Test Case: Minimising the ratio of two conditional moments

Enrico Schumann
es@enricoschumann.net

2012-12-27

(report rebuilt 2013-08-01)

Contents

1 Test case: Minimising the ratio of two conditional moments

We look at a benchmark from the Rsolnp package (Ghalanos and Theussl, 2011).

We first attach the packages and we change the display options. (The results are small numbers which would, with a lower `digits` setting, all be displayed as 1.)

```
> options(digits = 7)
> require("Rsolnp")
> require("NMOF")
```

We run a benchmark case.

```
> bmResults <- benchmark(id = "RachevRatio")
> strwrap(attr(bmResults, "description"), 0.75 * getOption("width"))
```

```
[1] "The Rachev Ratio problem minimizes a"
[2] "portfolio's Rachev ratio. It has one linear"
[3] "equality constraint and variable bounds. See"
[4] "Rachev (2000) for details."
```

The aim is to see whether Threshold Accepting (TA) can find the same solution. Thus, we define the objective function and the neighbourhood function for TA (see **?** for details).

```
> OF <- function(sol, Data) {
     alpha <- Data$alpha
     Rw <- sol$Rw
     VaRp <- quantile( Rw, probs = alpha, type = 1)
     VaRm <- quantile(-Rw, probs = alpha, type = 1)
     CVaRp <- VaRp - 0.5 * mean(((VaRp - Rw) + abs(VaRp - Rw))) / alpha
     CVaRm <- VaRm - 0.5 * mean(((VaRm + Rw) + abs(VaRm + Rw))) / alpha
     -CVaRm/CVaRp
 }
> neighbourU <- function(sol, Data){
     wn <- sol$w
     toSell <- wn > Data$winf
     toBuy  <- wn < Data$wsup
     i <- Data$resample(which(toSell), size = 1L)
     j <- Data$resample(which(toBuy),  size = 1L)
     eps <- Data$eps * runif(1)
     eps <- min(wn[i] - Data$winf, Data$wsup - wn[j], eps)
     wn[i] <- wn[i] - eps
     wn[j] <- wn[j] + eps
     Rw <- sol$Rw + Data$R[,c(i,j)] %*% c(-eps,eps)
     list(w = wn, Rw = Rw)
 }
```

We prepare data and see that our objective function indeed results in the same values as the benchmark solution.

```
> rets <- as.matrix(dji30ret)
> resample <- function(x, ...)
     x[sample.int(length(x), ...)]
> Data <- list(R = rets,
               na = dim(rets)[2L],
               ns = dim(rets)[1L],
               eps = 5/100,
               wsup = 0.1,
               winf = 0,
               alpha = 0.05,
               resample = resample)
> ## check the objective function:
> ## (1) get benchmark weights
> solnp <- bmResults[grep("par", attr(bmResults, "row.names")), "solnp"]
> snopt <- bmResults[grep("par", attr(bmResults, "row.names")), "snopt"]
> ## (2) setup solutions for TA
> sol.solnp <- list(w = solnp, Rw = rets %*% solnp)
> sol.snopt <- list(w = snopt, Rw = rets %*% snopt)
> ## (3) compare objective function values benchmark/OF
> bmResults[grep("func", attr(bmResults, "row.names")), ]
```

```
            solnp    snopt
funcValue -1.0025 -1.00216
```

```
> data.frame(solnp = OF(sol.solnp, Data),
             snopt = OF(sol.snopt, Data))
```

```
       solnp     snopt
5% -1.002496 -1.002161
```

Run `TAopt` with an random initial solution and 2000 iterations (which are not many).

```
> fun.maker <- function(Data)
     function() {
         ia <- sample.int(Data$na, sample(10:Data$na, 1))
         w0 <- numeric(Data$na)
         w0[ia] <- 1/length(ia)
         list(w = w0, Rw = Data$R %*% w0)
     }
> f0 <- fun.maker(Data)
> algo <- list(x0 = f0, neighbour = neighbourU,
               nS = 1L, nT = 2000L, nD = 1000L,
               q = 0.99,
               printBar = FALSE, printDetail = 1000L)
> res <- TAopt(OF,algo,Data)
```

