Package 'PerformanceAnalytics'

April 11, 2026

Type Package

Title Econometric Tools for Performance and Risk Analysis

Version 2.1.0

Date 2026-04-05

Description Collection of econometric functions for performance and risk analysis. In addition to standard risk and performance metrics, this package aims to aid practitioners and researchers in utilizing the latest research in analysis of non-normal return streams. In general, it is most tested on return (rather than price) data on a regular scale, but most functions will work with irregular return data as well, and increasing numbers of functions will work with P&L or price data where possible.

Imports methods, quadprog, zoo

Depends R (>= 4.0.0), xts (>= 0.10.0)

Suggests dygraphs, Hmisc, MASS, quantmod, gamlss, gamlss.dist, robustbase, quantreg, ggplot2, vdiffr, RColorBrewer, googleVis, plotly, gridExtra, ggpubr, nloptr, rgenoud, RPESE, RobStatTM, fit.models, R.rsp, testthat (>= 3.0.0), covr

VignetteBuilder R.rsp

License GPL-2 | GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/braverock/PerformanceAnalytics

Copyright (c) 2004-2026

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation yes

Author Brian G. Peterson [cre, aut, cph],
Peter Carl [aut, cph],
Kris Boudt [ctb, cph],
Ross Bennett [ctb],

Joshua Ulrich [ctb],
Eric Zivot [ctb],
Dries Cornilly [ctb],
Eric Hung [ctb],
Matthieu Lestel [ctb],
Kyle Balkissoon [ctb],
Diethelm Wuertz [ctb],
Anthony Alexander Christidis [ctb],
R. Douglas Martin [ctb],
Zeheng Zenith Zhou [ctb],
Justin M. Shea [ctb],
Dhairya Jain [ctb],
Talgat Daniyarov [ctb]

Maintainer Brian G. Peterson <brian@braverock.com>

Repository CRAN

Date/Publication 2026-04-11 09:40:02 UTC

Contents

PerformanceAnalytics-package

Econometric tools for performance and risk analysis.

Description

PerformanceAnalytics provides an package of econometric functions for performance and risk analysis of financial instruments or portfolios. This package aims to aid practitioners and researchers in using the latest research for analysis of both normally and non-normally distributed return streams.

Details

We created this package to include functionality that has been appearing in the academic literature on performance analysis and risk over the past several years, but had no functional equivalent in . In doing so, we also found it valuable to have wrappers for some functionality with good defaults and naming consistent with common usage in the finance literature.

In general, this package requires return (rather than price) data. Almost all of the functions will work with any periodicity, from annual, monthly, daily, to even minutes and seconds, either regular or irregular.

The following sections cover Time Series Data, Performance Analysis, Risk Analysis (with a separate treatment of VaR), Summary Tables of related statistics, Charts and Graphs, a variety of Wrappers and Utility functions, and some thoughts on work yet to be done.

In this summary, we attempt to provide an overview of the capabilities provided by PerformanceAnalytics and pointers to other literature and resources in useful for performance and risk analysis. We hope that this summary and the accompanying package and documentation partially fill a hole in the tools available to a financial engineer or analyst.

Time Series Data

Not all, but many of the measures in this package require time series data. PerformanceAnalytics uses the xts package for managing time series data for several reasons. Besides being fast and efficient, xts includes functions that test the data for periodicity and draw attractive and readable time-based axes on charts. Another benefit is that xts provides compatability with Rmetrics' timeSeries, zoo and other time series classes, such that PerformanceAnalytics functions that return a time series will return the results in the same format as the object that was passed in. Jeff Ryan and Joshua Ulrich, the authors of xts, have been extraordinarily helpful to the development of PerformanceAnalytics and we are very grateful for their contributions to the community. The xts package extends the excellent zoo package written by Achim Zeileis and Gabor Grothendieck. zoo provides more general time series support, whereas xts provides functionality that is specifically aimed at users in finance.

Users can easily load returns data as time series for analysis with PerformanceAnalytics by using the Return.read function. The Return.read function loads csv files of returns data where the data is organized as dates in the first column and the returns for the period in subsequent columns. See read.zoo and as.xts if more flexibility is needed.

The functions described below assume that input data is organized with asset returns in columns and dates represented in rows. All of the metrics in PerformanceAnalytics are calculated by column and return values for each column in the results. This is the default arrangement of time series data in xts.

Some sample data is available in the managers dataset. It is an xts object that contains columns of monthly returns for six hypothetical asset managers (HAM1 through HAM6), the EDHEC Long-Short Equity hedge fund index, the S&P 500 total returns, and total return series for the US Treasury 10-year bond and 3-month bill. Monthly returns for all series end in December 2006 and begin at different periods starting from January 1996. That data set is used extensively in our examples and should serve as a model for formatting your data.

For retrieving market data from online sources, see quantmod's getSymbols function for downloading prices and chartSeries for graphing price data. Also see the tseries package for the function get.hist.quote. Look at xts's to.period function to rationally coerce irregular price data into regular data of a specified periodicity. The aggregate function has methods for tseries and zoo timeseries data classes to rationally coerce irregular data into regular data of the correct periodicity.

Finally, see the function Return.calculate for calculating returns from prices.

Risk Analysis

Many methods have been proposed to measure, monitor, and control the risks of a diversified portfolio. Perhaps a few definitions are in order on how different risks are generally classified. *Market Risk* is the risk to the portfolio from a decline in the market price of instruments in the portfolio. *Liquidity Risk* is the risk that the holder of an instrument will find that a position is illiquid, and will incur extra costs in unwinding the position resulting in a less favorable price for the instrument. In extreme cases of liquidity risk, the seller may be unable to find a buyer for the instrument at all, making the value unknowable or zero. *Credit Risk* encompasses *Default Risk*, or the risk that promised payments on a loan or bond will not be made, or that a convertible instrument will not be converted in a timely manner or at all. There are also *Counterparty Risks* in over the counter markets, such as those for complex derivatives. Tools have evolved to measure all these different components of risk. Processes must be put into place inside a firm to monitor the changing risks in a portfolio, and to control the magnitude of risks. For an extensive treatment of these topics, see Litterman, Gumerlock, et. al.(1998). For our purposes, PerformanceAnalytics tends to focus on market and liquidity risk.

The simplest risk measure in common use is volatility, usually modeled quantitatively with a univariate standard deviation on a portfolio. See sd. Volatility or Standard Deviation is an appropriate risk measure when the distribution of returns is normal or resembles a random walk, and may be annualized using sd.annualized, or the equivalent function sd.multiperiod for scaling to an arbitrary number of periods. Many assets, including hedge funds, commodities, options, and even most common stocks over a sufficiently long period, do not follow a normal distribution. For such common but non-normally distributed assets, a more sophisticated approach than standard deviation/volatility is required to adequately model the risk.

Markowitz, in his Nobel acceptance speech and in several papers, proposed that SemiVariance would be a better measure of risk than variance. See *Zin, Markowitz, Zhao (2006)*. This measure is also called SemiDeviation. The more general case is DownsideDeviation, as proposed by *Sortino and Price(1994)*, where the minimum acceptable return (MAR) is a parameter to the function. It is interesting to note that variance and mean return can produce a smoothly elliptical efficient frontier

for portfolio optimization using `solve.QP` or `portfolio.optim` or `fPortfolio`. Use of semivariance or many other risk measures will not necessarily create a smooth ellipse, causing significant additional difficulties for the portfolio manager trying to build an optimal portfolio. We'll leave a more complete treatment and implementation of portfolio optimization techniques for another time.

Another very widely used downside risk measures is analysis of drawdowns, or loss from peak value achieved. The simplest method is to check the `maxDrawdown`, as this will tell you the worst cumulative loss ever sustained by the asset. If you want to look at all the drawdowns, you can `findDrawdowns` and `sortDrawdowns` in order from worst/major to smallest/minor. The `UpDownRatios` function will give you some insight into the impacts of the skewness and kurtosis of the returns, and letting you know how length and magnitude of up or down moves compare to each other. You can also plot drawdowns with `chart.Drawdown`.

One of the newer statistical methods developed for analyzing the risk of financial instruments is `Omega`. Omega analytically constructs a cumulative distribution function, in a manner similar to `chart.QQPlot`, but then extracts additional information from the location and slope of the derived function at the point indicated by the risk quantile that the researcher is interested in. Omega seeks to combine a large amount of data about the shape, magnitude, and slope of the distribution into one method. The academic literature is still exploring the best manner to use Omega in a risk measurement and control process, or in portfolio construction.

Any risk measure should be viewed with suspicion if there are not a large number of historical observations of returns for the asset in question available. Depending on the measure, the number of observations required will vary greatly from a statistical standpoint. As a heuristic rule, ideally you will have data available on how the instrument performed through several economic cycles and shocks. When such a long history is not available, the investor or researcher has several options. A full discussion of the various approaches is beyond the scope of this introduction, so we will merely touch on several areas that an interested party may wish to explore in additional detail. Examining the returns of assets with a similar style, industry, or asset class to which the asset in question is highly correlated and shares other characteristics can be quite informative. Factor analysis may be used to uncover specific risk factors where transparency is not available. Various resampling (see `tsbootstrap`) and simulation methods are available in R to construct an artificially long distribution for testing. If you use a method such as Monte Carlo simulation or the bootstrap, it is often valuable to use `chart.Boxplot` to visualize the different estimates of the risk measure produced by the simulation, to see how small (or wide) a range the estimates cover, and thus gain a level of confidence with the results. Proceed with extreme caution when your historical data is lacking. Problems with lack of historical data are a major reason why many institutional investors will not invest in an alternative asset without several years of historical return data available.

Value at Risk (VaR) and Expected Shortfall (ES, ETL, CVaR)

Traditional mean-VaR: In the early 90's, academic literature started referring to "value at risk", typically written as VaR. Take care to capitalize VaR in the commonly accepted manner, to avoid confusion with var (variance) and VAR (vector auto-regression). With a sufficiently large data set, you may choose to use a non-parametric VaR estimation method using the historical distribution and the probability quantile of the distribution calculated using `qnorm`. The negative return at the correct quantile (usually 95% or 99%), is the non-parametric VaR estimate. J.P. Morgan's RiskMetrics parametric mean-VaR was published in 1994 and this methodology for estimating parametric mean-VaR has become what people are generally referring to as "VaR" and what we have implemented as `VaR` with `method="historical"`. See *Return to RiskMetrics: Evolution of a Standard* at

https://www.msci.com/documents/10199/dbb975aa-5dc2-4441-aa2d-ae34ab5f0945. Parametric traditional VaR does a better job of accounting for the tails of the distribution by more precisely estimating the tails below the risk quantile. It is still insufficient if the assets have a distribution that varies widely from normality. That is available in VaR with method="gaussian".

