Package 'RPESE'

November 6, 2019

Type Package

Title Estimates of Standard Errors for Risk and Performance Measures

Version 1.0

Date 2019-10-26

Author Anthony Christidis <anthony.christidis@stat.ubc.ca>, Xin Chen <chenx26@uw.edu>

Maintainer Anthony Christidis <anthony.christidis@stat.ubc.ca>

Description Estimates of standard errors of popular risk and performance measures for asset or portfolio returns using methods as described in Chen and Martin (2019) <https://ssrn.com/abstract=3085672>.

Biarch true

License GPL (>= 2)

Imports RPEIF, RPEGLMEN, PerformanceAnalytics, xts, zoo, robustbase, boot, sandwich

Suggests testthat, R.rsp

Depends

RoxygenNote 6.1.1

VignetteBuilder R.rsp

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2019-11-06 12:50:02 UTC

R topics documented:

ES.SE 2
ESratio.SE 4
EstimatorSE 5
LPM.SE 6
Mean.SE 7

OmegaRatio.SE 9
printSE 10
RachevRatio.SE 11
SemiSD.SE 12
SharpeRatio.SE 13
SortinoRatio.SE 14
StdDev.SE 16
VaR.SE 17
VaRratio.SE 19

Index **21**

`ES.SE` *Standard Error Estimate for Expected Shortfall (ES) of Returns*

Description

`ES.SE` computes the standard error of the expected shortfall of the returns.

Usage

```
ES.SE(data, p = 0.95, se.method = c("IFiid", "IFcor", "IFcorAdapt",
  "IFcorPW", "BOOTiid", "BOOTcor")[1:2], cleanOutliers = FALSE,
  fitting.method = c("Exponential", "Gamma")[1], ...)
```

Arguments

`data`	Data of returns for one or multiple assets or portfolios.
`p`	Confidence level for calculation. Default value is p=0.95.
`se.method`	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One or a combination of: `"IFiid"` (default), `"IFcor"` (default), `"IFcorPW"`, `"IFcorAdapt"`, `"BOOTiid"` or `"BOOTcor"`.
`cleanOutliers`	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.
`fitting.method`	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
`...`	Additional parameters.

Value

A vector or a list depending on `se.method`.

Background

This function provides several estimation methods for the Expected Shortfall (ES) (also called Expected Tail Loss (ETL) or Conditional Value at Risk (CVaR)) of a return series and the Component ES (ETL/CVaR) of a portfolio.

At a preset probability level denoted c, which typically is between 1 and 5 per cent, the ES of a return series is the negative value of the expected value of the return when the return is less than its c-quantile. Unlike value-at-risk, conditional value-at-risk has all the properties a risk measure should have to be coherent and is a convex function of the portfolio weights (Pflug, 2000). With a sufficiently large data set, you may choose to estimate ES with the sample average of all returns that are below the c empirical quantile. More efficient estimates of VaR are obtained if a (correct) assumption is made on the return distribution, such as the normal distribution. If your return series is skewed and/or has excess kurtosis, Cornish-Fisher estimates of ES can be more appropriate. For the ES of a portfolio, it is also of interest to decompose total portfolio ES into the risk contributions of each of the portfolio components. For the above mentioned ES estimators, such a decomposition is possible in a financially meaningful way.

Note

The option to `invert` the ES measure should appease both academics and practitioners. The mathematical definition of ES as the negative value of extreme losses will (usually) produce a positive number. Practitioners will argue that ES denotes a loss, and should be internally consistent with the quantile (a negative number). For tables and charts, different preferences may apply for clarity and compactness. As such, we provide the option, and set the default to TRUE to keep the return consistent with prior versions of PerformanceAnalytics, but make no value judgement on which approach is preferable.

Author(s)

Xin Chen, <chenx26@uw.edu>

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

References

Boudt, Kris, Peterson, Brian, and Christophe Croux. 2008. Estimation and decomposition of downside risk for portfolios with non-normal returns. 2008. The Journal of Risk, vol. 11, 79-103.