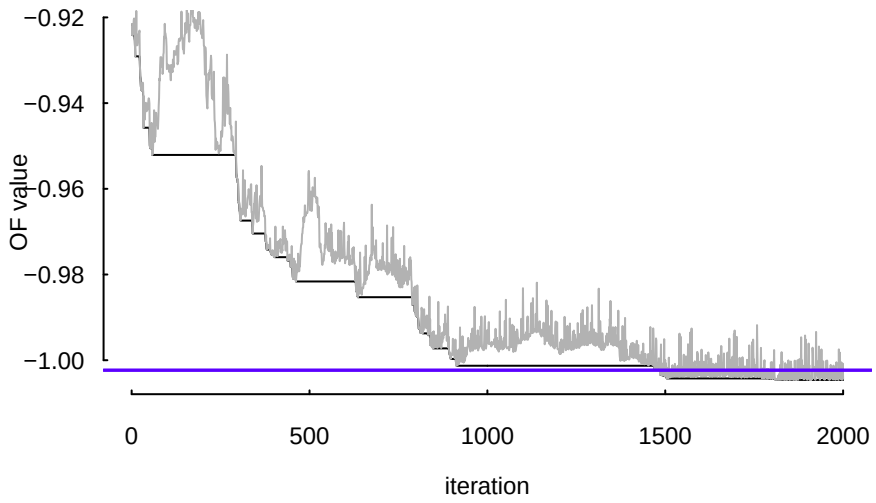
```
Threshold Accepting.

Computing thresholds ... OK.
Estimated remaining running time: 3.29 secs.


Running Threshold Accepting...
Initial solution:  -0.9185253
Best solution (iteration 1000/2000): -1.001284
Best solution (iteration 2000/2000): -1.004608
Finished.
Best solution overall: -1.004608
```

```
> res$OFvalue
```

```
       5%
-1.004608
```

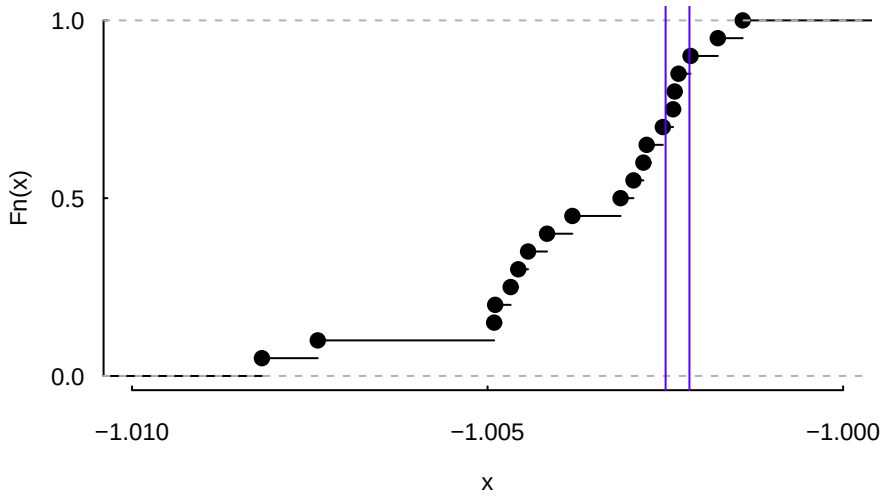
A path of TA: the current solution in grey, the best-so-far solution in black. The blue horizontal lines represent the benchmark solutions `solnp` and `snopt`.

```
> par(bty = "n", mar = c(4, 4, 1, 1),
     tck = 0.01, las = 1, ps = 9, mgp = c(1.8, 0.5, 0),
     lab = c(3,3,7))
> plot(cummin(res$Fmat[ ,2L]), type = "l",
      ylab = "OF value", xlab = "iteration")
> lines(res$Fmat[ ,1L], type = 'l', col = grey(.7))
> abline(h=c(OF(sol.solnp, Data), OF(sol.snopt, Data)), col = "blue")
```



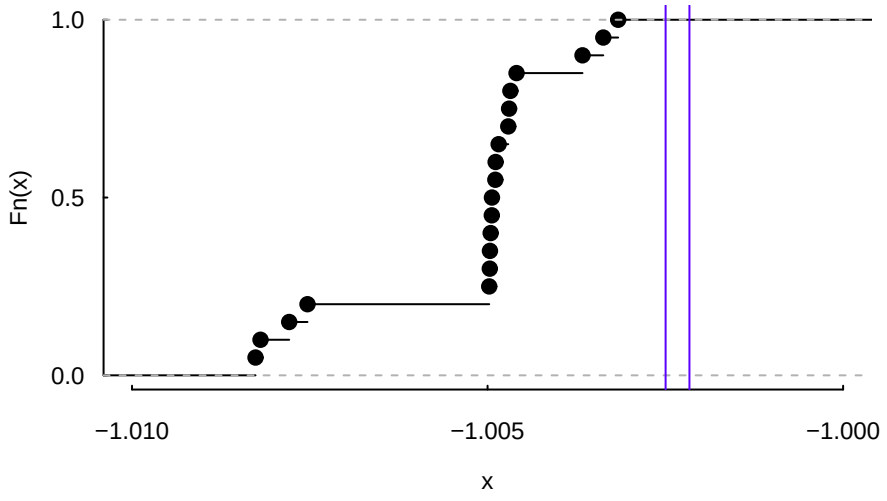
We do 20 restarts. Since these runs do not depend on each other, we can easily distribute them. The following plot shows the empirical distribution function of the resulting objective function values. We add, again in blue, the benchmark solutions.

```
> algo$printDetail <- algo$printBar <- FALSE
> algo$nT <- 2000L
> restarts <- restartOpt(TAopt, n = 20, OF = OF, algo = algo, Data = Data,
                         method = "snow", cl = 4)
> par(bty = "n", mar = c(4, 4, 1, 1),
     tck = 0.01, las = 1, ps = 9, mgp = c(1.8, 0.5, 0),
     lab = c(3,3,7))
> plot(ecdf(sapply(restarts, `[[`, "OFvalue")), main = "",
      xlim = c(-1.01,-1))
> abline(v=c(OF(sol.solnp, Data), OF(sol.snopt, Data)), col = "blue")
```



And another 20 restarts, this time with 10000 iterations.

