Package: RiskPortfolios (via r-universe)

September 16, 2024

Version 2.1.7

Date 2021-05-16

Title Computation of Risk-Based Portfolios

Maintainer David Ardia <`david.ardia.ch@gmail.com`>

Imports MASS, quadprog, nloptr

Description Collection of functions designed to compute risk-based portfolios as described in Ardia et al. (2017) <doi:10.1007/s10479-017-2474-7> and Ardia et al. (2017) <doi:10.21105/joss.00171>.

License GPL (>= 2)

BugReports `https://github.com/ArdiaD/RiskPortfolios/issues`

URL `https://github.com/ArdiaD/RiskPortfolios`

Suggests testthat

NeedsCompilation no

Author David Ardia [aut, cre, cph] (<`https://orcid.org/0000-0003-2823-782X`>), Kris Boudt [aut], Jean-Philippe Gagnon-Fleury [aut]

RoxygenNote 7.1.1

Repository https://ardiad.r-universe.dev

RemoteUrl https://github.com/ardiad/riskportfolios

RemoteRef HEAD

RemoteSha 9a8b3bda32fbc0f758e3eb454399754f42755389

Contents

`covEstimation` *Covariance matrix estimation*

Description

Function which performs various estimations of covariance matrices.

Usage

```
covEstimation(rets, control = list())
```

Arguments

`rets`	a matrix $(T \times N)$ of returns.
`control`	control parameters (see *Details*).

Details

The argument `control` is a list that can supply any of the following components:

- `type` method used to compute the covariance matrix, among `'naive'`, `'ewma'`, `'lw'`, `'factor'`,`'const'`, `'cor'`, `'oneparm'`, `'diag'` and `'large'` where:

  `'naive'` is used to compute the naive (standard) covariance matrix.

  `'ewma'` is used to compute the exponential weighting moving average covariance matrix. The following formula is used to compute the ewma covariance matrix:

  $$\Sigma_t := \lambda \Sigma_{t-1} + (1 - \lambda) r_{t-1} r_{t-1}$$

  where r_t is the $(N \times 1)$ vector of returns at time t. Note that the data must be sorted from the oldest to the latest. See RiskMetrics (1996)

  `'factor'` is used to compute the covariance matrix estimation using a K-factor approach. See Harman (1976).

  `'lw'` is a weighted average of the sample covariance matrix and a 'prior' or 'shrinkage target'. The prior is given by a one-factor model and the factor is equal to the cross-sectional average of all the random variables. See Ledoit and Wolf (2003).

  `'const'` is a weighted average of the sample covariance matrix and a 'prior' or 'shrinkage target'. The prior is given by constant correlation matrix. See Ledoit and Wolf (2002).

  `'cor'` is a weighted average of the sample covariance matrix and a 'prior' or 'shrinkage target'. The prior is given by the constant correlation covariance matrix given by Ledoit and Wolf (2003).

  `'oneparm'` is a weighted average of the sample covariance matrix and a 'prior' or 'shrinkage target'. The prior is given by the one-parameter matrix. All variances are the same and all covariances are zero. See Ledoit and Wolf (2004).

  `'diag'` is a weighted average of the sample covariance matrix and a 'prior' or 'shrinkage target'. The prior is given by a diagonal matrix. See Ledoit and Wolf (2002).

`'large'` This estimator is a weighted average of the sample covariance matrix and a 'prior' or 'shrinkage target'. Here, the prior is given by a one-factor model. The factor is equal to the cross-sectional average of all the random variables. The weight, or 'shrinkage intensity' is chosen to minimize quadratic loss measured by the Frobenius norm. The estimator is valid as the number of variables and/or the number of observations go to infinity, but Monte-Carlo simulations show that it works well for values as low as 10. The main advantage is that this estimator is guaranteed to be invertible and well-conditioned even if variables outnumber observations. See Ledoit and Wolf (2004).

`'bs'` is the Bayes-Stein estimator for the covariance matrix given by Jorion (1986).

Default: `type = 'naive'`.

- `lambda` decay parameter. Default: `lambda = 0.94`.
- `K` number of factors to use when the K-factor approach is chosen to estimate the covariance matrix. Default: `K = 1`.

Value

A $(N \times N)$ covariance matrix.