The R package VaR, now orphaned, contains methods for simulating and estimating lognormal VaR.norm and generalized Pareto VaR.gpd distributions to overcome some of the problems with nonparametric or parametric mean-VaR calculations on a limited sample size. There is also a VaR.backtest function to apply simulation methods to create a more robust estimate of the potential distribution of losses. The VaR package also provides plots for its functions. We will attempt to incorporate this orphaned functionality in PerformanceAnalytics in an upcoming release, and would welcome patches or pull requests in this direction.

Modified Cornish-Fisher VaR: The limitations of traditional mean-VaR are all related to the use of a symmetrical distribution function. Use of simulations, resampling, or Pareto distributions all help in making a more accurate prediction, but they are still flawed for assets with significantly non-normal (skewed and/or kurtotic) distributions. *Huisman (1999)* and *Favre and Galleano (2002)* propose to overcome this extensively documented failing of traditional VaR by directly incorporating the higher moments of the return distribution into the VaR calculation.

This VaR measure incorporates skewness and kurtosis via an analytical estimation using a Cornish-Fisher (special case of a Taylor) expansion. The resulting measure is referred to variously as "Cornish-Fisher VaR" or "Modified VaR". We provide this measure in function VaR with method="modified". Modified VaR produces the same results as traditional mean-VaR when the return distribution is normal, so it may be used as a direct replacement. Many papers in the finance literature have reached the conclusion that Modified VaR is a superior measure, and may be substituted in any case where mean-VaR would previously have been used. Note that estimates for Cornish Fisher modified VaR and ES may be unstable with small sample sizes. See Martin and Arora (2017) for more details.

Conditional VaR and Expected Shortfall: We have implemented Conditional Value at Risk, also called Expected Tail Loss or Expected Shortfall (not to be confused with shortfall probability, which is much less useful), in function ES. Expected Shortfall attempts to measure the magnitude of the average loss exceeding the traditional mean-VaR. Expected Shortfall has proven to be a reasonable risk predictor for many asset classes. We have provided traditional historical, Gaussian and modified Cornish-Fisher measures of Expected Shortfall by using method="historical", method="gaussian" or method="modified". See *Uryasev(2000)* and *Sherer and Martin(2005)* for more information on Conditional Value at Risk and Expected Shortfall. Please note that your mileage will vary; expect that values obtained from the normal distribution may differ radically from the real situation, depending on the assets under analysis.

Multivariate extensions to risk measures: We have extended all moments calculations to work in a multivariate portfolio context. In a portfolio context the multivariate moments are generally to be preferred to their univariate counterparts, so that all information is available to subsequent calculations. Both the VaR and ES functions allow calculation of metrics in a portfolio context when weights and a portfolio_method are passed into the function call.

Marginal, Incremental, and Component VaR: Marginal VaR is the difference between the VaR of the portfolio without the asset in question and the entire portfolio. The VaR function calculates Marginal VaR for all instruments in the portfolio if you set method="marginal". Marginal VaR as provided here may use traditional mean-VaR or Modified VaR for the calculation. Per *Artzner,et.al.(1997)* properties of a coherent risk measure include subadditivity (risks of the portfolio should not exceed the sum of the risks of individual components) as a significantly desirable trait. VaR measures, including Marginal VaR, on individual components of a portfolio are *not* subadditive.

Clearly, a general subadditive risk measure for downside risk is required. In Incremental or Component VaR, the Component VaR value for each element of the portfolio will sum to the total VaR of the portfolio. Several EDHEC papers suggest using Modified VaR instead of mean-VaR in the Incremental and Component VaR calculation. We have succeeded in implementing Component VaR and ES calculations that use Modified Cornish-Fisher VaR, historical decomposition, and kernel estimators. You may access these with `VaR` or `ES` by setting the appropriate `portfolio_method` and `method` arguments.

The `chart.VaRSensitivity` function creates a chart of Value-at-Risk or Expected Shortfall estimates by confidence interval for multiple methods. Useful for comparing a calculated VaR or ES method to the historical VaR or ES, it may also be used to visually examine whether the VaR method "breaks down" or gives nonsense results at a certain threshold.

Which VaR or ES measure to use will depend greatly on the portfolio and instruments being analyzed. If there is any generalization to be made on VaR measures, we agree with *Bali and Gokcan(2004)* who conclude that "the VaR estimations based on the generalized Pareto distribution and the Cornish-Fisher approximation perform best".

Most risk professionals are moving from using VaR to using Expected Shortfall preferentially. This preference has been endorsed by the Bank of International Settlements (BIS) and the Basel accord risk committee. In most cases, an ES measure with an appropriate probability target for your asset horizin will be more appropriate than a VaR measure.

Performance Analysis

The literature around the subject of performance analysis seems to have exploded with the popularity of alternative assets such as hedge funds, managed futures, commodities, and structured products. Simpler tools that may have seemed appropriate in a relative investment world seem inappropriate for an absolute return world. Risk measurement, which is nearly inseparable from performance assessment, has become multi-dimensional and multi-moment while trying to answer a simple question: "How much could I lose?" Portfolio construction and risk budgeting are two sides of the same coin: "How do I maximize my expected gain and avoid going broke?" But before we can approach those questions we first have to ask: "Is this something I might want in my portfolio?"

With the the increasing availability of complicated alternative investment strategies to both retail and institutional investors, and the broad availability of financial data, an engaging debate about performance analysis and evaluation is as important as ever. There won't be one *right* answer delivered in these metrics and charts. What there will be is an accretion of evidence, organized to *assist* a decision maker in answering a specific question that is pertinent to the decision at hand. Using such tools to uncover information and ask better questions will, in turn, create a more informed investor.

Performance measurement starts with returns. Traders may object, complaining that "You can't eat returns," and will prefer to look for numbers with currency signs. To some extent, they have a point - the normalization inherent in calculating returns can be deceiving. Most of the recent work in performance analysis, however, is focused on returns rather than prices and sometimes called "returns-based analysis" or RBA. This "price per unit of investment" standardization is important for two reasons - first, it helps the decision maker to compare opportunities, and second, it has some useful statistical qualities. As a result, the `PerformanceAnalytics` package focuses on returns. See `Return.calculate` for converting net asset values or prices into returns, either discrete or continuous. Many papers and theories refer to "excess returns": we implement a simple function for aligning time series and calculating excess returns in `Return.excess`.

Return.portfolio can be used to calculate weighted returns for a portfolio of assets. The function was recently changed to support several use-cases: a single weighting vector, an equal weighted portfolio, periodic rebalancing, or irregular rebalancing. That replaces functionality that had been split between that function and Return.rebalancing. The function will subset the return series to only include returns for assets for which weights are provided.

Returns and risk may be annualized as a way to simplify comparison over longer time periods. Although it requires a bit of estimating, such aggregation is popular because it offers a reference point for easy comparison. Examples are in Return.annualized, sd.annualized, and SharpeRatio.annualized.

Basic measures of performance tend to treat returns as independent observations. In this case, the entirety of R's base is applicable to such analysis. Some basic statistics we have collected in table.Stats include:

mean	arithmetic mean
mean.geometric	geometric mean
mean.stderr	standard error of the mean (S.E. mean)
mean.LCL	lower confidence level (LCL) of the mean
mean.UCL	upper confidence level (UCL) of the mean
quantile	for calculating various quantiles of the distribution
min	minimum return
max	maximum return
range	range of returns
length(R)	number of observations
sum(is.na(R))	number of NA's

It is often valuable when evaluating an investment to know whether the instrument that you are examining follows a normal distribution. One of the first methods to determine how close the asset is to a normal or log-normal distribution is to visually look at your data. Both chart.QQPlot and chart.Histogram will quickly give you a feel for whether or not you are looking at a normally distributed return history. Differences between var and SemiVariance will help you identify skewness in the returns. Skewness measures the degree of asymmetry in the return distribution. Positive skewness indicates that more of the returns are positive, negative skewness indicates that more of the returns are negative. An investor should in most cases prefer a positively skewed asset to a similar (style, industry, region) asset that has a negative skewness.

Kurtosis measures the concentration of the returns in any given part of the distribution (as you should see visually in a histogram). The kurtosis function will by default return what is referred to as "excess kurtosis", where 0 is a normal distribution, other methods of calculating kurtosis than method="excess" will set the normal distribution at a value of 3. In general a rational investor should prefer an asset with a low to negative excess kurtosis, as this will indicate more predictable returns (the major exception is generally a combination of high positive skewness and high excess kurtosis). If you find yourself needing to analyze the distribution of complex or non-smooth asset distributions, the nortest package has several advanced statistical tests for analyzing the normality of a distribution.

Modern Portfolio Theory (MPT) is the collection of tools and techniques by which a risk-averse investor may construct an "optimal" portfolio. It was pioneered by Markowitz's ground-breaking

1952 paper *Portfolio Selection*. It also encompasses CAPM, below, the efficient market hypothesis, and all forms of quantitative portfolio construction and optimization.

The Capital Asset Pricing Model (CAPM), initially developed by William Sharpe in 1964, provides a justification for passive or index investing by positing that assets that are not on the efficient frontier will either rise or fall in price until they are. The `CAPM.RiskPremium` is the measure of how much the asset's performance differs from the risk free rate. Negative Risk Premium generally indicates that the investment is a bad investment, and the money should be allocated to the risk free asset or to a different asset with a higher risk premium. `CAPM.alpha` is the degree to which the assets returns are not due to the return that could be captured from the market. Conversely, `CAPM.beta` describes the portions of the returns of the asset that could be directly attributed to the returns of a passive investment in the benchmark asset.

The Capital Market Line `CAPM.CML` relates the excess expected return on an efficient market portfolio to its risk (represented in CAPM by `sd`). The slope of the CML, `CAPM.CML.slope`, is the Sharpe Ratio for the market portfolio. The Security Market Line is constructed by calculating the line of `CAPM.RiskPremium` over `CAPM.beta`. For the benchmark asset this will be 1 over the risk premium of the benchmark asset. The slope of the SML, primarily for plotting purposes, is given by `CAPM.SML.slope`. CAPM is a market equilibrium model or a general equilibrium theory of the relation of prices to risk, but it is usually applied to partial equilibrium portfolios, which can create (sometimes serious) problems in valuation.

One extension to the CAPM contemplates evaluating an active manager's ability to time the market. Two other functions apply the same notion of best fit to positive and negative market returns, separately. The `CAPM.beta.bull` is a regression for only positive market returns, which can be used to understand the behavior of the asset or portfolio in positive or 'bull' markets. Alternatively, `CAPM.beta.bear` provides the calculation on negative market returns. The `TimingRatio` uses the ratio of those to help assess whether the manager has shown evidence that of timing skill.