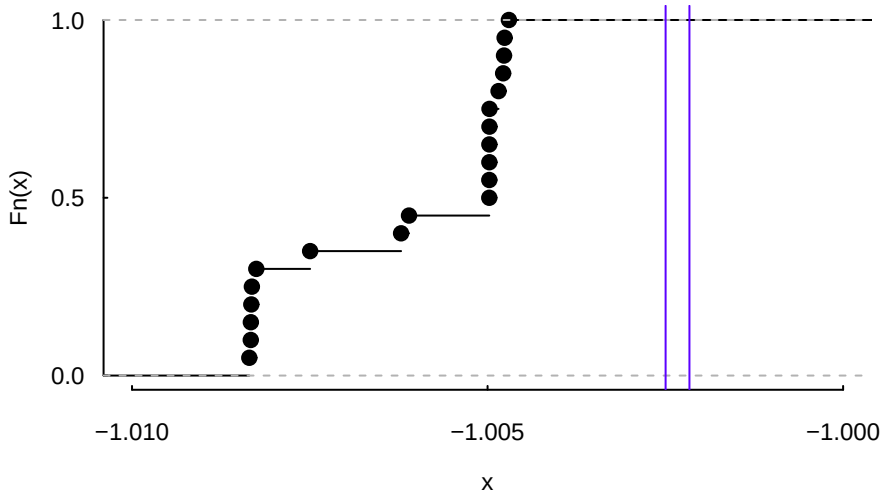
```
> algo$printDetail <- algo$printBar <- FALSE
> algo$nT <- 10000L
> restarts <- restartOpt(TAopt, n = 20, OF=OF, algo=algo, Data = Data,
                         method = "snow", cl = 4)
> par(bty = "n", mar = c(4, 4, 1, 1),
     tck = 0.01, las = 1, ps = 9, mgp = c(1.8, 0.5, 0),
     lab = c(3,3,7))
> plot(ecdf(sapply(restarts, `[[`, "OFvalue")), main = "",
      xlim = c(-1.01,-1))
> abline(v=c(OF(sol.solnp, Data), OF(sol.snopt, Data)), col = "blue")
```



Finally, we use 50000 iterations. We might increase the number of iterations further, but a

multiple-restarts strategy would seem more efficient (in particular because the restarts can be distributed; see Gilli and Schumann, 2010, for a discussion).

```
> algo$printDetail <- algo$printBar <- FALSE
> algo$nT <- 50000L
> restarts <- restartOpt(TAopt, n = 20, OF=OF, algo=algo, Data = Data,
                         method = "snow", cl = 4)
> par(bty = "n", mar = c(4, 4, 1, 1),
     tck = 0.01, las = 1, ps = 9, mgp = c(1.8, 0.5, 0),
     lab = c(3,3,7))
> plot(ecdf(sapply(restarts, `[[`, "OFvalue")), main = "",
      xlim = c(-1.01,-1))
> abline(v=c(OF(sol.solnp, Data), OF(sol.snopt, Data)), col = "blue")
```



A Resources

You can download all the book's code examples from the book's home page,

`http://nmof.net`

The latest version of the NMOF package is available from

`http://enricoschumann.net/R/packages/NMOF/index.htm`

but note that this is the development version. More stable versions are available from CRAN.

New versions of the package and other news are announced through the `NMOF-news` mailing list; to browse the archives or to subscribe, go to

`https://lists.r-forge.r-project.org/cgi-bin/mailman/listinfo/nmof-news`

B Package version

```
> toLatex(sessionInfo())
```

- R version 3.0.1 (2013-05-16), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_GB.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=en_GB.UTF-8, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_GB.UTF-8, LC_PAPER=C, LC_NAME=C, LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=en_US.UTF-8, LC_IDENTIFICATION=C
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: NMOF 0.28-1, Rsolnp 1.12, truncnorm 1.0-6
- Loaded via a namespace (and not attached): snow 0.3-12, tools 3.0.1

References

Alexios Ghalanos and Stefan Theussl. *Rsolnp: General Non-linear Optimization Using Augmented Lagrange Multiplier Method*, 2011. R package version 1.12.

Manfred Gilli and Enrico Schumann. Distributed optimisation of a portfolio's Omega. *Parallel Computing*, 36(7):381–389, 2010.

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