Note

Part of the code is adapted from the Matlab code by Ledoit and Wolf (2014).

Author(s)

David Ardia, Kris Boudt and Jean-Philippe Gagnon Fleury.

References

Jorion, P. (1986). Bayes-Stein estimation for portfolio analysis. *Journal of Financial and Quantitative Analysis* **21**(3), pp.279-292.

Harman, H.H. (1976) *Modern Factor Analysis*. 3rd Ed. Chicago: University of Chicago Press.

Ledoit, O., Wolf, M. (2002). Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance* **10**(5), pp.603-621.

Ledoit, O., Wolf, M. (2003). Honey, I Shrunk the Sample Covariance Matrix. *Journal of Portfolio Management* **30**(4), pp.110-119.

Ledoit, O., Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis* **88**(2), pp.365-411.

RiskMetrics (1996) *RiskMetrics Technical Document*. J. P. Morgan/Reuters.

Examples

```
# Load returns of assets or portfolios
data("Industry_10")
rets = Industry_10

# Naive covariance estimation
covEstimation(rets)
```

```
# Ewma estimation of the covariance with default lambda = 0.94
covEstimation(rets, control = list(type = 'ewma'))

# Ewma estimation of the covariance with default lambda = 0.90
covEstimation(rets, control = list(type = 'ewma', lambda = 0.9))

# Factor estimation of the covariance with dafault K = 1
covEstimation(rets, control = list(type = 'factor'))

# Factor estimation of the covariance with K = 3
covEstimation(rets, control = list(type = 'factor', K = 3))

# Ledot-Wolf's estimation of the covariance
covEstimation(rets, control = list(type = 'lw'))

# Shrinkage of the covariance matrix using constant correlation matrix
covEstimation(rets, control = list(type = 'const'))

# Shrinkage of the covariance matrix towards constant correlation matrix by
# Ledoit-Wolf.
covEstimation(rets, control = list(type = 'cor'))

# Shrinkage of the covariance matrix towards one-parameter matrix
covEstimation(rets, control = list(type = 'oneparm'))

# Shrinkage of the covaraince matrix towards diagonal matrix
covEstimation(rets, control = list(type = 'diag'))

# Shrinkage of the covariance matrix for large data set
covEstimation(rets, control = list(type = 'large'))
```

Industry_10 *Industry Portfolios*

Description

A matrix containing daily returns of 10 industry portfolios for the year 2014.

Usage

```
data("Industry_10")
```

Format

A matrix (of size 252 x 10) containing daily returns of 10 industry portfolios.

Note

Data are available from Kenneth French’s website.

Source

http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html.

`meanEstimation` *Estimation of mean returns*

Description

Function which is used to compute the estimation of the mean returns.

Usage

```
meanEstimation(rets, control = list())
```

Arguments

`rets`	a $(T \times N)$ matrix of past returns.
`control`	control parameters (see *Details*).

Details

The argument `control` is a list that can supply any of the following components:

- `type` method used to estimate the mean returns, among `'naive'`, `'ewma'`, `'bs'` and `'mart'` where:
  `'naive'` is used to compute the arithmetic mean of the returns.
  `'ewma'` is used to compute the exponential weighted moving average mean of the returns. The data must be sorted from the oldest to the latest. See RiskMetrics (1996).
  `'bs'` is used to compute the Bayes-Stein estimation. See Jorion (1986).
  `'mart'` is used to compute the Martinelli (2008) implied returns.
  Default: `type = 'naive'`.
- `lambda` decay parameter. Default: `lambda = 0.94`.

Value

A $(N \times 1)$ vector of expected returns.

Author(s)

David Ardia, Kris Boudt and Jean-Philippe Gagnon Fleury.

References

Jorion, P. (1986). Bayes-Stein estimation for portfolio analysis. *Journal of Finance and Quantitative Analysis* **21**(3), pp.279-292.

Martellini, L. (2008). Towards the design of better equity benchmarks. *Journal of Portfolio Management* **34**(4), Summer,pp.34-41.

RiskMetrics (1996) *RiskMetrics Technical Document*. J. P. Morgan/Reuters.