Performance/Risk Ratios:

In many cases, an analyst will be looking to find a measure or performance relative to the risk of the asset under study. PerformanceAnalytics has many functions of this type.

One of the most commonly used and cited measures of the risk/reward tradeoff of an investment or portfolio is the `SharpeRatio`, which measures return over standard deviation. If you are comparing multiple assets using Sharpe, you should use `SharpeRatio.annualized`. It is important to note that William Sharpe now recommends `InformationRatio` preferentially to the original Sharpe Ratio. The `SortinoRatio` uses mean return over `DownsideDeviation` below the MAR as the risk measure to produce a similar ratio that is more sensitive to downside risk. Sortino later enhanced his ideas to use upside returns for the numerator and `DownsideDeviation` as the denominator in `UpsidePotentialRatio`. *Favre and Galeano(2002)* propose using the ratio of expected excess return over the Cornish-Fisher `VaR` to produce `SharpeRatio.modified`. `TreynorRatio` is also similar to the Sharpe Ratio, except it uses `CAPM.beta` in place of the volatility measure to produce the ratio of the investment's excess return over the beta. Use of the downside semivariance as the denominator creates the . Use of the upside expected tail return over the ETL creates the `RachevRatio`. Utilizing the downside semivariance as the denominator produces the Downside Sharpe Ratio.

The performance premium provided by an investment over a passive strategy (the benchmark) is provided by `ActivePremium`, which is the investment's annualized return minus the benchmark's annualized return. A closely related measure is the `TrackingError`, which measures the unexplained portion of the investment's performance relative to a benchmark. The `InformationRatio`

of an investment in a MPT or CAPM framework is the Active Premium divided by the Tracking Error. Information Ratio may be used to rank investments in a relative fashion.

We have also included a function to compute the KellyRatio. The Kelly criterion applied to position sizing will maximize log-utility of returns and avoid risk of ruin. For our purposes, it can also be used as a stack-ranking method like InformationRatio to describe the "edge" an investment would have over a random strategy or distribution.

These metrics and others such as SharpeRatio, SortinoRatio, UpsidePotentialRatio, Spearman rank correlation (see rcorr), etc., are all methods of rank-ordering relative performance. *Alexander and Dimitriu (2004) in "The Art of Investing in Hedge Funds"* show that relative rankings across multiple pricing methodologies may be positively correlated with each other and with expected returns. This is quite an important finding because it shows that multiple methods of predicting returns and risk which have underlying measures and factors that are not directly correlated to another measure or factor will still produce widely similar quantile rankings, so that the "buckets" of target instruments will have significant overlap. This observation specifically supports the point made early in this document regarding "accretion of the evidence" for a positive or negative investment decision.

Standard Errors for Risk and Performance Estimators

While PerformanceAnalytics contains many functions for computing returns based risk and performance estimators, until now there has been no convenient way to accurately compute standard errors of the estimates for independent and identically distributed (i.i.d.) returns, and no way at all to do so for returns that are serially correlated. This is no longer the case due to the existence of a new frequency domain method of accurately computing standard errors when returns are serially dependent as well as when returns are independent and identically distributed. Details are provided in Xin and Martin (2019) at https://www.ssrn.com/abstract=3085672, and to appear in December 2020 issue of Journal of Risk. The new method makes novel use of statistical influence functions borrowed from robust statistics, combined with periodogram based regularized generalized linear polynomial model (GLM) fitting for exponential distributions. Influence function for risk and performance estimators are described in Zhang, Martin and Christidis (2020) available at SSRN https://www.ssrn.com/abstract=3415903. The new method has been implemented in the RPESE package available at CRAN. The RPESE package has been integrated into PerformanceAnalytics.

Standard errors can be easily computed using the PerformanceAnalytics risk estimator functions

StdDev	Sample standard deviation
SemiSD	Semi-standard deviation
lpm with argument n=1 or n=2	Lower partial moment of order 1 or 2
ES	Expected shortfall with tail probability α
VaR	Value-at-risk with tail probability α

and performance estimator functions

mean.arithmetic	Sample mean
SharpeRatio with argument FUN="StdDev"	Sharpe ratio

DownsideSharpeRatio or SharpeRatio with argument FUN="SemiSD"	Downside Sharpe ratio
SortinoRatio	Sortino ratio with threshold MAR
SharpeRatio with argument FUN="ES"	Mean excess return to ES ratio with tail probability
SharpeRatio with argument FUN="VaR"	Mean excess return to VaR ratio with tail probabilit
RachevRatio	Rachev ratio with lower upper tail probabilities α a
Omega	Omega ratio with threshold c

where the first columns gives the names of the functions in the PerformanceAnalytics package to compute the estimate and its standard error.

Each of the PerformanceAnalytics functions listed above have an optional argument with default SE = FALSE. By changing this default to SE = TRUE, the user obtains not only risk or performance estimate, but also a standard error for the esimate. Further details concerning the computation of standard errors for the risk and performance estimators in PerformanceAnalytics can be found in the Vignette "Standard Errors for Risk and Performance Estimators in PerformanceAnalytics" available at CRAN, where a reference to the underlying theory due to Chen and Martin (2020) may be found.

Moments and Co-moments

Analysis of financial time series often involves evaluating their mathematical moments. While var and cov for variance has always been available, as well as skewness and kurtosis (which we have extended to make multivariate and multi-column aware), a larger suite of multivariate moments calculations was not available in R. We have now implemented multivariate moments and co-moments and their beta or systematic co-moments in PerformanceAnalytics.

Ranaldo and Favre (2005) define coskewness and cokurtosis as the skewness and kurtosis of a given asset analysed with the skewness and kurtosis of the reference asset or portfolio. The co-moments are useful for measuring the marginal contribution of each asset to the portfolio's resulting risk. As such, co-moments of an asset return distribution should be useful as inputs for portfolio optimization in addition to the covariance matrix. Functions include CoVariance, CoSkewness, CoKurtosis.

Measuring the co-moments should be useful for evaluating whether or not an asset is likely to provide diversification potential to a portfolio. But the co-moments do not allow the marginal impact of an asset on a portfolio to be directly measured. Instead, *Martellini and Zieman (2007)* develop a framework that assesses the potential diversification of an asset relative to a portfolio. They use higher moment betas to estimate how much portfolio risk will be impacted by adding an asset.

Higher moment betas are defined as proportional to the derivative of the covariance, coskewness and cokurtosis of the second, third and fourth portfolio moment with respect to the portfolio weights. A beta that is less than 1 indicates that adding the new asset should reduce the resulting portfolio's volatility and kurtosis, and to an increase in skewness. More specifically, the lower the beta the higher the diversification effect, not only in terms of normal risk (i.e. volatility) but also the risk of assymetry (skewness) and extreme events (kurtosis). See the functions for BetaCoVariance, BetaCoSkewness, and BetaCoKurtosis.

Robust Data Cleaning

The functions Return.clean and clean.boudt implement statistically robust data cleaning methods tuned to portfolio construction and risk analysis and prediction in financial time series while

trying to avoid some of the pitfalls of standard robust statistical methods.

The primary value of data cleaning lies in creating a more robust and stable estimation of the distribution generating the large majority of the return data. The increased robustness and stability of the estimated moments using cleaned data should be used for portfolio construction. If an investor wishes to have a more conservative risk estimate, cleaning may not be indicated for risk monitoring.

In actual practice, it is probably best to back-test the out-of-sample results of both cleaned and uncleaned series to see what works best when forecasting risk with the particular combination of assets under consideration.

Summary Tabular Data

Summary statistics are then the necessary aggregation and reduction of (potentially thousands) of periodic return numbers. Usually these statistics are most palatable when organized into a table of related statistics, assembled for a particular purpose. A common offering of past returns organized by month and cumulated by calendar year is usually presented as a table, such as in `table.CalendarReturns`. Adding benchmarks or peers alongside the annualized data is helpful for comparing returns in calendar years.

When we started this project, we debated whether such tables would be broadly useful or not. No reader is likely to think that we captured the precise statistics to help their decision. We merely offer these as a starting point for creating your own. Add, subtract, do whatever seems useful to you. If you think that your work may be useful to others, please consider sharing it so that we may include it in a future version of this package.

Other tables for comparison of related groupings of statistics discussed elsewhere:

`table.Stats`	Basic statistics and stylized facts
`table.TrailingPeriods`	Statistics and stylized facts compared over different trailing periods
`table.AnnualizedReturns`	Annualized return, standard deviation, and Sharpe ratio
`table.CalendarReturns`	Monthly and calendar year return table
`table.CAPM`	CAPM-related measures
`table.Correlation`	Comparison of correlalations and significance statistics
`table.DownsideRisk`	Downside risk metrics and statistics
`table.Drawdowns`	Ordered list of drawdowns and when they occurred
`table.Autocorrelation`	The first six autocorrelation coefficients and significance
`table.HigherMoments`	Higher co-moments and beta co-moments
`table.Arbitrary`	Combines a function list into a table

Charts and Graphs

Graphs and charts can also help to organize the information visually. Our goal in creating these charts was to simplify the process of creating well-formatted charts that are used often in performance analysis, and to create high-quality graphics that may be used in documents for consumption by non-analysts or researchers. R's graphics capabilities are substantial, but the simplicity of the output of R default graphics functions such as `plot` does not always compare well against graphics delivered with commercial asset or performance analysis from places such as MorningStar or PerTrac.

The cumulative returns or wealth index is usually the first thing displayed, even though neither conveys much information. See `chart.CumReturns`. Individual period returns may be helpful for

identifying problematic periods, such as in chart.Bar. Risk measures can be helpful when overlaid on the period returns, to display the bounds at which losses may be expected. See chart.BarVaR and the prior section on Risk Analysis. More information can be conveyed when such charts are displayed together, as in charts.PerformanceSummary, which combines the performance data with detail on downside risk (see chart.Drawdown).

chart.RelativePerformance can plot the relative performance through time of two assets. This plot displays the ratio of the cumulative performance at each point in time and makes periods of under- or out-performance easy to see. The value of the chart is less important than the slope of the line. If the slope is positive, the first asset is outperforming the second, and vice verse. Affectionately known as the Canto chart, it was used effectively in Canto (2006).

Two-dimensional charts can also be useful while remaining easy to understand. chart.Scatter is a utility scatter chart with some additional attributes that are used in chart.RiskReturnScatter. Overlaying Sharpe ratio lines or boxplots helps to add information about relative performance along those dimensions.

For distributional analysis, a few graphics may be useful. chart.Boxplot is an example of a graphic that is difficult to create in Excel and is under-utilized as a result. A boxplot of returns is, however, a very useful way to instantly observe the shape of large collections of asset returns in a manner that makes them easy to compare to one another. chart.Histogram and chart.QQPlot are two charts originally found elsewhere and now substantially expanded in PerformanceAnalytics.

