the daily returns of the stocks
rets <- lapply(adjustedPrices[-1], dailyReturn)
rets <- do.call(merge, rets)
colnames(rets) <- tickers

# Get the mean return and covariance
mu <- apply(rets, 2, mean)
sigma <- cov(rets)
```

The next thing is to define the fitness function. This is easy; we will use the Sharpe Ratio as the fitness function because it takes the return and risk into account.

```r
# Get the (annualized) returns
get_return <- function(wt) return(sum(mu * wt)*252)
```

```
# Get the (annualized) risk
get_risk <- function(wt) return(sqrt(t(wt) %*% (sigma*252) %*% wt))

# Actual fitness function (Sharpe Ratio)
fitness <- function(wt) {
  pen <- sqrt(.Machine$double.xmax)
  penalty <- max(sum(wt) - 1, 0)*pen              # Define penalty
  sharpe_ratio <- -(get_return(wt))/get_risk(wt)  # Maximize the return function
  return(sharpe_ratio - penalty)                  # Impose penalty
}
```

Now, it's time to use evolutionary algorithms to extract the optimal weights. Our parameters will be the same as those described by Lin & Gen (2007).

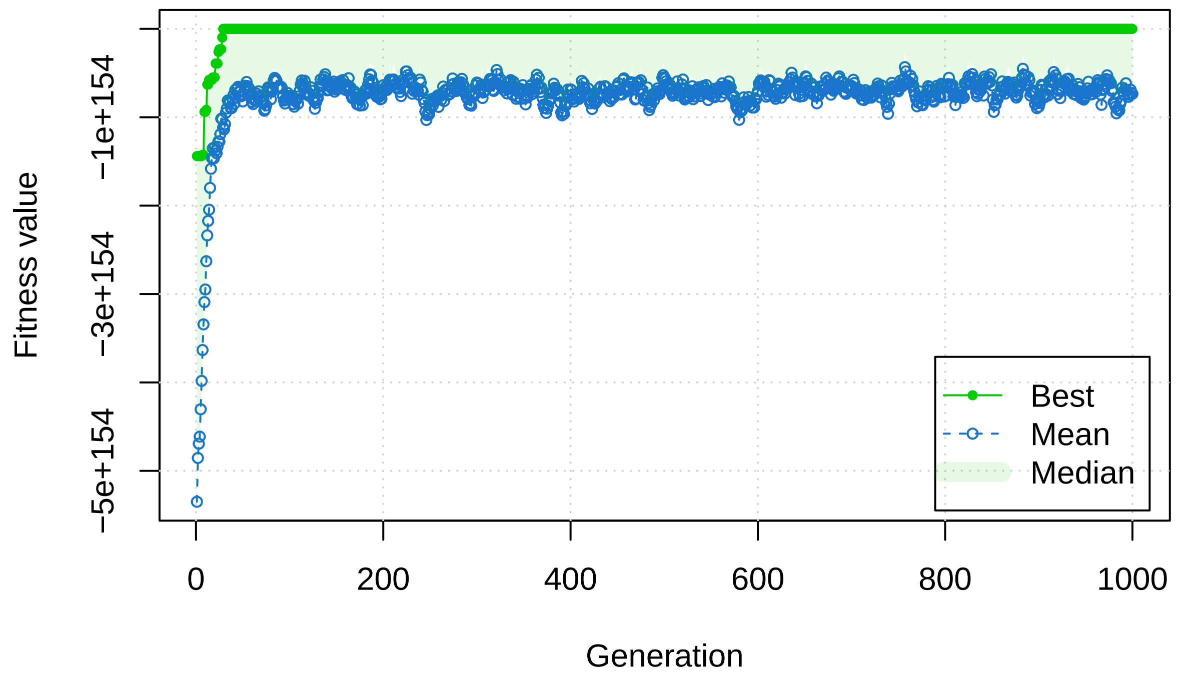
```
# Use the ga function with the parameters
ga_1 = ga(type = "real-valued", fitness = fitness,
          lower = rep(0, 10), upper = rep(1, 10), monitor = FALSE,
          popSize = 100, maxiter = 1000, pmutation = 0.5, pcrossover = 0.7)

# Extract the optimal weights
print(c("Portfolio Weights:", ga_1@solution))
```

```
##  [1] "Portfolio Weights:"  "0.0125900279352322"  "0.00967211415991187"
##  [4] "0.0010981471426409"  "0.0139304052572697"  "0.010060422384899"
##  [7] "0.0295975897461176"  "0.00838932414479377" "0.00364406976560785"
## [10] "0.321073983757197"   "0.576215667184442"
```

```
print(c("Portfolio Weights Sum:", round(sum(ga_1@solution), 0)))
```

```
## [1] "Portfolio Weights Sum:" "1"
```

```
plot(ga_1)
```



## Evaluation of The Portfolio on Unseen "Future" Data

We get the data for the entirety of 2023 as testing data, and evaluate the return and risk using the portfolio weights generated by the first GA.