Examples

```
# Load returns of assets or portfolios
data("Industry_10")
rets = Industry_10

# Naive estimation of the mean
meanEstimation(rets)

# Naive estimation of the mean
meanEstimation(rets, control = list(type = 'naive'))

# Ewma estimation of the mean with default lambda = 0.94
meanEstimation(rets, control = list(type = 'ewma'))

# Ewma estimation of the mean with lambda = 0.9
meanEstimation(rets, control = list(type = 'ewma', lambda = 0.9))

# Martinelli's estimation of the mean
meanEstimation(rets, control = list(type = 'mart'))

# Bayes-Stein's estimation of the mean
meanEstimation(rets, control = list(type = 'bs'))
```

`optimalPortfolio` *Optimal portfolio*

Description

Function wich computes the optimal portfolio's weights.

Usage

```
optimalPortfolio(Sigma, mu = NULL, semiDev = NULL, control = list())
```

Arguments

`Sigma`	a $(N \times N)$ covariance matrix.
`mu`	a $(N \times 1)$ vector of expected returns. Default: `mu = NULL`.
`semiDev`	a vector $(N \times 1)$ of semideviations. Default: `semiDev = NULL`.
`control`	control parameters (see *Details*).

Details

The argument `control` is a list that can supply any of the following components:

- `type` method used to compute the optimal portfolio, among `'mv'`, `'minvol'`, `'invvol'`, `'erc'`, `'maxdiv'`, `'riskeff'` and `'maxdec'` where:

  `'mv'` is used to compute the weights of the mean-variance portfolio. The weights are computed following this equation:
  $$w = \frac{1}{\gamma}\Sigma^{-1}\mu$$
  .

  `'minvol'` is used to compute the weights of the minimum variance portfolio.

  `'invvol'` is the inverse volatility portfolio.

  `'erc'` is used to compute the weights of the equal-risk-contribution portfolio. For a portfolio w, the percentage volatility risk contribution of the i-th asset in the portfolio is given by:
  $$\%RC_i = \frac{w_i[\Sigma w]_i}{w'\Sigma w}$$
  . Then we compute the optimal portfolio by solving the following optimization problem:
  $$w = argmin\left\{\sum_{i=1}^{N}(\%RC_i - \frac{1}{N})^2\right\}$$
  .

  `'maxdiv'` is used to compute the weights of the maximum diversification portfolio where:
  $$DR(w) = \frac{w'\sigma}{\sqrt{w'\Sigma w}} \geq 1$$
  is used in the optimization problem.

  `'riskeff'` is used to compute the weights of the risk-efficient portfolio:
  $$w = argmax\left\{\frac{w'J\xi}{\sqrt{w'\Sigma w}}\right\}$$
  where J is a $(N \times 10)$ matrix of zeros whose (i,j)-th element is one if the semi-deviation of stock i belongs to decile j,$\xi = (\xi_1, \ldots, \xi_{10})'$.

  `'maxdec'` is used to compute the weights of the maximum-decorrelation portfolio:
  $$w = argmax\left\{1 - \sqrt{w'\Sigma w}\right\}$$
  where R is the correlation matrix.

  Default: `type = 'mv'`.

  These portfolios are summarized in Ardia and Boudt (2015) and Ardia et al. (2017). Below we list the various references.

- `constraint` constraint used for the optimization, among `'none'`, `'lo'`, `'gross'` and `'user'`, where: `'none'` is used to compute the unconstraint portfolio, `'lo'` is the long-only constraints (non-negative weighted), `'gross'` is the gross exposure constraint, and `'user'` is the set of user constraints (typically lower and upper boundaries. Default: `constraint = 'none'`. Note that the summability constraint is always imposed.

- `LB` lower boundary for the weights. Default: `LB = NULL`.

- `UB` lower boundary for the weights. Default: `UB = NULL`.

- `w0` starting value for the optimizer. Default: `w0 = NULL` takes the equally-weighted portfolio as a starting value. When `LB` and `UB` are provided, it is set to mid-point of the bounds.
- `gross.c` gross exposure constraint. Default: `gross.c = 1.6`.
- `gamma` risk aversion parameter. Default: `gamma = 0.89`.
- `ctr.slsqp` list with control parameters for slsqp function.

Value

A $(N \times 1)$ vector of optimal portfolio weights.

Author(s)

David Ardia, Kris Boudt and Jean-Philippe Gagnon Fleury.