Cont, Rama, Deguest, Romain and Giacomo Scandolo. Robustness and sensitivity analysis of risk measurement procedures. Financial Engineering Report No. 2007-06, Columbia University Center for Financial Engineering.

Laurent Favre and Jose-Antonio Galeano. Mean-Modified Value-at-Risk Optimization with Hedge Funds. Journal of Alternative Investment, Fall 2002, v 5.

Martellini, Lionel, and Volker Ziemann. Improved Forecasts of Higher-Order Comoments and Implications for Portfolio Selection. 2007. EDHEC Risk and Asset Management Research Centre working paper.

Pflug, G. Ch. Some remarks on the value-at-risk and the conditional value-at-risk. In S. Uryasev, ed., Probabilistic Constrained Optimization: Methodology and Applications, Dordrecht: Kluwer, 2000, 272-281.

Scaillet, Olivier. Nonparametric estimation and sensitivity analysis of expected shortfall. Mathematical Finance, 2002, vol. 14, 74-86.

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
# Computing the standard errors for
# the two influence functions based approaches
ES.SE(edhec, se.method=c("IFiid","IFcor"),
      cleanOutliers=FALSE,
      fitting.method=c("Exponential", "Gamma")[1])
```

`ESratio.SE` *Standard Error Estimate for Expected Shortfall Ratio (ESratio) of Returns*

Description

`ESratio.SE` computes the standard error of the expected shortfall ratio of the returns.

Usage

```
ESratio.SE(data, alpha = 0.1, rf = 0, se.method = c("IFiid", "IFcor",
  "IFcorAdapt", "IFcorPW", "BOOTiid", "BOOTcor")[c(1, 3)],
  cleanOutliers = FALSE, fitting.method = c("Exponential", "Gamma")[1],
  ...)
```

Arguments

`data`	Data of returns for one or multiple assets or portfolios.
`alpha`	Lower tail probability.
`rf`	Risk-free interest rate.
`se.method`	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One or a combination of: `"IFiid"` (default), `"IFcor"`, `"IFcorPW"`, `"IFcorAdapt"` (default), `"BOOTiid"` or `"BOOTcor"`.
`cleanOutliers`	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.
`fitting.method`	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
`...`	Additional parameters.

Value

A vector or a list depending on se.method.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
# Computing the standard errors for
# the two influence functions based approaches
ESratio.SE(edhec, se.method=c("IFiid","IFcorAdapt"),
           cleanOutliers=FALSE,
           fitting.method=c("Exponential", "Gamma")[1])
```

EstimatorSE *Wrapper Function for Standard Errors Estimates Functions*

Description

EstimatorSE computes the standard error for specified risk and performance measures.

Usage

```
EstimatorSE(data, estimator.fun = c("Mean", "SD", "VaR", "ES", "SR",
  "SoR", "ESratio", "VaRratio", "SoR", "LPM", "OmegaRatio", "SemiSD",
  "RachevRatio"), se.method = c("IFiid", "IFcor", "IFcorAdapt",
  "IFcorPW", "BOOTiid", "BOOTcor"), cleanOutliers = FALSE,
  fitting.method = c("Exponential", "Gamma")[1], a = 0.3, b = 0.7,
  ...)
```

Arguments

data	Data of returns for one or multiple assets or portfolios.
estimator.fun	Risk or performance measure to compute estimates of standard errors.
se.method	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One of: "IFiid", "IFcor", "IFcorAdapt", "IFcorPW", "BOOTiid", "BOOTcor", or "none".
cleanOutliers	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.

fitting.method	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
a	First adaptive method parameter.
b	Second adaptive method parameter.
...	Additional parameters.

Value

A vector standard error estimates.