Rolling performance is typically used as a way to assess stability of a return stream. Although perhaps it doesn't get much credence in the financial literature as it derives from work in digital signal processing, many practitioners find it a useful way to examine and segment performance and risk periods. See chart.RollingPerformance, which is a way to display different metrics over rolling time periods. chart.RollingMean is a specific example of a rolling mean and standard error bands. A group of related metrics is offered in charts.RollingPerformance. These charts use utility functions such as rollapply.

chart.SnailTrail is a scatter chart that shows how rolling calculations of annualized return and annualized standard deviation have proceeded through time where the color of lines and dots on the chart diminishes with respect to time. chart.RollingCorrelation shows how correlations change over rolling periods. chart.RollingRegression displays the coefficients of a linear model fitted over rolling periods. A group of charts in charts.RollingRegression displays alpha, beta, and R-squared estimates in three aligned charts in a single device.

chart.StackedBar creates a stacked column chart with time on the horizontal axis and values in categories. This kind of chart is commonly used for showing portfolio 'weights' through time, although the function will plot any values by category.

We have been greatly inspired by other peoples' work, some of which is on display at gallery.r-enthusiasts.com. Particular inspiration came from Dirk Eddelbuettel and John Bollinger for their work at http://gallery.r-enthusiasts.com/graph/65. Those interested in price charting in R should also look at the quantmod package.

Wrapper and Utility Functions

R is a very powerful environment for manipulating data. It can also be quite confusing to a user more accustomed to Excel or even MatLAB. As such, we have written some wrapper functions that may aid you in coercing data into the correct forms or finding data that you need to use regularly. To simplify the management of multiple-source data stored in R in multiple data formats, we have provided checkData. This function will attempt to coerce data in and out of R's multitude of mostly

fungible data classes into the class required for a particular analysis. The data-coercion function has been hidden inside the functions here, but it may also save you time and trouble in your own code and functions as well.

R's built-in `apply` function in enormously powerful, but is can be tricky to use with timeseries data, so we have provided wrapper functions to `apply.fromstart` and `apply.rolling` to make handling of "from inception" and "rolling window" calculations easier.

Further Work

We have attempted to standardize function parameters and variable names, but more work exists to be done here.

Any comments, suggestions, or code patches are invited.

If you've implemented anything that you think would be generally useful to include, please consider donating it for inclusion in a later version of this package.

Acknowledgments

Data series `edhec` used in `PerformanceAnalytics` and related publications with the kind permission of the EDHEC Risk and Asset Management Research Center.
https://climateinstitute.edhec.edu/retirement-investing

Kris Boudt was instrumental in our research on component risk for portfolios with non-normal distributions, and is responsible for much of the code for multivariate moments and co-moments. This works was later extended and made faster and more robust by Dries Cornilly

Jeff Ryan and Joshua Ulrich are active participants in the R finance community and created `xts`, upon which much of PerformanceAnalytics depends.

Prototypes of the drawdowns functionality were provided by Sankalp Upadhyay, and modified with permission. Stephan Albrecht provided detailed feedback on the Getmansky/Lo Smoothing Index. The late Diethelm Wuertz provided prototypes of modified VaR and skewness and kurtosis functions (and was of course the maintainer of the RMetrics suite of pricing and optimization functions). Diethelm also contributed prototypes for many other functions from Bacon's book that were incorporated into PerformanceAnalytics by Matthieu Lestel.

Thanks to Joe Wayne Byers and Dirk Eddelbuettel for comments on early versions of these functions, and to Khanh Nguyen, Tobias Verbeke, H. Felix Wittmann, and Ryan Sheftel for careful testing and detailed problem reports.

Thanks also to our Google Summer of Code students through the years for their contributions. Significant contributions from GSOC students to this package have come from Dries Cornilly, Anthony-Alexander Cristidis, Zenith "Ziheng" Zhou, Matthieu Lestel and Andrii Babii so far. We expect to eventually incorporate contributions from Pulkit Mehrotra and Shubhankit Mohan, who worked with us during the summer of 2013.

Thanks to the R-SIG-Finance community without whom this package would not be possible. We are indebted to the R-SIG-Finance community for many helpful suggestions, bugfixes, and requests.

Any errors are, of course, our own.

Author(s)

Maintainer: Brian G. Peterson <brian@braverock.com> [copyright holder]

Authors:

- Peter Carl <peter@braverock.com> [copyright holder]

Other contributors:

- Kris Boudt [contributor, copyright holder]
- Ross Bennett [contributor]
- Joshua Ulrich [contributor]
- Eric Zivot [contributor]
- Dries Cornilly [contributor]
- Eric Hung [contributor]
- Matthieu Lestel [contributor]
- Kyle Balkissoon [contributor]
- Diethelm Wuertz [contributor]
- Anthony Alexander Christidis [contributor]
- R. Douglas Martin [contributor]
- Zeheng Zenith Zhou [contributor]
- Justin M. Shea [contributor]
- Dhairya Jain [contributor]

References

Amenc, N. and Le Sourd, V. *Portfolio Theory and Performance Analysis*. Wiley. 2003.

Pfaff, B. *Financial Risk Modeling and Portfolio Optimization with R, Second Edition*. Wiley. 2016.

Bacon, C. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004.

Canto, V. *Understanding Asset Allocation*. FT Prentice Hall. 2006.

Chen, X. and Martin, R. D. *Standard Errors of Risk and Performance Measure Estimators for Serially Correlated Returns*. SSRN eLibrary. 2019.

Lhabitant, F. *Hedge Funds: Quantitative Insights*. Wiley. 2004.

Litterman, R., Gumerlock R., et. al. *The Practice of Risk Management: Implementing Processes for Managing Firm-Wide Market Risk*. Euromoney. 1998.

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

Martin, R. Douglas, and Arora, Rohit. *Inefficiency and bias of modified value-at-risk and expected shortfall*. 2017. Journal of Risk 19(6), 59–84

Ranaldo, A., and Laurent Favre Sr. *How to Price Hedge Funds: From Two- to Four-Moment CAPM.* SSRN eLibrary. 2005.

Murrel, P. *R Graphics*. Chapman and Hall. 2006.

Ruppert, D., and Matteson, D *Statistics and Data Analysis for Financial Engineering, with R Examples. Second Edition*. Springer. 2015.

Scherer, B. and Martin, D. *Modern Portfolio Optimization*. Springer. 2005.

Shumway, R. and Stoffer, D. *Time Series Analysis and it's Applications, with R examples*, Springer, 2006.

Tsay, R. *Analysis of Financial Time Series*. Wiley. 2001.

Chen, X. and Martin, R. D. *Standard Errors of Risk and Performance Measure Estimators for Serially Correlated Returns*. SSRN eLibrary. 2019.

Zhang, S., Martin, R. Douglas., and Christidis, A. *Influence Functions for Risk and Performance Estimators*. SSRN eLibrary. 2020.

Zin, M., Zhao. A Note on Semivariance. Mathematical Finance, Vol. 16, No. 1, pp. 53-61, January 2006

Zivot, E. and Wang, Z. *Modeling Financial Time Series with S-Plus: second edition*. Springer. 2006.

See Also

CRAN task view on Empirical Finance
https://CRAN.R-project.org/view=Econometrics

Grant Farnsworth's Econometrics in R
https://cran.r-project.org/doc/contrib/Farnsworth-EconometricsInR.pdf

.coefficients *Wrapper for SFM's regression models.*

Description

This is intended to be called using SFM.coefficients function, and not directly.

Usage

```
.coefficients(xRa, xRb, subset, ..., method = "LS", family = "mopt")
```

Arguments

xRa	an xts, vector, matrix, data frame, timeSeries or zoo object of excess asset returns
xRb	excess return vector of the benchmark asset
subset	a logical vector
...	arguments passed to other methods
method	Which linear model to use for SFM regression
family	If method is "Rob", then this is a string specifying the name of the family of loss function to be used (current valid options are "bisquare", "opt" and "mopt"). Incomplete entries will be matched to the current valid options. Defaults to "mopt".

Author(s)

Dhairya Jain

ActiveReturn *Active Premium or Active Return*

Description

The return on an investment's annualized return minus the benchmark's annualized return.

Usage

```
ActiveReturn(Ra, Rb, scale = NA, ...)
```

Arguments

Ra	return vector of the portfolio
Rb	return vector of the benchmark asset
scale	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
...	any other passthru parameters to Return.annualized (e.g., geometric=FALSE)

Details

Active Premium = Investment's annualized return - Benchmark's annualized return

Also commonly referred to as 'active return'.

Author(s)

Peter Carl

References

Sharpe, W.F. The Sharpe Ratio,*Journal of Portfolio Management*, Fall 1994, 49-58.

See Also

`InformationRatio` `TrackingError` `Return.annualized`

Examples

```
data(managers)
ActivePremium(managers[, "HAM1", drop=FALSE], managers[, "SP500 TR", drop=FALSE])
ActivePremium(managers[,1,drop=FALSE], managers[,8,drop=FALSE])
ActivePremium(managers[,1:6], managers[,8,drop=FALSE])
ActivePremium(managers[,1:6], managers[,8:7,drop=FALSE])
```

AdjustedSharpeRatio *Adjusted Sharpe ratio of the return distribution*

Description

Adjusted Sharpe ratio was introduced by Pezier and White (2006) to adjusts for skewness and kurtosis by incorporating a penalty factor for negative skewness and excess kurtosis.

Usage

```
AdjustedSharpeRatio(R, Rf = 0, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rf	the risk free rate
...	any other passthru parameters

Details

$$AdjustedSharpeRatio = SR * [1 + (\frac{S}{6}) * SR - (\frac{K-3}{24}) * SR^2]$$

where SR is the sharpe ratio with data annualized, S is the skewness and K is the kurtosis

Author(s)

Matthieu Lestel, Brian G. Peterson

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.99

Pezier, Jaques and White, Anthony. 2006. The Relative Merits of Investable Hedge Fund Indices and of Funds of Hedge Funds in Optimal Passive Portfolios. https://econpapers.repec.org/paper/rdgicmadp/icma-dp2006-10.htm

See Also

SharpeRatio.annualized

Examples

```
data(portfolio_bacon)
print(AdjustedSharpeRatio(portfolio_bacon[,1])) #expected 0.7591435

data(managers)
print(AdjustedSharpeRatio(managers['1996']))
```

---

apply.fromstart — *calculate a function over an expanding window always starting from the beginning of the series*

Description

A function to calculate a function over an expanding window from the start of the timeseries. This wrapper allows easy calculation of "from inception" statistics.

Usage

```
apply.fromstart(R, FUN = "mean", gap = 1, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
FUN	any function that can be evaluated using a single set of returns (e.g., rolling beta won't work, but Return.annualized will)
gap	the number of data points from the beginning of the series required to "train" the calculation
...	any other passthru parameters

Author(s)

Peter Carl

See Also

rollapply

Examples

```
data(managers)
apply.fromstart(managers[,1,drop=FALSE], FUN="mean", width=36)
```

---

apply.rolling *calculate a function over a rolling window*

Description

Creates a results timeseries of a function applied over a rolling window.

