Asset selection with Local Search

Enrico Schumann
es@enricoschumann.net

1 Introduction

We provide a code example for a simple asset-selection model; please see Gilli et al. [2011, ch. 12 and 13] for more details. The code example here differs slightly from the book's presentation. To see the latter, check the script `exampleLS.R` (after attaching the package):

```
> showExample("exampleLS", chapter = "Portfolio")
```

We start by attaching the package and fixing a seed.

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

2 The model

We wish to select between $K_{\min}$ and $K_{\max}$ 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

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

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 simulate 500 assets: each gets a random volatility between 20% and 40%, and all pairwise correlations are set to 0.6.

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

The objective function.

```
> OF <- function(x, Data) {
      w <- x/sum(x)
      res <-  crossprod(w[x], Data$Sigma[x, x])
      tcrossprod(w[x], res)
  }
```

... or even simpler:

```
> OF2 <- function(x, Data) {
      w <- 1/sum(x)
      sum(w * w * Data$Sigma[x, x])
  }
```

The neighbourhood function.

```
> neighbour <- function(xc, Data) {
      xn <- xc
      p <- sample.int(Data$na, Data$nc, replace = FALSE)
      xn[p] <- !xn[p]

      ## reject infeasible solution
      if (sum(xn) > Data$Kmax || sum(xn) < Data$Kmin)
          xc
      else
          xn
  }
```

We collect all necessary information in the list `Data`: the variance–corvariance matrix `Sigma`, the cardinality limits `Kmin` and `Kmax`, the total number of assets `na` (ie, the cardinality of the asset universe), and the parameter `nc`. 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,
                Kmin = 30L,
                Kmax = 60L,
                na = na,
                nc = 1L)
```

4 Solving the model

As an initial solution we use a random portfolio.

```
> card0 <- sample(Data$Kmin:Data$Kmax, 1L, replace = FALSE)
> assets <- sample.int(na, card0, replace = FALSE)
> x0 <- logical(na)
> x0[assets] <- TRUE
```

With this implementation we assume that `Data$Kmax > Data$Kmin`. (If `Data$Kmax` equals `Data$Kmin`, then `sample` returns a draw from `1:Data$Kmin`.)

We collect all settings for the algorithm in a list `algo`.

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

It remains to run the algorithm.

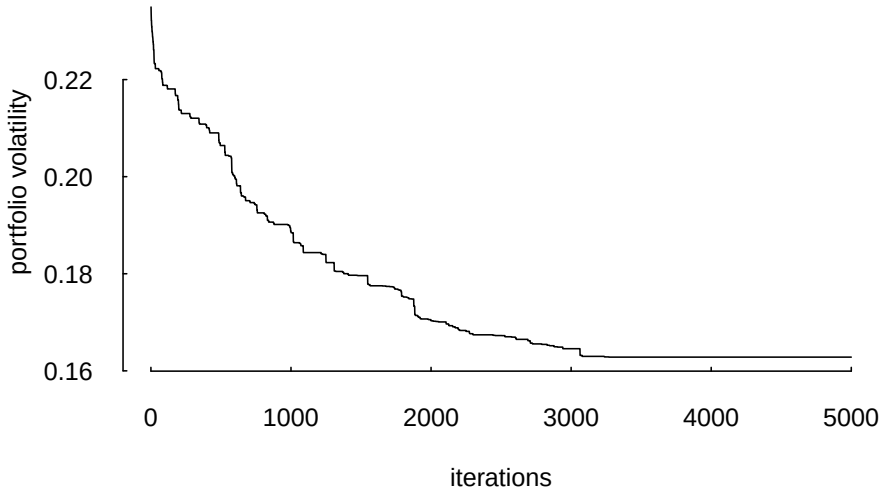
```
> system.time(sol1 <- LSopt(OF, algo, Data))
```

```
   user  system elapsed
  0.148   0.004   0.153
```

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

```
        [,1]
[1,] 0.16282
```

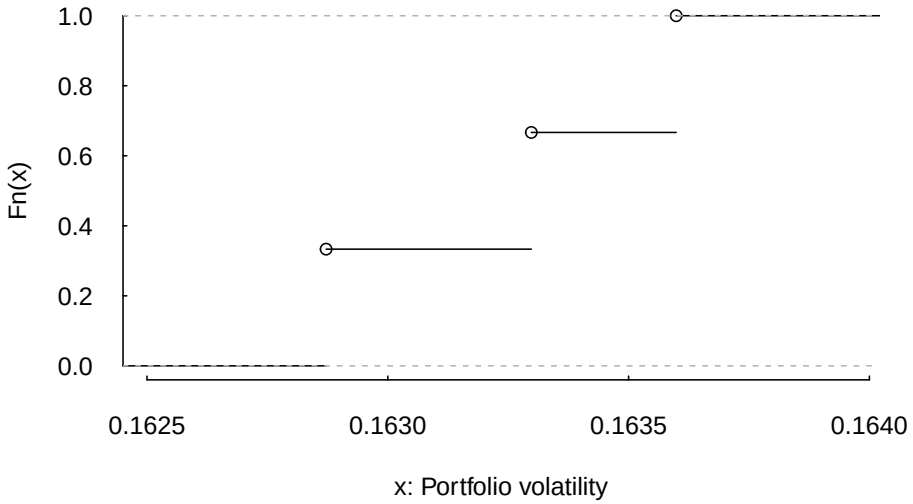
```
> par(ylog = TRUE, bty = "n", las = 1, tck = 0.01, mar = c(4,4,1,1))
> plot(sqrt(sol1$Fmat[ ,2L]), main = "",
       type = "l", ylab = "portfolio volatility", xlab = "iterations")
```



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

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

```
> nRuns <- 3L
> 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", las = 1, tck = 0.01, mar = c(4,4,1,1))
> plot(ecdf(allResOF), xlab = "x: Portfolio volatility", pch = 21,
       main = "")
```



(We run `LSopt` only 3 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/Academic Press, 2011. URL `http://enricoschumann.net/NMOF`.