Author(s)

Xin Chen, <chenx26@uw.edu>

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
# Computing the standard errors for
# the three influence functions based approaches
EstimatorSE(edhec[,"CA"], se.method=c("IFcor"),
            cleanOutliers=FALSE,
            fitting.method=c("Exponential", "Gamma")[1])
```

LPM.SE *Standard Error Estimate for Lower Partial Moment (LPM) of Returns*

Description

LPM.SE computes the standard error of the LPM of the returns.

Usage

```
LPM.SE(data, const = 0, k = 1, se.method = c("IFiid", "IFcor",
  "IFcorAdapt", "IFcorPW", "BOOTiid", "BOOTcor")[1:2],
  cleanOutliers = FALSE, fitting.method = c("Exponential", "Gamma")[1],
  ...)
```

Arguments

data	Data of returns for one or multiple assets or portfolios.
const	Constant threshold.
k	Range parameter for the shape of the IF (the SD gets multiplied k times).
se.method	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One or a combination of: "IFiid" (default), "IFcor" (default), "IFcorPW", "IFcorAdapt", "BOOTiid" or "BOOTcor".
cleanOutliers	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.
fitting.method	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
...	Additional parameters.

Value

A vector or a list depending on se.method.

Author(s)

Xin Chen, <chenx26@uw.edu>

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
# Computing the standard errors for
# the two influence functions based approaches
LPM.SE(edhec, se.method=c("IFiid","IFcor"),
       cleanOutliers=FALSE,
       fitting.method=c("Exponential", "Gamma")[1])
```

Mean.SE *Standard Error Estimate for Mean of Returns*

Description

Mean.SE computes the standard error of the mean of the returns.

Usage

```
Mean.SE(data, se.method = c("IFiid", "IFcor", "IFcorAdapt", "IFcorPW",
  "BOOTiid", "BOOTcor")[c(1, 3)], cleanOutliers = FALSE,
  fitting.method = c("Exponential", "Gamma")[1], ...)
```

Arguments

data	Data of returns for one or multiple assets or portfolios.
se.method	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One or a combination of: "IFiid" (default), "IFcor", "IFcorPW", "IFcorAdapt" (default), "BOOTiid" or "BOOTcor".
cleanOutliers	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.
fitting.method	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
...	Additional parameters.

Value

A vector or a list depending on se.method

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
# Computing the standard errors for
# the two influence functions based approaches
Mean.SE(edhec, se.method=c("IFiid","IFcorAdapt"),
        cleanOutliers=FALSE,
        fitting.method=c("Exponential", "Gamma")[1])
```

OmegaRatio.SE *Standard Error Estimate for Omega Ratio of Returns*

Description

OmegaRatio.SE computes the standard error of the Omega ratio of the returns.

Usage

```
OmegaRatio.SE(data, const = 0, k = 4, se.method = c("IFiid", "IFcor",
  "IFcorAdapt", "IFcorPW", "BOOTiid", "BOOTcor")[c(1, 3)],
  cleanOutliers = FALSE, fitting.method = c("Exponential", "Gamma")[1],
  ...)
```

Arguments

data	Data of returns for one or multiple assets or portfolios.
const	Constant threshold.
k	Range parameter for the shape of the IF (the SD gets multiplied k times).
se.method	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One or a combination of: "IFiid" (default), "IFcor", "IFcorPW", "IFcorAdapt" (default), "BOOTiid", "BOOTcor".
cleanOutliers	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.
fitting.method	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
...	Additional parameters.

Value

A vector or a list depending on se.method.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
```

```
# Computing the standard errors for
# the two influence functions based approaches
OmegaRatio.SE(edhec, se.method=c("IFiid","IFcorAdapt")[1],
              cleanOutliers=FALSE,
              fitting.method=c("Exponential", "Gamma")[1])
```

---

printSE *Formatted Output for Standard Errors Functions in RPESE*

Description

printSE returns a formatted output from standard error functions from RPESE.