Usage

```
apply.rolling(R, width, trim = TRUE, gap = 12, by = 1, FUN = "mean", ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
width	number of periods to apply rolling function window over
trim	TRUE/FALSE, whether to keep alignment caused by NA's
gap	numeric number of periods from start of series to use to train risk calculation
by	calculate FUN for trailing width points at every by-th time point.
FUN	any function that can be evaluated using a single set of returns (e.g., rolling beta won't work, but Return.annualized will)
...	any other passthru parameters

Details

Wrapper function for rollapply to hide some of the complexity of managing single-column zoo objects.

Value

A timeseries in a zoo object of the calculation results

Author(s)

Peter Carl

See Also

apply
rollapply

Examples

```
data(managers)
apply.rolling(managers[,1,drop=FALSE], FUN="mean", width=36)
```

AppraisalRatio *Appraisal ratio of the return distribution*

Description

Appraisal ratio is the Jensen's alpha adjusted for specific risk. The numerator is divided by specific risk instead of total risk.

Usage

```
AppraisalRatio(
  Ra,
  Rb,
  Rf = 0,
  method = c("appraisal", "modified", "alternative"),
  ...
)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
Rf	risk free rate, in same period as your returns
method	is one of "appraisal" to calculate appraisal ratio, "modified" to calculate modified Jensen's alpha or "alternative" to calculate alternative Jensen's alpha.
...	any other passthru parameters

Details

Modified Jensen's alpha is Jensen's alpha divided by beta.

Alternative Jensen's alpha is Jensen's alpha divided by systematic risk.

$$Appraisalratio = \frac{\alpha}{\sigma_\epsilon}$$

$$ModifiedJensen'salpha = \frac{\alpha}{\beta}$$

$$AlternativeJensen's alpha = \frac{\alpha}{\sigma_S}$$

where $alpha$ is the Jensen's alpha, $\sigma_{epsilon}$ is the specific risk, σ_S is the systematic risk.

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.77

Examples

```
data(portfolio_bacon)
print(AppraisalRatio(portfolio_bacon[,1], portfolio_bacon[,2], method="appraisal")) #expected -0.430
print(AppraisalRatio(portfolio_bacon[,1], portfolio_bacon[,2], method="modified"))
print(AppraisalRatio(portfolio_bacon[,1], portfolio_bacon[,2], method="alternative"))

data(managers)
print(AppraisalRatio(managers['1996',1], managers['1996',8]))
print(AppraisalRatio(managers['1996',1:5], managers['1996',8]))
```

`AverageDrawdown` *Calculates the average depth of the observed drawdowns.*

Description

ADD = abs(sum[j=1,2,...,d](D_j/d)) where D'_j = jth drawdown over entire period d = total number of drawdowns in the entire period

Usage

```
AverageDrawdown(R, ...)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`...`	any other passthru parameters

Author(s)

Peter Carl

AverageLength — *Calculates the average length (in periods) of the observed drawdowns.*

Description

Similar to AverageDrawdown, which calculates the average depth of drawdown, this function calculates the average length of the drawdowns observed.

Usage

```
AverageLength(R, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters

Author(s)

Peter Carl

AverageRecovery — *Calculates the average length (in periods) of the observed recovery period.*

Description

Similar to AverageDrawdown, which calculates the average depth of drawdown, this function calculates the average length of the recovery period of the drawdowns observed.

Usage

```
AverageRecovery(R, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters

Author(s)

Peter Carl

`BernardoLedoitRatio` *Bernardo and Ledoit ratio of the return distribution*

Description

To calculate Bernardo and Ledoit ratio we take the sum of the subset of returns that are above 0 and we divide it by the opposite of the sum of the subset of returns that are below 0

Usage

```
BernardoLedoitRatio(R, ...)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`...`	any other passthru parameters

Details

$$BernardoLedoitRatio(R) = \frac{\frac{1}{n}\sum_{t=1}^{n} max(R_t, 0)}{\frac{1}{n}\sum_{t=1}^{n} max(-R_t, 0)}$$

where n is the number of observations of the entire series

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.95

Examples

```
data(portfolio_bacon)
print(BernardoLedoitRatio(portfolio_bacon[,1])) #expected 1.78

data(managers)
print(BernardoLedoitRatio(managers['1996']))
print(BernardoLedoitRatio(managers['1996',1])) #expected 4.598
```

`BetaCoMoments` *Functions to calculate systematic or beta co-moments of return series*

Description

calculate higher co-moment betas, or 'systematic' variance, skewness, and kurtosis

Usage

```
BetaCoVariance(Ra, Rb)

BetaCoSkewness(Ra, Rb, test = FALSE)

BetaCoKurtosis(Ra, Rb)
```

Arguments

`Ra`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rb`	an xts, vector, matrix, data frame, timeSeries or zoo object of index, benchmark, or secondary asset returns to compare against
`test`	condition not implemented yet

Details

The co-moments, including covariance, coskewness, and cokurtosis, do not allow the marginal impact of an asset on a portfolio to be directly measured. Instead, Martellini and Zieman (2007) develop a framework that assesses the potential diversification of an asset relative to a portfolio. They use higher moment betas to estimate how much portfolio risk will be impacted by adding an asset, in terms of symmetric risk (i.e., volatility), in asymmetry risk (i.e., skewness), and extreme risks (i.e. kurtosis). That allows them to show that adding an asset to a portfolio (or benchmark) will reduce the portfolio's variance to be reduced if the second-order beta of the asset with respect to the portfolio is less than one. They develop the same concepts for the third and fourth order moments. The authors offer these higher moment betas as a measure of the diversification potential of an asset.

Higher moment betas are defined as proportional to the derivative of the covariance, coskewness and cokurtosis of the second, third and fourth portfolio moment with respect to the portfolio weights. The beta co-variance is calculated as:

$$BetaCoV(Ra, Rb) = \beta_{a,b}^{(2)} = \frac{CoV(R_a, R_b)}{\mu^{(2)}(R_b)}$$

Beta co-skewness is given as:

$$BetaCoS(Ra, Rb) = \beta_{a,b}^{(3)} = \frac{CoS(R_a, R_b)}{\mu^{(3)}(R_b)}$$

Beta co-kurtosis is:

$$BetaCoK(Ra, Rb) = \beta_{a,b}^{(4)} = \frac{CoK(R_a, R_b)}{\mu^{(4)}(R_b)}$$

where the n-th centered moment is calculated as

$$\mu^{(n)}(R) = E[(R - E(R))^n]$$

A beta is greater than one indicates that no diversification benefits should be expected from the introduction of that asset into the portfolio. Conversely, a beta that is less than one indicates that adding the new asset should reduce the resulting portfolio's volatility and kurtosis, and to an increase in skewness. More specifically, the lower the beta the higher the diversification effect on normal risk (i.e. volatility). Similarly, since extreme risks are generally characterised by negative skewness and positive kurtosis, the lower the beta, the higher the diversification effect on extreme risks (as reflected in Modified Value-at-Risk or ER).

The addition of a small fraction of a new asset to a portfolio leads to a decrease in the portfolio's second moment (respectively, an increase in the portfolio's third moment and a decrease in the portfolio's fourth moment) if and only if the second moment (respectively, the third moment and fourth moment) beta is less than one (see Martellini and Ziemann (2007) for more details).

For skewness, the interpretation is slightly more involved. If the skewness of the portfolio is negative, we would expect an increase in portfolio skewness when the third moment beta is lower than one. When the skewness of the portfolio is positive, then the condition is that the third moment beta is greater than, as opposed to lower than, one.

Because the interpretation of beta coskewness is made difficult by the need to condition on it's skewness, we deviate from the more widely used measure slightly. To make the interpretation consistent across all three measures, the beta coskewness function tests the skewness and multiplies the result by the sign of the skewness. That allows an analyst to review the metric and interpret it without needing additional information. To use the more widely used metric, simply set the parameter `test = FALSE`.

Author(s)

Kris Boudt, Peter Carl, Brian Peterson

References

Boudt, Kris, Brian G. Peterson, and Christophe Croux. 2008. Estimation and Decomposition of Downside Risk for Portfolios with Non-Normal Returns. Journal of Risk. Winter.

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

See Also

CoMoments

Examples

```
data(managers)

BetaCoVariance(managers[, "HAM2", drop=FALSE], managers[, "SP500 TR", drop=FALSE])
BetaCoSkewness(managers[, "HAM2", drop=FALSE], managers[, "SP500 TR", drop=FALSE])
BetaCoKurtosis(managers[, "HAM2", drop=FALSE], managers[, "SP500 TR", drop=FALSE])
BetaCoKurtosis(managers[,1:6], managers[,8,drop=FALSE])
BetaCoKurtosis(managers[,1:6], managers[,8:7])
```

BurkeRatio *Burke ratio of the return distribution*

Description

To calculate Burke ratio we take the difference between the portfolio return and the risk free rate and we divide it by the square root of the sum of the square of the drawdowns. To calculate the modified Burke ratio we just multiply the Burke ratio by the square root of the number of datas.

Usage

```
BurkeRatio(R, Rf = 0, modified = FALSE, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rf	the risk free rate
modified	a boolean to decide which ratio to calculate between Burke ratio and modified Burke ratio.
...	any other passthru parameters

Details

$$BurkeRatio = \frac{r_P - r_F}{\sqrt{\sum_{t=1}^{d} {D_t}^2}}$$

$$ModifiedBurkeRatio = \frac{r_P - r_F}{\sqrt{\sum_{t=1}^{d} \frac{{D_t}^2}{n}}}$$

where n is the number of observations of the entire series, d is number of drawdowns, r_P is the portfolio return, r_F is the risk free rate and D_t the t^{th} drawdown.

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.90-91

Examples

```
data(portfolio_bacon)
print(BurkeRatio(portfolio_bacon[,1])) #expected 0.74
print(BurkeRatio(portfolio_bacon[,1], modified = TRUE)) #expected 3.65

data(managers)
print(BurkeRatio(managers['1996']))
print(BurkeRatio(managers['1996',1]))
print(BurkeRatio(managers['1996'], modified = TRUE))
print(BurkeRatio(managers['1996',1], modified = TRUE))
```

`CalmarRatio` *calculate a Calmar or Sterling reward/risk ratio Calmar and Sterling Ratios are yet another method of creating a risk-adjusted measure for ranking investments similar to the* `SharpeRatio`.

Description

Both the Calmar and the Sterling ratio are the ratio of annualized return over the absolute value of the maximum drawdown of an investment. The Sterling ratio adds an excess risk measure to the maximum drawdown, traditionally and defaulting to 10%.

Usage

```
CalmarRatio(R, scale = NA)

SterlingRatio(R, scale = NA, excess = 0.1)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`scale`	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
`excess`	for Sterling Ratio, excess amount to add to the max drawdown, traditionally and default .1 (10%)

Details

It is also traditional to use a three year return series for these calculations, although the functions included here make no effort to determine the length of your series. If you want to use a subset of your series, you'll need to truncate or subset the input data to the desired length.