```
# Use the tickers to get the data of 2023
test_data <- lapply(tickers, function (x) {
  na.omit(getSymbols(x, src = "yahoo", from = "2023-01-01",
                     to = "2023-12-31", auto.assign = FALSE))
})

# Adjust the dataset to be able to work with it
names(test_data) <- tickers
adjustedPrices_test <- lapply(test_data, Ad)
adjustedPrices_test <- do.call(merge, adjustedPrices_test)

# Get the daily returns
rets_test <- lapply(adjustedPrices_test[-1], dailyReturn)
rets_test <- do.call(merge, rets_test)
colnames(rets_test) <- tickers

# Get the mean returns and covariance
mu_test <- apply(rets_test, 2, mean)
sigma_test <- cov(rets_test)
```

```r
# Get optimal risk and return
print(c("Optimal Portfolio Return:",
        round(sum(mu_test * ga_1@solution)*252, digits = 3)))
```

```
## [1] "Optimal Portfolio Return:" "0.209"
```

```r
print(c("Optimal Portfolio Risk:",
        round(sqrt(sum(t(ga_1@solution) %*% (ga_1@solution %*% (sigma_test*252)))),
              digits = 3)))
```

```
## [1] "Optimal Portfolio Risk:" "0.445"
```

Generally, we see that the risk of this investment is higher than its return. Looking at the trends of the chosen stock prices in previous years, we see that the retail and energy sectors experienced a drop in stock prices, especially in the period 2020-2021. The energy sector has been particularly volatile ever since the pandemic of 2020. Additionally, while technology companies like Google and Apple experienced jumps in stock prices, their stock price trends tend to be volatile. We conclude that the values we see are normal based on the trends of historical data.

## Comparison of The Evolved Portfolio With Evenly Weighted and Random Portfolios

We generate 10 portfolios, where each portfolio consists of 10 random weights, and 10 equal weights for the stocks, and calculate the portfolio return and variance using them.

```r
# 10 Random Portfolios
for (i in 1:10) {
  rand_wts <- runif(10)
  rand_wts <- rand_wts / sum(rand_wts)
  print(c(sprintf("Random Portfolio Return %s:", i),
          round(sum(mu_test * rand_wts) * 252,
                digits = 3),
          sprintf("Random Portfolio Risk %s:", i),
          round(sqrt(sum(rand_wts %*% (rand_wts %*% (sigma_test*252)))),
                digits = 3)))
}
```

```
## [1] "Random Portfolio Return 1:" "0.36"
## [3] "Random Portfolio Risk 1:"   "0.519"
## [1] "Random Portfolio Return 2:" "0.381"
## [3] "Random Portfolio Risk 2:"   "0.537"
## [1] "Random Portfolio Return 3:" "0.241"
## [3] "Random Portfolio Risk 3:"   "0.499"
## [1] "Random Portfolio Return 4:" "0.264"
## [3] "Random Portfolio Risk 4:"   "0.486"
## [1] "Random Portfolio Return 5:" "0.265"
## [3] "Random Portfolio Risk 5:"   "0.532"
## [1] "Random Portfolio Return 6:" "0.321"
## [3] "Random Portfolio Risk 6:"   "0.553"
## [1] "Random Portfolio Return 7:" "0.333"
## [3] "Random Portfolio Risk 7:"   "0.545"
## [1] "Random Portfolio Return 8:" "0.269"
## [3] "Random Portfolio Risk 8:"   "0.519"
## [1] "Random Portfolio Return 9:" "0.317"
## [3] "Random Portfolio Risk 9:"   "0.52"
## [1] "Random Portfolio Return 10:" "0.283"
## [3] "Random Portfolio Risk 10:"   "0.531"
```