Usage

```
printSE(SE.data, round.digit = 3, round.out = TRUE)
```

Arguments

SE.data	Standard error estimates output from RPESE functions.
round.digit	Number of digits for rounding.
round.out	Round data (TRUE) with round.digit number of digits. Default is TRUE.

Value

A data frame with formatted output from standard error functions from RPESE.

Author(s)

Xin Chen, <chenx26@uw.edu>

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
# Computing the standard errors for
# the two influence functions based approaches
ES.out <- ES.SE(edhec, se.method=c("IFiid","IFcor"),
                cleanOutliers=FALSE,
                fitting.method=c("Exponential", "Gamma")[1])
# Print the output
printSE(ES.out)
```

RachevRatio.SE *Standard Error Estimate for Rachev Ratio of Returns*

Description

RachevRatio.SE computes the standard error of the Rachev ratio of the returns.

Usage

```
RachevRatio.SE(data, alpha = 0.1, beta = 0.1, rf = 0,
  se.method = c("IFiid", "IFcor", "IFcorAdapt", "IFcorPW", "BOOTiid",
  "BOOTcor")[c(1, 3)], cleanOutliers = FALSE,
  fitting.method = c("Exponential", "Gamma")[1], ...)
```

Arguments

data	Data of returns for one or multiple assets or portfolios.
alpha	Lower tail probability.
beta	Upper tail probability.
rf	Risk-free interest rate.
se.method	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One or a combination of: "IFiid" (default), "IFcor", "IFcorPW", "IFcorAdapt" (default), "BOOTiid" or "BOOTcor".
cleanOutliers	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.
fitting.method	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
...	Additional parameters.

Value

A vector or a list depending on se.method.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
```

```
                "RV", "SS", "FOF")
# Computing the standard errors for
# the two influence functions based approaches
RachevRatio.SE(edhec, se.method=c("IFiid","IFcorAdapt"),
               cleanOutliers=FALSE,
               fitting.method=c("Exponential", "Gamma")[1])
```

SemiSD.SE *Standard Error Estimate for Semi-Standared Deviation (SemiSD) of Returns*

Description

SemiSD.SE computes the standard error of the SSD of the returns.

Usage

```
SemiSD.SE(data, rf = 0, se.method = c("IFiid", "IFcor", "IFcorAdapt",
  "IFcorPW", "BOOTiid", "BOOTcor")[1:2], cleanOutliers = FALSE,
  fitting.method = c("Exponential", "Gamma")[1], ...)
```

Arguments

data	Data of returns for one or multiple assets or portfolios.
rf	Risk-free interest rate.
se.method	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One or a combination of: "IFiid" (default), "IFcor" (default), "IFcorPW", "IFcorAdapt", "BOOTiid", "BOOTcor", or "none".
cleanOutliers	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.
fitting.method	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
...	Additional parameters.

Value

A vector or a list depending on se.method.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
# Computing the standard errors for
# the two influence functions based approaches
SemiSD.SE(edhec, se.method=c("IFiid","IFcor"),
          cleanOutliers=FALSE,
          fitting.method=c("Exponential", "Gamma")[1])
```

SharpeRatio.SE *Standard Error Estimate for Sharpe Ratio of Returns*

Description

SharpeRatio.SE computes the standard error of the Sharpe ratio of the returns.

Usage