Many other measures have been proposed to do similar reward to risk ranking. It is the opinion of this author that newer measures such as Sortino's `UpsidePotentialRatio` or Favre's modified `SharpeRatio` are both "better" measures, and should be preferred to the Calmar or Sterling Ratio.

Author(s)

Brian G. Peterson

References

Bacon, Carl. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004.

See Also

`Return.annualized`,
`maxDrawdown`,
`SharpeRatio.modified`,
`UpsidePotentialRatio`

Examples

```
data(managers)
CalmarRatio(managers[,1,drop=FALSE])
CalmarRatio(managers[,1:6])
SterlingRatio(managers[,1,drop=FALSE])
SterlingRatio(managers[,1:6])
```

`CAPM.CML.slope` *utility functions for single factor (CAPM) CML, SML, and RiskPremium*

Description

The Capital Asset Pricing Model, from which the popular `SharpeRatio` is derived, is a theory of market equilibrium. These utility functions provide values for various measures proposed in the CAPM.

Usage

```
CAPM.CML.slope(Rb, Rf = 0)

CAPM.CML(Ra, Rb, Rf = 0)

CAPM.RiskPremium(Ra, Rf = 0)

CAPM.SML.slope(Rb, Rf = 0)
```

Arguments

Rb	return vector of the benchmark asset
Rf	risk free rate, in same period as your returns
Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns

Details

At it's core, the CAPM is a single factor linear model. In light of the general ustility and wide use of single factor model, all functions in the CAPM suite will also be available with SFM (single factor model) prefixes.

The CAPM provides a justification for passive or index investing by positing that assets that are not on the efficient frontier will either rise or lower in price until they are on the efficient frontier of the market portfolio.

The CAPM Risk Premium on an investment is the measure of how much the asset's performance differs from the risk free rate. Negative Risk Premium generally indicates that the investment is a bad investment, and the money should be allocated to the risk free asset or to a different asset with a higher risk premium.

The Capital Market Line relates the excess expected return on an efficient market portfolio to it's Risk. The slope of the CML is the Sharpe Ratio for the market portfolio. The Security Market line is constructed by calculating the line of Risk Premium over `CAPM.beta`. For the benchmark asset this will be 1 over the risk premium of the benchmark asset. The CML also describes the only path allowed by the CAPM to a portfolio that outperforms the efficient frontier: it describes the line of reward/risk that a leveraged portfolio will occupy. So, according to CAPM, no portfolio constructed of the same assets can lie above the CML.

Probably the most complete criticism of CAPM in actual practice (as opposed to structural or theory critiques) is that it posits a market equilibrium, but is most often used only in a partial equilibrium setting, for example by using the S&P 500 as the benchmark asset. A better method of using and testing the CAPM would be to use a general equilibrium model that took global assets from all asset classes into consideration.

Chapter 7 of Ruppert(2004) gives an extensive overview of CAPM, its assumptions and deficiencies.

`SFM.RiskPremium` is the premium returned to the investor over the risk free asset

$$\overline{(R_a - R_f)}$$

`SFM.CML` calculates the expected return of the asset against the benchmark Capital Market Line

SFM.CML.slope calculates the slope of the Capital Market Line for looking at how a particular asset compares to the CML

SFM.SML.slope calculates the slope of the Security Market Line for looking at how a particular asset compares to the SML created by the benchmark

Author(s)

Brian G. Peterson

References

Sharpe, W.F. The Sharpe Ratio,*Journal of Portfolio Management*,Fall 1994, 49-58.
Sharpe, W.F. Capital Asset Prices: A theory of market equilibrium under conditions of risk. *Journal of finance*, vol 19, 1964, 425-442.
Ruppert, David. *Statistics and Finance, an Introduction*. Springer. 2004.

See Also

CAPM.beta CAPM.alpha SharpeRatio InformationRatio TrackingError ActivePremium

Examples

```
data(managers)
CAPM.CML.slope(managers["SP500 TR",drop=FALSE], managers[,10,drop=FALSE])
CAPM.CML(managers["HAM1",drop=FALSE], managers["SP500 TR",drop=FALSE], Rf=0)
CAPM.RiskPremium(managers["SP500 TR",drop=FALSE], Rf=0)
CAPM.RiskPremium(managers["HAM1",drop=FALSE], Rf=0)
CAPM.SML.slope(managers["SP500 TR",drop=FALSE], Rf=0)
# should create plots like in Ruppert 7.1 7.2
```

CAPM.dynamic *Time-varying conditional single factor model beta*

Description

CAPM is estimated assuming that betas and alphas change over time. It is assumed that the market prices of securities fully reflect readily available and public information. A matrix of market information variables, Z measures this information. Possible variables in Z could be the divident yield, Tresaury yield, etc. The betas of stocks and managed portfolios are allowed to change with market conditions:

Usage

```
CAPM.dynamic(Ra, Rb, Rf = 0, Z, lags = 1, ...)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of the asset returns
Rb	an xts, vector, matrix, data frame, timeSeries or zoo object of the benchmark asset return
Rf	risk free rate, in same period as your returns
Z	an xts, vector, matrix, data frame, timeSeries or zoo object of k variables that reflect public information
lags	number of lags before the current period on which the alpha and beta are conditioned
...	any other passthrough parameters

Details

$$\beta_p(z_t) = b_{0p} + B'_p z_t$$

where $z_t = Z_t - E[Z]$

- a normalized vector of the deviations of Z_t, B_p

- a vector with the same dimension as Z_t.

The coefficient b_{0p} can be interpreted as the "average beta" or the beta when all infromation variables are at their means. The elements of B_p measure the sensitivity of the conditional beta to the deviations of the Z_t from their means. In the similar way the time-varying conditional alpha is modeled:

$$\alpha_{pt} = \alpha_p(z_t) = \alpha_{0p} + A'_p z_t$$

The modified regression is therefore:

$$r_{pt+1} = \alpha_{0p} + A'_p z_t + b_{0p} r_{bt+1} + B'_p [z_t r_{bt+1}] + \mu_{pt+1}$$

Author(s)

Andrii Babii

References

J. Christopherson, D. Carino, W. Ferson. *Portfolio Performance Measurement and Benchmarking*. 2009. McGraw-Hill. Chapter 12.
Wayne E. Ferson and Rudi Schadt, "Measuring Fund Strategy and Performance in Changing Economic Conditions," *Journal of Finance*, vol. 51, 1996, pp.425-462

See Also

CAPM.beta

Examples

```
data(managers)
CAPM.dynamic(managers[,1,drop=FALSE], managers[,8,drop=FALSE],
             Rf=.035/12, Z=managers[, 9:10])

CAPM.dynamic(managers[80:120,1:6], managers[80:120,7,drop=FALSE],
             Rf=managers[80:120,10,drop=FALSE], Z=managers[80:120, 9:10])

CAPM.dynamic(managers[80:120,1:6], managers[80:120,8:7],
              managers[80:120,10,drop=FALSE], Z=managers[80:120, 9:10])
```

CAPM.epsilon *Regression epsilon of the return distribution*

Description

The regression epsilon is an error term measuring the vertical distance between the return predicted by the equation and the real result.

Usage

```
CAPM.epsilon(Ra, Rb, Rf = 0, ...)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
Rf	risk free rate, in same period as your returns
...	any other passthru parameters

Details

$$\epsilon_r = r_p - \alpha_r - \beta_r * b$$

where α_r is the regression alpha, β_r is the regression beta, r_p is the portfolio return and b is the benchmark return

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.71

Examples

```
data(portfolio_bacon)
print(SFM.epsilon(portfolio_bacon[,1], portfolio_bacon[,2])) #expected -0.013

data(managers)
print(SFM.epsilon(managers['1996',1], managers['1996',8]))
print(SFM.epsilon(managers['1996',1:5], managers['1996',8]))
```

`CAPM.jensenAlpha` *Jensen's alpha of the return distribution*

Description

The Jensen's alpha is the intercept of the regression equation in the Capital Asset Pricing Model and is in effect the exess return adjusted for systematic risk.

Usage

```
CAPM.jensenAlpha(
  Ra,
  Rb,
  Rf = 0,
  ...,
  method = "LS",
  family = "mopt",
  series = FALSE
)
```

Arguments

`Ra`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rb`	return vector of the benchmark asset
`Rf`	risk free rate, in same period as your returns
`...`	any other pass thru parameters
`method`	(Optional): string representing linear regression model, "LS" for Least Squares and "Rob" for robust
`family`	(Optional): If method == "Rob": This is a string specifying the name of the family of loss function to be used (current valid options are "bisquare", "opt" and "mopt"). Incomplete entries will be matched to the current valid options. Defaults to "mopt". Else: the parameter is ignored
`series`	(Optional): Boolean to return a time series of Jensen's Alpha instead of a single value. Defaults to FALSE.

Details

$$\alpha = r_p - r_f - \beta_p * (b - r_f)$$

where r_f is the risk free rate, β_r is the regression beta, r_p is the portfolio return and b is the benchmark return

Author(s)

Matthieu Lestel, Dhairya Jain

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.72

Examples

```
data(portfolio_bacon)
print(SFM.jensenAlpha(portfolio_bacon[, 1], portfolio_bacon[, 2])) # expected -0.014

data(managers)
print(SFM.jensenAlpha(managers["1996", 1], managers["1996", 8]))
print(SFM.jensenAlpha(managers["1996", 1:5], managers["1996", 8]))
```

CDaR.alpha *Conditional Drawdown alpha*

Description

The difference between the actual rate of return and the rate of return of the instrument estimated via the conditional drawdown beta is called $CDaR.alpha$ and it is the equivalent of the typical CAPM alpha but focusing on market drawdowns.

Positive $CDaR.alpha$ implies that the instrument performed better than it was predicted, and consequently, $CDaR.alpha$ can be used as a performance measure to rank instrument who overperform under market drawdowns.