References

Amenc, N., Goltz, F., Martellini, L., Retowsky, P. (2011). Efficient indexation: An alternatice to cap-weightes indices. *Journal of Investment Management* **9**(4), pp.1-23.

Ardia, D., Boudt, K. (2015). Implied expected returns and the choice of a mean-variance efficient portfolio proxy. *Journal of Portfolio Management* **41**(4), pp.66-81. doi:10.3905/jpm.2015.41.4.068

Ardia, D., Bolliger, G., Boudt, K., Gagnon-Fleury, J.-P. (2017). The Impact of covariance misspecification in risk-based portfolios. *Annals of Operations Research* **254**(1-2), pp.1-16. doi:10.1007/s1047901724747

Choueifaty, Y., Coignard, Y. (2008). Toward maximum diversification. *Journal of Portfolio Management* **35**(1), pp.40-51.

Choueifaty, Y., Froidure, T., Reynier, J. (2013). Properties of the most diversified portfolio. *Journal of Investment Strategies* **2**(2), pp.49-70.

Das, S., Markowitz, H., Scheid, J., Statman, M. (2010). Portfolio optimization with mental accounts. *Journal of Financial and Quantitative Analysis* **45**(2), pp.311-334.

DeMiguel, V., Garlappi, L., Uppal, R. (2009). Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy. *Review of Financial Studies* **22**(5), pp.1915-1953.

Fan, J., Zhang, J., Yu, K. (2012). Vast portfolio selection with gross-exposure constraints. *Journal of the American Statistical Association* **107**(498), pp.592-606.

Maillard, S., Roncalli, T., Teiletche, J. (2010). The properties of equally weighted risk contribution portfolios. *Journal of Portfolio Management* **36**(4), pp.60-70.

Martellini, L. (2008). Towards the design of better equity benchmarks. *Journal of Portfolio Management* **34**(4), Summer,pp.34-41.