```
SharpeRatio.SE(data, Rf = 0, se.method = c("IFiid", "IFcor",
  "IFcorAdapt", "IFcorPW", "BOOTiid", "BOOTcor")[c(1, 3)],
  cleanOutliers = FALSE, fitting.method = c("Exponential", "Gamma")[1],
  ...)
```

Arguments

data	Data of returns for one or multiple assets or portfolios.
Rf	Risk free rate.
se.method	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One or a combination of: "IFiid" (default), "IFcor", "IFcorPW", "IFcorAdapt" (default), "BOOTiid" or "BOOTcor".
cleanOutliers	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.
fitting.method	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
...	Additional parameters.

Details

The Sharpe ratio is simply the return per unit of risk (represented by variability). In the classic case, the unit of risk is the standard deviation of the returns.

$$\frac{\overline{(R_a - R_f)}}{\sqrt{\sigma_{(R_a - R_f)}}}$$

William Sharpe now recommends `InformationRatio` preferentially to the original Sharpe Ratio.

The higher the Sharpe ratio, the better the combined performance of "risk" and return.

As noted, the traditional Sharpe Ratio is a risk-adjusted measure of return that uses standard deviation to represent risk.

Value

A vector or a list depending on `se.method`.

Author(s)

Xin Chen, <chenx26@uw.edu>

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
# Computing the standard errors for
# the two influence functions based approaches
SharpeRatio.SE(edhec, se.method=c("IFiid","IFcorAdapt"),
               cleanOutliers=FALSE,
               fitting.method=c("Exponential", "Gamma")[1])
```

`SortinoRatio.SE` *Standard Error Estimate for Sortino Ratio of Returns*

Description

`SortinoRatio.SE` computes the standard error of the Sortino ratio of the returns.

Usage

```
SortinoRatio.SE(data, MAR = 0, threshold = c("mean", "const")[1],
  se.method = c("IFiid", "IFcor", "IFcorAdapt", "IFcorPW", "BOOTiid",
  "BOOTcor")[c(1, 3)], cleanOutliers = FALSE,
  fitting.method = c("Exponential", "Gamma")[1], ...)
```

Arguments

data	Data of returns for one or multiple assets or portfolios.
MAR	Minimum Acceptable Return for threshold.
threshold	Parameter to determine whether we use a "mean" or "const" threshold.
se.method	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One or a combination of: "IFiid" (default), "IFcor", "IFcorPW", "IFcorAdapt" (default), "BOOTiid" or "BOOTcor".
cleanOutliers	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.
fitting.method	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
...	Additional parameters.

Details

Sortino proposed an improvement on the Sharpe Ratio to better account for skill and excess performance by using only downside semivariance as the measure of risk.

Sortino contends that risk should be measured in terms of not meeting the investment goal. This gives rise to the notion of “Minimum Acceptable Return” or MAR. All of Sortino’s proposed measures include the MAR, and are more sensitive to downside or extreme risks than measures that use volatility(standard deviation of returns) as the measure of risk.

Choosing the MAR carefully is very important, especially when comparing disparate investment choices. If the MAR is too low, it will not adequately capture the risks that concern the investor, and if the MAR is too high, it will unfavorably portray what may otherwise be a sound investment. When comparing multiple investments, some papers recommend using the risk free rate as the MAR. Practitioners may wish to choose one MAR for consistency, several standardized MAR values for reporting a range of scenarios, or a MAR customized to the objective of the investor.

Value

A vector or a list depending on se.method.

Author(s)

Xin Chen, <chenx26@uw.edu>

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
# Computing the standard errors for
# the two influence functions based approaches
SortinoRatio.SE(edhec[,"CA"], se.method=c("IFiid","IFcorAdapt"),
                cleanOutliers=FALSE,
                fitting.method=c("Exponential", "Gamma")[1])
```

StdDev.SE *Standard Error Estimate for Standard Deviation (StdDev) of Returns*

Description

StdDev.SE computes the standard error of the standard deviation of the returns.

Usage

```
StdDev.SE(data, se.method = c("IFiid", "IFcor", "IFcorAdapt", "IFcorPW",
  "BOOTiid", "BOOTcor")[1:2], cleanOutliers = FALSE,
  fitting.method = c("Exponential", "Gamma")[1], ...)
```

Arguments

data	Data of returns for one or multiple assets or portfolios.
se.method	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One or a combination of: "IFiid" (default), "IFcor" (default), "IFcorPW", "IFcorAdapt" (default), "BOOTiid" or "BOOTcor".
cleanOutliers	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.
fitting.method	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
...	Additional parameters.

Value

A vector or a list depending on se.method.

Author(s)

Xin Chen, <chenx26@uw.edu>

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
# Computing the standard errors for
# the two influence functions based approaches
StdDev.SE(edhec, se.method=c("IFiid","IFcor","IFcorAdapt"),
          cleanOutliers=FALSE,
          fitting.method=c("Exponential", "Gamma")[1])
```