```
# Balanced Portfolio
eq_wts <- rep(1/10, 10)
print(c("Balanced Portfolio Return:",
        round(sum(mu_test * eq_wts) * 252, digits = 3)))
```

```
## [1] "Balanced Portfolio Return:" "0.281"
```

```
print(c("Balanced Portfolio Risk:",
        round(sqrt(sum(eq_wts %*% (eq_wts %*% (sigma_test*252)))),
      digits = 3)))
```

```
## [1] "Balanced Portfolio Risk:" "0.518"
```

By analyzing the returns and risks of the 10 random portfolios, the return and risk of the balanced portfolio, and comparing them all to the return and risk of the optimal portfolio, we see that the same trend exists; the risk is always higher than the return. We conclude that this is normal due to the previously mentioned reasons.

## Creation and Evaluation of Portfolios With Differently Balanced Risk and Return

We shift our attention to creating new portfolio weights based on changing the fitness function of the GA. We will create 3 portfolios: one that minimizes risk and return, one that maximizes risk, one that maximizes risk and return.

```
# Minimizing risk and return using historical "old" training data
fitness1 <- function(wt) {
  pen <- sqrt(.Machine$double.xmax)
  penalty <- max(sum(wt) - 1, 0)*pen
  f <- get_return(wt)/get_risk(wt)
  return(f-penalty)
}

ga_min_both <- ga(type = "real-valued", fitness = fitness1,
                 lower = rep(0, length(tickers)), upper = rep(1, length(tickers)),
                 monitor = FALSE, popSize = 100, maxiter = 1000,
                 pmutation = 0.5, pcrossover = 0.7)

# Print the weights
print(ga_min_both@solution)
```

```
##              x1        x2          x3         x4        x5          x6
## [1,] 0.01002896 0.3634732 0.008221003 0.02857444 0.3492632 0.005338866
##             x7          x8          x9         x10
## [1,] 0.1783374 0.003594832 0.004219796 0.007714302
```

```r
# Use the "new" testing data to get the return and risk for this portfolio
print(c("Min Both Portfolio Return:",
        round(sum(mu_test * ga_min_both@solution) * 252, digits = 3)))
```

```
## [1] "Min Both Portfolio Return:" "0.351"
```

```r
print(c("Min Both Portfolio Risk:",
        round(sqrt(sum(t(ga_min_both@solution)
                       %*% (ga_min_both@solution %*% (sigma_test*252)))),
      digits = 3)))
```

```
## [1] "Min Both Portfolio Risk:" "0.503"
```

```r
# Maximizing risk
fitness2 <- function(wt) {
  pen <- sqrt(.Machine$double.xmax)
  penalty <- max(sum(wt) - 1, 0)*pen
  f <- get_return(wt)/-get_risk(wt)
  return(f-penalty)
}

ga_max_risk <- ga(type = "real-valued", fitness = fitness2,
                  lower = rep(0, length(tickers)), upper = rep(1, length(tickers)),
                  monitor = FALSE, popSize = 100, maxiter = 1000,
                  pmutation = 0.5, pcrossover = 0.7)

# Print the weights
print(ga_max_risk@solution)
```

```
##              x1          x2         x3          x4         x5          x6
## [1,] 0.01629266 0.003134385 0.01457843 0.001386613 0.00343819 0.009883747
##              x7          x8        x9       x10
## [1,] 0.00612101 0.005419046 0.5701328 0.3623548
```

```r
# Use the "new" testing data to get the return and risk for this portfolio
cat(c("\nMax Risk Portfolio Return:",
        round(sum(mu_test * ga_max_risk@solution) * 252, digits = 3)))
```

```
##
## Max Risk Portfolio Return: 0.178
```

```r
print(c("Max Risk Portfolio Risk:",
        round(sum(sqrt(t(ga_max_risk@solution)
                       %*% (ga_max_risk@solution %*% (sigma_test*252)))),
      digits = 3)))
```

```
## [1] "Max Risk Portfolio Risk:" "2.656"
```

```r
# Maximizing both
fitness3 <- function(wt) {
  pen <- sqrt(.Machine$double.xmax)
  penalty <- max(sum(wt) - 1, 0)*pen
  f <- -get_return(wt)/-get_risk(wt)
  return(f-penalty)
}

ga_max_both <- ga(type = "real-valued", fitness = fitness3,
                  lower = rep(0, length(tickers)), upper = rep(1, length(tickers)),
                  monitor = FALSE, popSize = 100, maxiter = 1000,
                  pmutation = 0.5, pcrossover = 0.7)

