

# Asset selection with Local Search

Enrico Schumann  
es@enricoschumann.net

## 1 Introduction

We provide a code example for a simple asset selection problem. The purpose of this vignette is to provide the code in a convenient way; for more details, please see Gilli et al. [2011]. We start by attaching the package.

```
> require("NMOF")
> set.seed(112233)
```

## 2 The problem

We wish to select between  $K_{\inf}$  and  $K_{\sup}$  out of  $n_A$  assets such that an equally-weighted portfolio of these assets has the lowest-possible variance. The formal model is:

$$\min_w w' \Sigma w \tag{1}$$

subject to the constraints

$$\begin{aligned} w_j &= 1/K \quad \text{for } j \in J, \\ K_{\inf} &\leq K \leq K_{\sup}. \end{aligned}$$

The weights are stored in the vector  $w$ ; the symbol  $J$  stands for the set of assets in the portfolio; and  $K = \#J$  is the cardinality of this set, ie, the number of assets in the portfolio.

## 3 Setting up the algorithm

We start by attaching the package and creating random data. We simulate 500 assets: each gets a random volatility between 20% and 40%, and all pairwise correlations are set to 0.6.

```
> na <- 500L
> C <- array(0.6, dim = c(na, na)); diag(C) <- 1
> minVol <- 0.20; maxVol <- 0.40
> Vols <- (maxVol - minVol) * runif(na) + minVol
> Sigma <- outer(Vols, Vols) * C
```

The objective function.

```
> OF <- function(x, data) {
  xx <- as.logical(x)
  w <- x/sum(x)
  res <- crossprod(w[xx], data$Sigma[xx, xx])
  res <- tcrossprod(w[xx], res)
  res
}
```

...or even simpler:

```
> OF2 <- function(x, data) {
  xx <- as.logical(x); w <- 1/sum(x)
  res <- sum(w * w * data$Sigma[xx, xx])
  res
}
```

The neighbourhood function.

```
> neighbour <- function(xc, data) {  
  xn <- xc  
  p <- sample.int(data$na, data$nn, replace = FALSE)  
  xn[p] <- abs(xn[p] - 1L)  
  ## reject infeasible solution  
  if((sum(xn) > data$Ksup) || (sum(xn) < data$Kinf))  
    xc else xn  
}
```

We collect all necessary information in the list data: the variance–covariance matrix  $\Sigma$ , the cardinality limits  $K_{\text{inf}}$  and  $K_{\text{sup}}$ , the total number of assets  $na$  (ie, the cardinality of the asset universe), and the parameter  $nn$ . This parameter controls the neighbourhood: it gives the number of assets that are to be changed when a new solution is computed.

```
> data <- list(Sigma = Sigma,  
              Kinf = 30L,  
              Ksup = 60L,  
              na = na,  
              nn = 1L)
```

## 4 Solving the model

As an initial solution we use a random portfolio.

```
> card0 <- sample(data$Kinf:data$Ksup, 1L, replace = FALSE)  
> assets <- sample.int(na, card0, replace = FALSE)  
> x0 <- numeric(na)  
> x0[assets] <- 1L
```

With this implementation we need assume that  $\text{data}\$K_{\text{sup}} > \text{data}\$K_{\text{inf}}$ . (If  $\text{data}\$K_{\text{sup}} == \text{data}\$K_{\text{inf}}$ , then `sample` returns a draw  $1:\text{data}\$K_{\text{inf}}$ .)

We collect all settings in the list `algo`.

```
> ## settings  
> algo <- list(x0 = x0,  
              neighbour = neighbour,  
              nS = 5000L,  
              printDetail = FALSE,  
              printBar = FALSE)
```

It remains to run the algorithm.

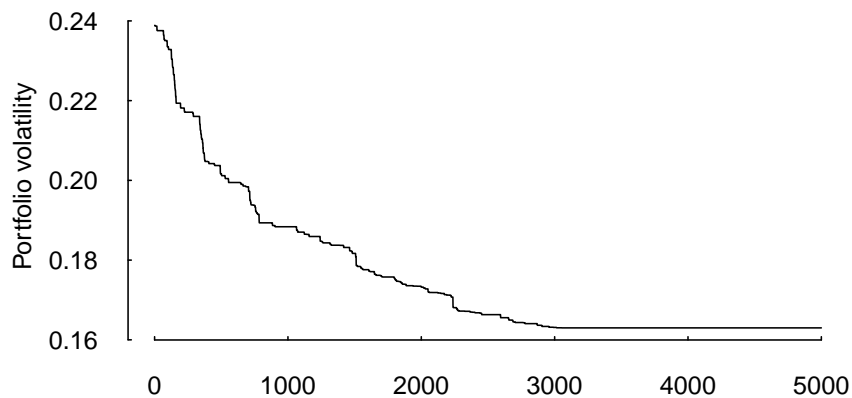
```
> system.time(sol1 <- LSopt(OF, algo, data))
```

user	system	elapsed
0.536	0.004	0.540

```
> sqrt(sol1$OFvalue)
```

[,1]
[1,] 0.163

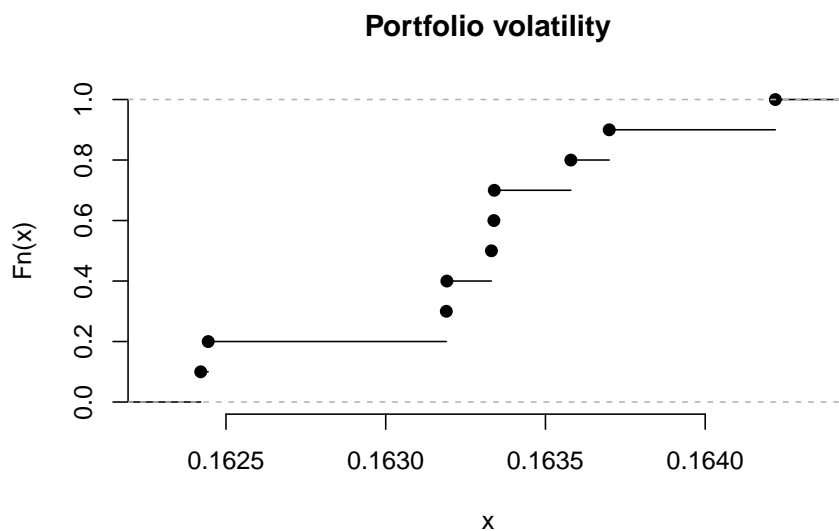
```
> par(ylog = TRUE, bty = "n", las = 1, tck = 0.01)  
> plot(sqrt(sol1$Fmat[,2L]),  
      type = "l", xlab = "", ylab = "Portfolio volatility")
```



(Recall that the simulated data had volatilities between 20 and 40%.)

We can also run the search repeatedly with the same starting value.

```
> nRuns <- 10L
> allRes <- restartOpt(LSopt, n = nRuns, OF, algo = algo, data = data)
> allResOF <- numeric(nRuns)
> for (i in seq_len(nRuns))
+   allResOF[i] <- sqrt(allRes[[i]]$OFvalue)
> par(bty = "n")
> plot(ecdf(allResOF), main = "Portfolio volatility")
```



(We run LSopt only ten times to keep the build time for the vignette acceptable. To get more meaningful results you should increase nRuns.)

## References

Manfred Gilli, Dietmar Maringer, and Enrico Schumann. *Numerical Methods and Optimization in Finance*. Elsevier, 2011.