VaR.SE *Standard Error Estimate for Value-at-Risk (VaR) of Returns*

Description

VaR.SE computes the standard error of the value-at-risk of the returns.

Usage

```
VaR.SE(data = NULL, p = 0.95, se.method = c("IFiid", "IFcor",
  "IFcorAdapt", "IFcorPW", "BOOTiid", "BOOTcor")[1:2],
  cleanOutliers = FALSE, fitting.method = c("Exponential", "Gamma")[1],
  ...)
```

Arguments

data	Data of returns for one or multiple assets or portfolios.
p	Confidence level for calculation. Default is p=0.95.
se.method	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One or a combination of: "IFiid" (default), "IFcor" (default), "IFcorPW", "IFcorAdapt", "BOOTiid" or "BOOTcor".
cleanOutliers	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.
fitting.method	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
...	Additional parameters.

Value

A vector or a list depending on `se.method`.

Background

This function provides several estimation methods for the Value at Risk (typically written as VaR) of a return series and the Component VaR of a portfolio. Take care to capitalize VaR in the commonly accepted manner, to avoid confusion with var (variance) and VAR (vector auto-regression). VaR is an industry standard for measuring downside risk. For a return series, VaR is defined as the high quantile (e.g. ~a 95 quantile) of the negative value of the returns. This quantile needs to be estimated. With a sufficiently large data set, you may choose to utilize the empirical quantile calculated using `quantile`. More efficient estimates of VaR are obtained if a (correct) assumption is made on the return distribution, such as the normal distribution. If your return series is skewed and/or has excess kurtosis, Cornish-Fisher estimates of VaR can be more appropriate. For the VaR of a portfolio, it is also of interest to decompose total portfolio VaR into the risk contributions of each of the portfolio components. For the above mentioned VaR estimators, such a decomposition is possible in a financially meaningful way.

Note

The option to `invert` the VaR measure should appease both academics and practitioners. The mathematical definition of VaR as the negative value of a quantile will (usually) produce a positive number. Practitioners will argue that VaR denotes a loss, and should be internally consistent with the quantile (a negative number). For tables and charts, different preferences may apply for clarity and compactness. As such, we provide the option, and set the default to TRUE to keep the return consistent with prior versions of PerformanceAnalytics, but make no value judgment on which approach is preferable.

The prototype of the univariate Cornish Fisher VaR function was completed by Prof. Diethelm Wuertz. All corrections to the calculation and error handling are the fault of Brian Peterson.

Author(s)

Anthony-Alexander Christidis, `<anthony.christidis@stat.ubc.ca>`

References

Boudt, Kris, Peterson, Brian, and Christophe Croux. 2008. Estimation and decomposition of downside risk for portfolios with non-normal returns. 2008. The Journal of Risk, vol. 11, 79-103.

Cont, Rama, Deguest, Romain and Giacomo Scandolo. Robustness and sensitivity analysis of risk measurement procedures. Financial Engineering Report No. 2007-06, Columbia University Center for Financial Engineering.

Denton M. and Jayaraman, J.D. Incremental, Marginal, and Component VaR. Sunguard. 2004.

Epperlein, E., Smillie, A. Cracking VaR with kernels. RISK, 2006, vol. 19, 70-74.

Gourieroux, Christian, Laurent, Jean-Paul and Olivier Scaillet. Sensitivity analysis of value at risk. Journal of Empirical Finance, 2000, Vol. 7, 225-245.

