Functions for portfolio selection

Enrico Schumann
es@enricoschumann.net
https://CRAN.R-project.org/package=NMOF

A main topic of Gilli et al. [2019] is non-standard portfolio-selection models; see Chapters 12–14. Nevertheless, the `NMOF` package also offers several functions that help with standard portfolio models, i.e. models that can be solved with traditional optimisation techniques such as quadratic programming.

1 Minimum-variance portfolios

The function `minvar` computes the minimum-variance portfolio for a given variance–covariance matrix, `minvar` subject to holding-size constraints. As example data, the variable `var` contains a small variance–covariance matrix, computed from daily returns of five German stocks. The data are taken from http://enricoschumann. net/data/gilli_accuracy.html; the code to build the matrix is in the source file of this vignette.

```
> var
           CBK.DE     VOW.DE    CON.DE    LIN.DE   MUV2.DE
CBK.DE    0.000988 -1.80e-05 3.69e-04 2.08e-04 2.63e-04
VOW.DE   -0.000018  1.72e-03 8.57e-05 2.15e-05 2.84e-05
CON.DE    0.000369  8.57e-05 7.59e-04 1.94e-04 1.89e-04
LIN.DE    0.000208  2.15e-05 1.94e-04 2.66e-04 1.33e-04
MUV2.DE   0.000263  2.84e-05 1.89e-04 1.33e-04 2.59e-04
```

An example call, with minimum and maximum holding sizes specified.

```
> minvar(var, wmin = 0, wmax = 0.5)
[1] 6.94e-18 9.25e-02 4.69e-05 4.45e-01 4.62e-01
attr(,"variance")
[1] 0.000182
```

The function returns the portfolio weights with an attribute `variance` that provides the variance of this portfolio. The holding size constraints can also be specified as vectors, with different values for different assets.

```
> minvar(var,
         wmin = c(0.1, 0, 0, 0, 0), ## enforce at least 10% weight in CBK.DE
         wmax = 0.5)
```

Use `Inf` to switch off weight constraints.

```
> minvar(var, wmin = -Inf, wmax = Inf)    ## no bounds
> minvar(var, wmin = -Inf, wmax = 0.45)   ## no lower bounds
> minvar(var, wmin =  0.1, wmax = Inf)    ## no upper bounds
```

The function also supports group constraints:

```
> ## group 1 consists of asset 1 only,   and must have weight [0.25,0.30]
> ## group 2 consists of assets 4 and 5, and must have weight [0.10,0.20]
> minvar(var, wmin = 0, wmax = 0.40,
         groups = list(1, 4:5),
         groups.wmin = c(0.25, 0.1),
         groups.wmax = c(0.30, 0.2))
[1] 0.250 0.217 0.333 0.149 0.051
attr(,"variance")
[1] 0.000357
```

Alternatively, group constraints can be specified through group names instead of positions.

```
> ## group A consists of asset 1 only,    and must have weight [0.25,0.30]
> ## group B consists of assets 4 and 5, and must have weight [0.10,0.20]
> minvar(var, wmin = 0, wmax = 0.40,
        groups = c("A", "none", "none", "B", "B"),
        groups.wmin = c(A = 0.25, B = 0.1),
        groups.wmax = c(A = 0.30, B = 0.2))
```

```
[1] 0.250 0.217 0.333 0.149 0.051
attr(,"variance")
[1] 0.000357
```

2 Mean–variance efficient portfolios and frontiers

The function `mvPortfolio` computes a mean–variance-efficient portfolio for a given variance–covariance   `mvPortfolio`
matrix and mean-return assumption, subject to holding-size constraints. We make up some data for four
assets, with a constant correlation of 0.5.

```
> vols <- c(0.10, 0.15, 0.20, 0.22)  ## expected vols
> m    <- c(0.06, 0.12, 0.09, 0.07)  ## expected mean returns
> const_cor <- function(rho, na) {
      C <- array(rho, dim = c(na, na))
      diag(C) <- 1
      C
  }
> var <- diag(vols) %*% const_cor(0.5, length(vols)) %*% diag(vols)
```

One way to compute a mean–variance-efficient portfolio is by requiring a minimum return.

```
> mvPortfolio(m, var, min.return = 0.08, wmax = 1)
```

```
[1] 0.667 0.333 0.000 0.000
```

```
> mvPortfolio(m, var, min.return = 0.10, wmax = 1)
```

```
[1] 3.33e-01 6.67e-01 0.00e+00 3.10e-18
```

```
> mvPortfolio(m, var, min.return = 0.12, wmax = 1)
```

```
[1] -1.11e-16  1.00e+00 -5.55e-17  2.64e-17
```