Examples

```
# Load returns of assets or portfolios
data("Industry_10")
rets = Industry_10

# Mean estimation
mu = meanEstimation(rets)
```

```
# Covariance estimation
Sigma = covEstimation(rets)

# Semi-deviation estimation
semiDev = semidevEstimation(rets)

# Mean-variance portfolio without constraint and gamma = 0.89
optimalPortfolio(mu = mu, Sigma = Sigma)

# Mean-variance portfolio without constraint and gamma = 1
optimalPortfolio(mu = mu, Sigma = Sigma,
  control = list(gamma = 1))

# Mean-variance portfolio without constraint and gamma = 0.89
optimalPortfolio(mu = mu, Sigma = Sigma,
  control = list(type = 'mv'))

# Mean-variance portfolio without constraint and gamma = 0.89
optimalPortfolio(mu = mu, Sigma = Sigma,
  control = list(type = 'mv', constraint = 'none'))

# Mean-variance portfolio with the long-only constraint and gamma = 0.89
optimalPortfolio(mu = mu, Sigma = Sigma,
  control = list(type = 'mv', constraint = 'lo'))

# Mean-variance portfolio with LB and UB constraints
optimalPortfolio(mu = mu, Sigma = Sigma,
  control = list(type = 'mv', constraint = 'user', LB = rep(0.02, 10), UB = rep(0.8, 10)))

# Mean-variance portfolio with the gross constraint,
# gross constraint parameter = 1.6 and gamma = 0.89
optimalPortfolio(mu = mu, Sigma = Sigma,
  control = list(type = 'mv', constraint = 'gross'))

# Mean-variance portfolio with the gross constraint,
# gross constraint parameter = 1.2 and gamma = 0.89
optimalPortfolio(mu = mu, Sigma = Sigma,
  control = list(type = 'mv', constraint = 'gross', gross.c = 1.2))

# Minimum volatility portfolio without constraint
optimalPortfolio(Sigma = Sigma,
  control = list(type = 'minvol'))

# Minimum volatility portfolio without constraint
optimalPortfolio(Sigma = Sigma,
  control = list(type = 'minvol', constraint = 'none'))

# Minimim volatility portfolio with the long-only constraint
optimalPortfolio(Sigma = Sigma,
  control = list(type = 'minvol', constraint = 'lo'))

# Minimim volatility portfolio with LB and UB constraints
```

```
optimalPortfolio(Sigma = Sigma,
 control = list(type = 'minvol', constraint = 'user', LB = rep(0.02, 10), UB = rep(0.8, 10)))

# Minimum volatility portfolio with the gross constraint
# and the gross constraint parameter = 1.6
optimalPortfolio(Sigma = Sigma,
  control = list(type = 'minvol', constraint = 'gross'))

# Minimum volatility portfolio with the gross constraint
# and the gross parameter = 1.2
optimalPortfolio(Sigma = Sigma,
  control = list(type = 'minvol', constraint = 'gross', gross.c = 1.2))

# Inverse volatility portfolio
optimalPortfolio(Sigma = Sigma,
  control = list(type = 'invvol'))

# Equal-risk-contribution portfolio with the long-only constraint
optimalPortfolio(Sigma = Sigma,
  control = list(type = 'erc', constraint = 'lo'))

# Equal-risk-contribution portfolio with LB and UB constraints
optimalPortfolio(Sigma = Sigma,
 control = list(type = 'erc', constraint = 'user', LB = rep(0.02, 10), UB = rep(0.8, 10)))

# Maximum diversification portfolio without constraint
optimalPortfolio(Sigma = Sigma,
  control = list(type = 'maxdiv'))

# Maximum diversification portoflio with the long-only constraint
optimalPortfolio(Sigma = Sigma,
  control = list(type = 'maxdiv', constraint = 'lo'))

# Maximum diversification portoflio with LB and UB constraints
optimalPortfolio(Sigma = Sigma,
 control = list(type = 'maxdiv', constraint = 'user', LB = rep(0.02, 10), UB = rep(0.8, 10)))

# Risk-efficient portfolio without constraint
optimalPortfolio(Sigma = Sigma, semiDev = semiDev,
  control = list(type = 'riskeff'))

# Risk-efficient portfolio with the long-only constraint
optimalPortfolio(Sigma = Sigma, semiDev = semiDev,
  control = list(type = 'riskeff', constraint = 'lo'))

# Risk-efficient portfolio with LB and UB constraints
optimalPortfolio(Sigma = Sigma, semiDev = semiDev,
 control = list(type = 'riskeff', constraint = 'user', LB = rep(0.02, 10), UB = rep(0.8, 10)))

# Maximum decorrelation portfolio without constraint
optimalPortfolio(Sigma = Sigma,
  control = list(type = 'maxdec'))
```

```
# Maximum decorrelation portoflio with the long-only constraint
optimalPortfolio(Sigma = Sigma,
  control = list(type = 'maxdec', constraint = 'lo'))

# Maximum decorrelation portoflio with LB and UB constraints
optimalPortfolio(Sigma = Sigma,
 control = list(type = 'maxdec', constraint = 'user', LB = rep(0.02, 10), UB = rep(0.8, 10)))
```

RiskPortfolios — *RiskPortfolios: Computation of risk-based portfolios in R*

Description

`RiskPortfolios` (Ardia et al., 2017) is an R package for constructing risk-based portfolios dedicated to portfolio managers and quantitative analysts. It provides a set of functionalities to build mean-variance, minimum variance, inverse-volatility weighted (Leote et al., 2012), equal-risk-contribution (Maillard et al. 2010), maximum diversification (Choueifaty and Coignard, 2008), and risk-efficient (Amenc et al., 2011) portfolios. Optimization is achieved with the R functions `solve.QP` and `slsqp`. Long or gross constraints can be added to the optimizer. As risk-based portfolios are mainly based on covariances, the package also provides a large set of covariance matrix estimators. A simulation study relying on the package is described in Ardia et al. (2017).

Functions

- `optimalPortfolio`: Optimization of portfolios.
- `meanEstimation`: Computation of expected returns.
- `covEstimation`: Computation of covariance matrix.
- `semidevEstimation`: Computation of semi-deviation.

Update

The latest version of the package is available at `https://github.com/ArdiaD/RiskPortfolios`.

Note

By using `RiskPortfolios` you agree to the following rules: (1) You must cite Ardia et al. (2017) in working papers and published papers that use `RiskPortfolios` (use `citation("RiskPortfolios")`), (2) you must place the URL `https://CRAN.R-project.org/package=RiskPortfolios` in a footnote to help others find `RiskPortfolios`, and (3) you assume all risk for the use of `RiskPortfolios`.