Keel, Simon and Ardia, David. Generalized marginal risk. Aeris CAPITAL discussion paper.

Laurent Favre and Jose-Antonio Galeano. Mean-Modified Value-at-Risk Optimization with Hedge Funds. Journal of Alternative Investment, Fall 2002, v 5.

Martellini, Lionel, and Volker Ziemann. Improved Forecasts of Higher-Order Comoments and Implications for Portfolio Selection. 2007. EDHEC Risk and Asset Management Research Centre working paper.

Zangari, Peter. A VaR Methodology for Portfolios that include Options. 1996. RiskMetrics Monitor, First Quarter, 4-12.

Rockafellar, Terry and Uryasev, Stanislav. Optimization of Conditional VaR. The Journal of Risk, 2000, vol. 2, 21-41.

Dowd, Kevin. Measuring Market Risk, John Wiley and Sons, 2010.

Jorian, Phillippe. Value at Risk, the new benchmark for managing financial risk. 3rd Edition, McGraw Hill, 2006.

Hallerback, John. "Decomposing Portfolio Value-at-Risk: A General Analysis", 2003. The Journal of Risk vol 5/2.

Yamai and Yoshiba (2002). "Comparative Analyses of Expected Shortfall and Value-at-Risk: Their Estimation Error, Decomposition, and Optimization", Bank of Japan.

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
# Computing the standard errors for
# the two influence functions based approaches
VaR.SE(edhec, se.method=c("IFiid","IFcor"),
       cleanOutliers=FALSE,
       fitting.method=c("Exponential", "Gamma")[1])
```

`VaRratio.SE` *Standard Error Estimate for Value-at-Risk Ratio (VaRratio) of Returns*

Description

`VaRratio.SE` computes the standard error of the value-at-risk ratio of the returns.

Usage

```
VaRratio.SE(data, alpha = 0.1, rf = 0, se.method = c("IFiid",
  "IFcor", "IFcorAdapt", "IFcorPW", "BOOTiid", "BOOTcor")[c(1, 3)],
  cleanOutliers = FALSE, fitting.method = c("Exponential", "Gamma")[1],
  ...)
```

Arguments

data	Data of returns for one or multiple assets or portfolios.
alpha	The tail probability of interest.
rf	Risk-free interest rate.
se.method	A character string indicating which method should be used to compute the standard error of the estimated standard deviation. One or a combination of: "IFiid" (default), "IFcor", "IFcorPW", "IFcorAdapt" (default), "BOOTiid" or "BOOTcor".
cleanOutliers	Boolean variable to indicate whether the pre-whitenning of the influence functions TS should be done through a robust filter.
fitting.method	Distribution used in the standard errors computation. Should be one of "Exponential" (default) or "Gamma".
...	Additional parameters.

Value

A vector or a list depending on se.method.

Author(s)

Anthony-Alexander Christidis, <anthony.christidis@stat.ubc.ca>

Examples

```
# Loading data from PerformanceAnalytics
data(edhec, package = "PerformanceAnalytics")
class(edhec)
# Changing the data colnames
names(edhec) = c("CA", "CTA", "DIS", "EM", "EMN",
                 "ED", "FIA", "GM", "LS", "MA",
                 "RV", "SS", "FOF")
# Computing the standard errors for
# the two influence functions based approaches
VaRratio.SE(edhec, se.method=c("IFiid","IFcorAdapt"),
            cleanOutliers=FALSE,
            fitting.method=c("Exponential", "Gamma")[1])
```

Index

ES.SE, 2
ESratio.SE, 4
EstimatorSE, 5

InformationRatio, *14*

LPM.SE, 6

Mean.SE, 7

OmegaRatio.SE, 9

printSE, 10

quantile, *18*

RachevRatio.SE, 11

SemiSD.SE, 12
SharpeRatio.SE, 13
SortinoRatio.SE, 14
StdDev.SE, 16

VaR.SE, 17
VaRratio.SE, 19