Alternatively, we may specify a trade-off between return and variance and minimise

$$-\lambda \mathrm{m}'w + \frac{1}{2}(1-\lambda)w'\mathrm{var}\,w,$$

in which w are the weights. If λ is a vector of length 2, then the function minimises

$$-\lambda_1 \mathrm{m}'w + \frac{1}{2}\lambda_2 w'\mathrm{var}\,w.$$

The function `mvFrontier` traces out a whole frontier of mean–variance efficient portfolios. (But see the   `mvFrontier`
discussion on frontiers in Chapter 14 of Gilli et al., 2019.)

```
> if (requireNamespace("quadprog")) {
      wmin <- 0
      wmax <- 1
      p1 <- mvFrontier(m, var, wmin = wmin, wmax = wmax, n = 50)

      ## with a 'risk-free' asset rf
```
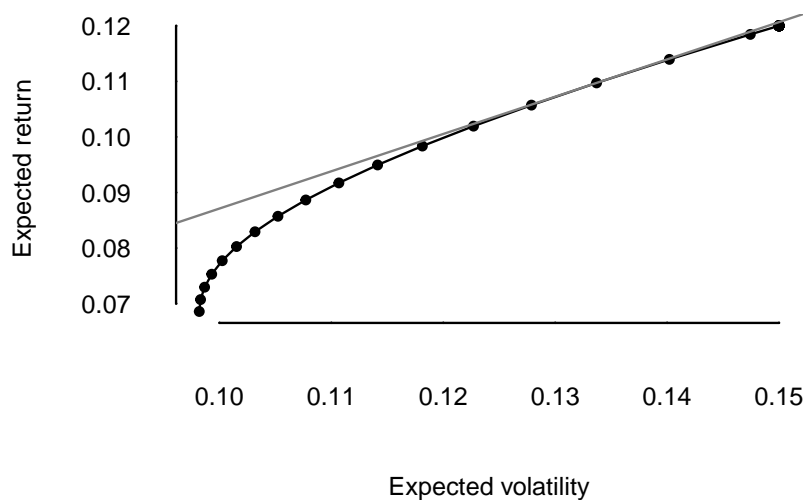
```
        rf <- 0.02
        p2 <- mvFrontier(m, var, wmin = wmin, wmax = wmax, n = 50, rf = rf)

        par(las = 1, bty = "n", tck = 0.001, ps = 8)
        plot(p1$volatility, p1$return, pch = 19, cex = 0.5, type = "o",
             xlab = "Expected volatility",
             ylab = "Expected return")
        lines(p2$volatility, p2$return, col = grey(0.5))
        abline(v = 0, h = rf)
    } else
        plot(1)
```


3   Return-based tracking portfolios

Function `trackingPortfolio` computes a portfolio that is close to another portfolio in the mean-square/   `trackingPortfolio` variance sense. The function to be minimised is determined by argument `objective`: supported are `variance` (the default) or `sum.of.squares`.

```
> ns <- 120
> R <- randomReturns(na = 1 + 10,  ## first asset is the benchmark
                     ns = ns,
                     sd = 0.03,
                     mean = 0.005,
                     rho = 0.7)
> var <- cov(R)
> trackingPortfolio(var, wmax = 0.4)
```
```
 [1] 9.87e-18 5.87e-02 7.99e-03 1.53e-01 0.00e+00 2.39e-01
 [7] 9.89e-02 1.69e-01 2.47e-01 2.72e-02
```

4   Minimum-Expected-Shortfall portfolios

Function `minCVaR` computes a portfolio that minimises conditional Value-at-Risk; its default method is   `minCVaR`

the LP approach described in Rockafellar and Uryasev [2000]. See *Minimising Conditional Value-at-Risk (CVaR)* (`http://enricoschumann.net/notes/minimising-conditional-var.html`) for more details

```
> ns <- 5000  ## number of scenarios
> na <- 20    ## nunber of assets
> R <- randomReturns(na, ns, sd = 0.01, rho = 0.5)
> if (requireNamespace("Rglpk")) { ## example requires "Rglpk" package
      sol <- minCVaR(R, q = 0.1)
  } else
      message("Package ", sQuote("Rglpk"), " not available")
```

5 Minimum Mean–Absolute-Deviation portfolios

Function `minMAD` computes a portfolio that minimises the mean absolute-deviation of portfolio returns, as described in Konno and Yamazaki [1991].

`minMAD`

```
> ns <- 5000  ## number of scenarios
> na <- 5     ## nunber of assets
> R <- randomReturns(na, ns, sd = 0.01, rho = 0.5)
> minMAD(R = R)
```
```
[1] 0.197 0.176 0.185 0.256 0.187
```

References

Manfred Gilli, Dietmar Maringer, and Enrico Schumann. *Numerical Methods and Optimization in Finance*. Elsevier/Academic Press, 2nd edition, 2019. URL `http://enricoschumann.net/NMOF`.

Hiroshi Konno and Hiroaki Yamazaki. Mean-absolute deviation portfolio optimization model and its applications to Tokyo stock market. *Management Science*, 37(5):519–531, 1991. URL `http://www.jstor.org/stable/2632458`.

R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2 (3):21–41, 2000. doi: 10.21314/JOR.2000.038.