# Print the weights
print(ga_max_both@solution)
```

```
##              x1        x2          x3         x4        x5          x6
## [1,] 0.02400886 0.3401212 0.006293064 0.01217412 0.4101673 0.003111759
##             x7          x8         x9         x10
## [1,] 0.1832414 0.001847854 0.01200142 0.002270259
```

```r
# Use the "new" testing data to get the return and risk for this portfolio
cat(c("\nMax Both Portfolio Return:",
        round(sum(mu_test * ga_max_both@solution) * 252, digits = 3)))
```

```
##
## Max Both Portfolio Return: 0.35
```

```r
print(c("Max Both Portfolio Risk:",
        round(sum(sqrt(t(ga_max_both@solution) %*% (ga_max_both@solution %*% (sigma_test*252)))),
      digits = 3)))
```

```
## [1] "Max Both Portfolio Risk:" "3.514"
```

We see that the trend of risk being larger than the return persists. At this point, we can begin to assume that our choice of assets was not the best one as most of the stocks display high levels of volatility.

## Using GAs to select the assets

We select 10 assets from each of the sectors we defined earlier to get a pool 50 total assets to choose from. We employ a GA to select the best 10 assets from the 50 total assets and use the weights produced from the first GA to determine the new returns.

```r
# We add 40 new tickers to the original 10 to get a pool of 50 assets to choose from
new_tickers <- c("GOOG", "AAPL", "MSFT", "NVDA", "AMZN", "META", "CRM", "BABA", "INTC", "PYPL",
                 "GS", "MS","JPM", "BAC", "WFC", "HSBC", "RY", "HDB", "SCHW", "UBS",
                 "TSCO", "JSAIY", "WMT", "TGT", "LUX", "ANF", "AEO", "URBN", "CRI", "FL",
                 "TSLA", "VWAGY", "STLA", "F", "GM", "RIVN", "VFS", "LVWR", "NWTN", "BLBD",
                 "BP", "SHEL", "XOM", "CVX", "PBR", "TTE", "COP", "EQNR", "SLB", "E")
```

```r
# Use the tickers to get the data from Yahoo Finance
new_data <- lapply(new_tickers, function (x) {
  na.omit(getSymbols(x, src = "yahoo", from = "2019-01-01",
                     to = "2023-12-31", auto.assign = FALSE))
})

# Adjust the tables to be able to work with them
names(new_data) <- new_tickers
adjustedPrices_new <- lapply(new_data, Ad)
adjustedPrices_new <- do.call(merge, adjustedPrices_new)
rets_new <- lapply(adjustedPrices_new[-1], dailyReturn)
rets_new <- do.call(merge, rets_new)
colnames(rets_new) <- new_tickers

# Get the mean return and covariance
mu_new <- apply(rets_new, 2, mean)
sigma_new <- cov(rets_new)

# Use the weights from the very first GA to get the new returns
new_returns <- na.omit((mu_new %*% ga_1@solution) * 252)

# Use the new returns in the fitness function
# Maximize the return
selection_fitness_function <- function(wt) {
  assets_returns <- (wt * new_returns) * 252
  return(-sum(assets_returns))
}

GA <- ga(type = "binary", fitness = selection_fitness_function,
         monitor = FALSE, popSize = 10, maxiter = 20,
         pmutation = 0.5, pcrossover = 0.7, names = new_tickers,
         nBits = 50, keepBest = TRUE)

# Get the best 10 assets with the highest returns and print them
df <- as.data.frame(GA@solution, col.names = new_tickers)
print(colnames(df[, df == 1, drop = FALSE][1:10]))
```

```
##  [1] "MSFT" "NVDA" "WFC"  "TGT"  "LUX"  "ANF"  "AEO"  "BLBD" "BP"   "SHEL"
```

We see that the GA has chosen the assets of the companies that represent the highest returns. It is important to keep in mind that we will have to deal with a lot of volatility based on the analysis that we have conducted previously. In summary, the GA displays good performance when optimizing the weights and selecting the assets, but we need to have a wider selection of companies from various sectors to truly eliminate unsystematic risk.

## References

Lin, C.M. and Gen, M., 2007. An effective decision-based genetic algorithm approach to multi-objective portfolio optimization problem. *Applied mathematical sciences*, 1(5), pp.201-210.