Usage

```
CDaR.alpha(R, Rm, p = 0.95, weights = NULL, geometric = TRUE, type = NULL, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rm	an xts, vector, matrix, data frame, timeSeries or zoo object of benchmark returns
p	confidence level for calculation ,default(p=0.95)
weights	portfolio weighting vector, default NULL
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
type	(Optional) Overrides the p parameter. If "average" then p = 0 and if "max" then p = 1
...	any passthru variable

Value

The annualized alpha (input data are assumed to be of monthly frequency)

Author(s)

Tasos Grivas <tasos@openriskcalculator.com>, Pulkit Mehrotra

References

Zabarankin, M., Pavlikov, K., and S. Uryasev. Capital Asset Pricing Model (CAPM) with Drawdown Measure.Research Report 2012-9, ISE Dept., University of Florida,September 2012.

See Also

CDaR CDaR.beta

Examples

```
data(edhec)
CDaR.alpha(edhec[, 1], edhec[, 2])

CDaR.alpha(edhec[, 1], edhec[, 2], type = "max")

CDaR.alpha(edhec[, 1], edhec[, 2], type = "average")
```

CDaR.beta *Conditional Drawdown beta*

Description

The conditional drawdown beta is a measure of capturing performance under market drawdowns and it is given by the ratio of the average rate of return of the instrument over time periods corresponding to the $(1-p)T$ largest drawdowns of the benchmark portfolio.

The difference in CDaR and standard beta boils down to the fact that the standard beta accounts for the fund returns over the whole return history, including the upside while CDaR beta focuses only on market drawdowns.

Usage

```
CDaR.beta(R, Rm, p = 0.95, weights = NULL, geometric = TRUE, type = NULL, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rm	an xts, vector, matrix, data frame, timeSeries or zoo object of benchmark returns
p	confidence level for calculation ,default(p=0.95)
weights	portfolio weighting vector, default NULL, see Details
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
type	(Optional) Overrides the p parameter. If "average" then p = 0 and if "max" then p = 1
...	any passthru variable.

Author(s)

Tasos Grivas <tasos@openriskcalculator.com>,Pulkit Mehrotra

References

Zabarankin, M., Pavlikov, K., and S. Uryasev. Capital Asset Pricing Model (CAPM) with Drawdown Measure.Research Report 2012-9, ISE Dept., University of Florida,September 2012.

See Also

CDaR.alpha CDaR

Examples

```
data(edhec)
CDaR.beta(edhec[,1],edhec[,2])
CDaR.beta(edhec[,1],edhec[,2],type="max")
CDaR.beta(edhec[,1],edhec[,2],type="average")
```

CDD *Calculate Uryasev's proposed Conditional Drawdown at Risk (CDD or CDaR) measure*

Description

For some confidence level p, the conditional drawdown is the mean of the worst $p\%$ drawdowns.

Usage

```
CDD(
  R,
  weights = NULL,
  geometric = TRUE,
  invert = TRUE,
  p = 0.95,
  method = c("discrete", "average", "quantile"),
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
weights	portfolio weighting vector, default NULL, see Details
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
invert	TRUE/FALSE whether to invert the drawdown measure. see Details.
p	confidence level for calculation, default p=0.95
method	one of "discrete", "average", or "quantile". See Details.
...	any other passthru parameters

Details

The method parameter allows for three mathematical variations of CDD/CDaR:

discrete (Default) Computes the Expected Shortfall of the discrete, isolated peak-to-trough drawdown sequences. This isolates major discrete events without penalizing the prolonged duration of shallow drawdowns.

average Computes the mean of the continuous sample path of drawdowns that exceed the threshold. This strictly adheres to the Chekhlov (2003) continuous definition but can be heavily autocorrelated, mathematically overweighting long, shallow drawdowns over sharp, brief crashes.

quantile (Deprecated) A legacy implementation that merely computed the VaR (quantile) of the discrete drawdowns, rather than the Expected Shortfall (mean).

Author(s)

Brian G. Peterson

References

Chekhlov, A., Uryasev, S., and M. Zabarankin. Portfolio Optimization With Drawdown Constraints. B. Scherer (Ed.) Asset and Liability Management Tools, Risk Books, London, 2003 https://www.ise.ufl.edu/uryasev/drawdown.pdf

See Also

ES maxDrawdown

Examples

```
data(edhec)
t(round(CDD(edhec), 4))
```

---

chart.ACF *Create ACF chart or ACF with PACF two-panel chart*

Description

Creates an ACF chart or a two-panel plot with the ACF and PACF set to some specific defaults.

Usage

```
chart.ACF(R, maxlag = NULL, elementcolor = "gray", main = NULL, ...)

chart.ACFplus(R, maxlag = NULL, elementcolor = "gray", main = NULL, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
maxlag	the number of lags to calculate for, optional
elementcolor	the color to use for chart elements, defaults to "gray"
main	title of the plot; uses the column name by default.
...	any other passthru parameters

Note

Inspired by the website: https://www.stat.pitt.edu/stoffer/tsa2/Rcode/acf2.R "...here's an R function that will plot the ACF and PACF of a time series at the same time on the SAME SCALE, and it leaves out the zero lag in the ACF: acf2.R. If your time series is in x and you want the ACF and PACF of x to lag 50, the call to the function is acf2(x,50). The number of lags is optional, so acf2(x) will use a default number of lags [sqrt(n) + 10, where n is the number of observations]."

That description made a lot of sense, so it's implemented here for both the ACF alone and the ACF with the PACF.

Author(s)

Peter Carl

See Also

plot

Examples

```
data(edhec)
chart.ACFplus(edhec[,1,drop=FALSE])
```

---

chart.Bar *wrapper for barchart of returns*

Description

A wrapper to create a chart of periodic returns in a bar chart. This is a difficult enough graph to read that it doesn't get much use. Still, it is useful for viewing a single set of data.

Usage

```
chart.Bar(R, legend.loc = NULL, colorset = (1:12), ...)

charts.Bar(R, main = "Returns", cex.legend = 0.8, cex.main = 1, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
legend.loc	places a legend into one of nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center
colorset	color palette to use, set by default to rational choices
...	any other passthru parameters, see plot
main	sets the title text, such as in chart.TimeSeries
cex.legend	(deprecated) sets the legend text size, such as in chart.TimeSeries
cex.main	sets the title text size, such as in chart.TimeSeries

Details

This is really a wrapper for chart.TimeSeries, so several other attributes can also be passed.

Creates a plot of time on the x-axis and vertical lines for each period to indicate value on the y-axis.

Author(s)

Peter Carl

See Also

chart.TimeSeries
plot

Examples

```
data(edhec)
chart.Bar(edhec[,"Funds of Funds"], main="Monthly Returns")
```

---

chart.BarVaR *Periodic returns in a bar chart with risk metric overlay*

Description

Plots the periodic returns as a bar chart overlayed with a risk metric calculation.

Usage

```
chart.BarVaR(
  R,
  width = 0,
  gap = 12,
  methods = c("none", "ModifiedVaR", "GaussianVaR", "HistoricalVaR", "StdDev",
    "ModifiedES", "GaussianES", "HistoricalES"),
  p = 0.95,
  clean = c("none", "boudt", "geltner"),
  all = FALSE,
  ...,
  show.clean = FALSE,
  show.horizontal = FALSE,
  show.symmetric = FALSE,
  show.endvalue = FALSE,
  show.greenredbars = FALSE,
  legend.loc = "bottomleft",
  ylim = NA,
  lwd = 2,
  colorset = 1:12,
```

```
  lty = c(1, 2, 4, 5, 6),
  ypad = 0,
  legend.cex = 0.8,
  plot.engine = "default"
)

charts.BarVaR(
  R,
  main = "Returns",
  cex.legend = 0.8,
  colorset = 1:12,
  ylim = NA,
  ...,
  perpanel = NULL,
  show.yaxis = c("all", "firstonly", "alternating", "none")
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
width	periods specified for rolling-period calculations. Note that VaR, ES, and Std Dev with width=0 are calculated from the start of the timeseries
gap	numeric number of periods from start of series to use to train risk calculation
methods	Used to select the risk parameter of trailing `width` returns to use: May be any of: • none - does not add a risk line, • ModifiedVaR - uses Cornish-Fisher modified VaR, • GaussianVaR - uses traditional Value at Risk, • HistoricalVaR - calculates historical Value at Risk, • ModifiedES - uses Cornish-Fisher modified Expected Shortfall, • GaussianES - uses traditional Expected Shortfall, • HistoricalES - calculates historical Expected Shortfall, • StdDev - per-period standard deviation
p	confidence level for `VaR` or `ModifiedVaR` calculation, default is .99
clean	the method to use to clean outliers from return data prior to risk metric estimation. See `Return.clean` and `VaR` for more detail
all	if TRUE, calculates risk lines for each column given in R. If FALSE, only calculates the risk line for the first column
...	any other passthru parameters to `chart.TimeSeries`
show.clean	if TRUE and a method for 'clean' is specified, overlays the actual data with the "cleaned" data. See `Return.clean` for more detail
show.horizontal	if TRUE, shows a line across the timeseries at the value of the most recent VaR estimate, to help the reader evaluate the number of exceptions thus far

show.symmetric	if TRUE and the metric is symmetric, this will show the metric's positive values as well as negative values, such as for method "StdDev".
show.endvalue	if TRUE, show the final (out of sample) value
show.greenredbars	if TRUE, show the per-period returns using green and red bars for positive and negative returns
legend.loc	legend location, such as in chart.TimeSeries
ylim	set the y-axis limit, same as in plot
lwd	set the line width, same as in plot
colorset	color palette to use, such as in chart.TimeSeries
lty	set the line type, same as in plot
ypad	adds a numerical padding to the y-axis to keep the data away when legend.loc="bottom". See examples below.
legend.cex	sets the legend text size, such as in chart.TimeSeries
plot.engine	Choose the engine for plotting, including "default","dygraph","ggplot","plotly" and "googleVis"
main	sets the title text, such as in chart.TimeSeries
cex.legend	sets the legend text size, such as in chart.TimeSeries
perpanel	default NULL, controls column display
show.yaxis	one of "all", "firstonly", "alternating", or "none" to control where y axis is plotted in multipanel charts

Details

Note that StdDev and VaR are symmetric calculations, so a high and low measure will be plotted. ModifiedVaR, on the other hand, is assymetric and only a lower bound will be drawn.