Author(s)

David Ardia, Kris Boudt and Jean-Philippe Gagnon-Fleury.

References

Amenc, N., Goltz, F., Martellini, L., Retowsky, P. (2011). Efficient indexation: An alternative to cap-weighted indices. *Journal of Investment Management* **9**(4), pp.1-23.

Ardia, D., Boudt, K. (2015). Implied expected returns and the choice of a mean-variance efficient portfolio proxy. *Journal of Portfolio Management* **41**(4), pp.66-81. doi:10.3905/jpm.2015.41.4.068

Ardia, D., Bolliger, G., Boudt, K., Gagnon-Fleury, J.-P. (2017). The Impact of covariance misspecification in risk-based portfolios. *Annals of Operations Research* **254**(1-2), pp.1-16. doi:10.1007/s1047901724747

Ardia, D., Boudt, K., Gagnon-Fleury, J.-P. (2017). RiskPortfolios: Computation of risk-based portfolios in R. *Journal of Open Source Software* **10**(2). doi:10.21105/joss.00171

Choueifaty, Y., Coignard, Y. (2008). Toward maximum diversification. *Journal of Portfolio Management* **35**(1), pp.40-51.

Choueifaty, Y., Froidure, T., Reynier, J. (2013). Properties of the most diversified portfolio. *Journal of Investment Strategies* **2**(2), pp.49-70.

Das, S., Markowitz, H., Scheid, J., Statman, M. (2010). Portfolio optimization with mental accounts. *Journal of Financial and Quantitative Analysis* **45**(2), pp.311-334.

DeMiguel, V., Garlappi, L., Uppal, R. (2009). Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy. *Review of Financial Studies* **22**(5), pp.1915-1953.

Fan, J., Zhang, J., Yu, K. (2012). Vast portfolio selection with gross-exposure constraints. *Journal of the American Statistical Association* **107**(498), pp.592-606.

Maillard, S., Roncalli, T., Teiletche, J. (2010). The properties of equally weighted risk contribution portfolios. *Journal of Portfolio Management* **36**(4), pp.60-70.

Martellini, L. (2008). Towards the design of better equity benchmarks. *Journal of Portfolio Management* **34**(4), Summer,pp.34-41.

`semidevEstimation` *Estimation of the semideviation*

Description

Function which computes the semideviation.

Usage

```
semidevEstimation(rets, control = list())
```

Arguments

`rets`	a $(T \times N)$ matrix of past returns.
`control`	control parameters (see *Details*).

Details

The argument `control` is a list that can supply any of the following components:

- `type` method used to compute the semideviation vector, among `'naive'` and `'ewma'` where:

  `'naive'` is used to compute the simple semideviation.

  `'ewma'` is used to compute the exponential weighted moving average semideviation. The data must be sorted from the oldest to the latest. See RiskMetrics (1996).

  The semideviation for one stock is computed as follows. First we select the returns which are smaller than the average of the past returns; we get a new vector of dimension $K \times 1, K \leq N$. Then, the weight w_i for each observation at its corresponding time t is computed as $w = \lambda^t$. We obtain a $K \times 1$ vector. The vector of weights is then normalized. Finally, the semideviation is obtained as the weighted standard deviation.

  Default: `type = 'naive'`.
- `lambda` decay parameter. Default: `lambda = 0.94`.

Value

A $(N \times 1)$ vector of semideviations.

Author(s)

David Ardia, Kris Boudt and Jean-Philippe Gagnon Fleury.

References

RiskMetrics (1996) *RiskMetrics Technical Document*. J. P. Morgan/Reuters.

Examples

```
# Load returns of assets or portfolios
data("Industry_10")
rets = Industry_10

# Naive semideviation estimation
semidevEstimation(rets)

# Naive estimation of the semideviation
semidevEstimation(rets, control = list(type = 'naive'))

# Ewma estimation of the semideviation. Default lambda = 0.94
semidevEstimation(rets, control = list(type = 'ewma'))

# Ewma estimation of the semideviation. lambda = 0.9
semidevEstimation(rets, control = list(type = 'ewma', lambda = 0.9))
```

Index