Creates a plot of time on the x-axis and vertical lines for each period to indicate value on the y-axis. Overlays a line to indicate the value of a risk metric calculated at that time period.

charts.BarVaR places multile bar charts in a single graphic, with associated risk measures

Author(s)

Peter Carl

See Also

chart.TimeSeries
plot
ES
VaR
Return.clean

Examples

```
## Not run:  # not run on CRAN because of example time
data(managers)
# plain
chart.BarVaR(managers[,1,drop=FALSE], main="Monthly Returns")

# with risk line
chart.BarVaR(managers[,1,drop=FALSE],
methods="HistoricalVaR",
main="... with Empirical VaR from Inception")

# with lines for all managers in the sample
chart.BarVaR(managers[,1:6],
methods="GaussianVaR",
all=TRUE, lty=1, lwd=2,
colorset= c("red", rep("gray", 5)),
main="... with Gaussian VaR and Estimates for Peers")

# with multiple methods
chart.BarVaR(managers[,1,drop=FALSE],
methods=c("HistoricalVaR", "ModifiedVaR", "GaussianVaR"),
main="... with Multiple Methods")

# cleaned up a bit
chart.BarVaR(managers[,1,drop=FALSE],
methods=c("HistoricalVaR", "ModifiedVaR", "GaussianVaR"),
lwd=2, ypad=.01,
main="... with Padding for Bottom Legend")

# with 'cleaned' data for VaR estimates
chart.BarVaR(managers[,1,drop=FALSE],
methods=c("HistoricalVaR", "ModifiedVaR"),
lwd=2, ypad=.01, clean="boudt",
main="... with Robust ModVaR Estimate")

# Cornish Fisher VaR estimated with cleaned data,
# with horizontal line to show exceptions
chart.BarVaR(managers[,1,drop=FALSE],
methods="ModifiedVaR",
lwd=2, ypad=.01, clean="boudt",
show.horizontal=TRUE, lty=2,
main="... with Robust ModVaR and Line for Identifying Exceptions")

## End(Not run)
```

`chart.Boxplot` *box whiskers plot wrapper*

Description

A wrapper to create box and whiskers plot with some defaults useful for comparing distributions.

Usage

```
chart.Boxplot(
  R,
  names = TRUE,
  as.Tufte = FALSE,
  plot.engine = "default",
  sort.by = c(NULL, "mean", "median", "variance"),
  colorset = "black",
  symbol.color = "red",
  mean.symbol = 1,
  median.symbol = "|",
  outlier.symbol = 1,
  show.data = NULL,
  add.mean = TRUE,
  sort.ascending = FALSE,
  xlab = "Return",
  main = "Return Distribution Comparison",
  element.color = "darkgray",
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
names	logical. if TRUE, show the names of each series
as.Tufte	logical. default FALSE. if TRUE use method derived for Tufte for limiting chartjunk
plot.engine	choose the plot engine you wish to use: ggplot2, plotly, googlevis and default
sort.by	one of "NULL", "mean", "median", "variance"
colorset	color palette to use, set by default to rational choices
symbol.color	draws the symbols described in mean.symbol,median.symbol,outlier.symbol in the color specified
mean.symbol	symbol to use for the mean of the distribution
median.symbol	symbol to use for the median of the distribution
outlier.symbol	symbol to use for the outliers of the distribution
show.data	numerical vector of column numbers to display on top of boxplot, default NULL
add.mean	logical. if TRUE, show a line for the mean of all distributions plotted
sort.ascending	logical. If TRUE sort the distributions by ascending sort.by
xlab	set the x-axis label, same as in plot
main	set the chart title, same as in plot
element.color	specify the color of chart elements. Default is "darkgray"
...	any other passthru parameters

Details

We have also provided controls for all the symbols and lines in the chart. One default, set by `as.Tufte=TRUE`, will strip chartjunk and draw a Boxplot per recommendations by Edward Tufte. It can also be useful when comparing several series to sort them in order of ascending or descending "mean", "median", "variance" by use of `sort.by` and `sort.ascending=TRUE`.

Value

box plot of returns

Author(s)

Peter Carl

References

Tufte, Edward R. *The Visual Display of Quantitative Information*. Graphics Press. 1983. p. 124-129

See Also

`boxplot`

Examples

```
data(edhec)
chart.Boxplot(edhec)
chart.Boxplot(edhec, as.Tufte = TRUE)
```

---

chart.CaptureRatios *Chart of Capture Ratios against a benchmark*

Description

Scatter plot of Up Capture versus Down Capture against a benchmark

Usage

```
chart.CaptureRatios(
  Ra,
  Rb,
  main = "Capture Ratio",
  add.names = TRUE,
  xlab = "Downside Capture",
  ylab = "Upside Capture",
  colorset = 1,
  symbolset = 1,
  legend.loc = NULL,
```

```
  xlim = NULL,
  ylim = NULL,
  cex.legend = 1,
  cex.axis = 0.8,
  cex.main = 1,
  cex.lab = 1,
  element.color = "darkgray",
  benchmark.color = "darkgray",
  ...
)
```

Arguments

Ra	Returns to test, e.g., the asset to be examined
Rb	Returns of a benchmark to compare the asset with
main	Set the chart title, same as in plot
add.names	Plots the row name with the data point. Default TRUE. Can be removed by setting it to NULL
xlab	Set the x-axis label, as in plot
ylab	Set the y-axis label, as in plot
colorset	Color palette to use, set by default to "black"
symbolset	From pch in plot. Submit a set of symbols to be used in the same order as the data sets submitted
legend.loc	Places a legend into one of nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center.
xlim	set the x-axis limit, same as in plot
ylim	set the y-axis limit, same as in plot
cex.legend	The magnification to be used for sizing the legend relative to the current setting of 'cex'.
cex.axis	The magnification to be used for axis annotation relative to the current setting of 'cex', same as in plot.
cex.main	The magnification to be used for sizing the title relative to the current setting of 'cex'.
cex.lab	The magnification to be used for x and y labels relative to the current setting of 'cex'.
element.color	Specify the color of the box, axes, and other chart elements. Default is "darkgray"
benchmark.color	Specify the color of the benchmark reference and crosshairs. Default is "darkgray"
...	Any other passthru parameters to plot

Details

Scatter plot shows the coordinates of each set of returns' Up and Down Capture against a benchmark. The benchmark value is by definition plotted at (1,1) with solid crosshairs. A diagonal dashed line with slope equal to 1 divides the plot into two regions: above that line the UpCapture exceeds the DownCapture, and vice versa.

Author(s)

Peter Carl

See Also

plot,
par,
UpDownRatios,
table.UpDownRatios

Examples

```
data(managers)
chart.CaptureRatios(managers[,1:6], managers[,7,drop=FALSE])
```

---

chart.Correlation *correlation matrix chart*

Description

Visualization of a Correlation Matrix. On top the (absolute) value of the correlation plus the result of the cor.test as stars. On bottom, the bivariate scatterplots, with a fitted line. When the `method` parameter is set to "spearman" or "kendall", the lower panel scatterplots are automatically transformed to plot the monotonically ranked data rather than the raw values, providing visual alignment with the evaluated correlation method.

Usage

```
chart.Correlation(
  R,
  histogram = TRUE,
  method = c("pearson", "kendall", "spearman"),
  pch = 1,
  ...
)
```

Arguments

R	data for the x axis, can take matrix,vector, or timeseries
histogram	TRUE/FALSE whether or not to display a histogram
method	a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated.
pch	See par
...	any other passthru parameters into pairs

Note

based on plot at originally found at gallery.r-enthusiasts.com/graph/137

Author(s)

Peter Carl

See Also

table.Correlation par

Examples

```
data(managers)
chart.Correlation(managers[, 1:8], histogram = TRUE, pch = "+")
```

---

chart.CumReturns *Cumulates and graphs a set of periodic returns*

Description

Chart that cumulates the periodic returns given and draws a line graph of the results as a "wealth index".

Usage

```
chart.CumReturns(
  R,
  wealth.index = FALSE,
  geometric = TRUE,
  legend.loc = NULL,
  colorset = (1:12),
  begin = c("first", "axis"),
  plot.engine = "default",
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
wealth.index	if wealth.index is TRUE, shows the "value of $1", starting the cumulation of returns at 1 rather than zero
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
legend.loc	places a legend into one of nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center.
colorset	color palette to use, set by default to rational choices
begin	Align shorter series to: • first - prior value of the first column given for the reference or longer series or, • axis - the initial value (1 or zero) of the axis.
plot.engine	choose the plot engine you wish to use" ggplot2, plotly,dygraph,googlevis and default
...	any other passthru parameters

Details

Cumulates the return series and displays either as a wealth index or as cumulative returns.

Author(s)

Peter Carl

References

Bacon, Carl. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004.

See Also

chart.TimeSeries
plot

Examples

```
data(edhec)
chart.CumReturns(edhec[,"Funds of Funds"],main="Cumulative Returns")
chart.CumReturns(edhec[,"Funds of Funds"],wealth.index=TRUE, main="Growth of $1")
data(managers)
chart.CumReturns(managers,main="Cumulative Returns",begin="first")
chart.CumReturns(managers,main="Cumulative Returns",begin="axis")
```

`chart.Drawdown` *Time series chart of drawdowns through time*

Description

A time series chart demonstrating drawdowns from peak equity attained through time, calculated from periodic returns.

Usage

```
chart.Drawdown(
  R,
  geometric = TRUE,
  legend.loc = NULL,
  colorset = (1:12),
  plot.engine = "default",
  ...
)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`geometric`	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
`legend.loc`	places a legend into one of nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center.
`colorset`	color palette to use, set by default to rational choices
`plot.engine`	choose the plot engine you wish to use: ggplot2, plotly,dygraph,googlevis and default
`...`	any other passthru parameters

Details

Any time the cumulative returns dips below the maximum cumulative returns, it's a drawdown. Drawdowns are measured as a percentage of that maximum cumulative return, in effect, measured from peak equity.

Author(s)

Peter Carl

References

Bacon, C. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004. p. 88

See Also

plot
chart.TimeSeries
findDrawdowns
sortDrawdowns
maxDrawdown
table.Drawdowns
table.DownsideRisk

Examples

```
data(edhec)
chart.Drawdown(edhec[,c(1,2)],
main="Drawdown from Peak Equity Attained",
legend.loc="bottomleft")
```

---

chart.ECDF *Create an ECDF overlaid with a Normal CDF*

Description

Creates an emperical cumulative distribution function (ECDF) overlaid with a cumulative distribution function (CDF)

Usage

```
chart.ECDF(
  R,
  main = "Empirical CDF",
  xlab = "x",
  ylab = "F(x)",
  colorset = c("black", "#005AFF"),
  lwd = 1,
  lty = c(1, 1),
  element.color = "darkgray",
  xaxis = TRUE,
  yaxis = TRUE,
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
main	set the chart title, same as in plot
xlab	set the x-axis label, same as in plot
ylab	set the y-axis label, same as in plot

colorset	color palette to use, defaults to c("black", "#005AFF"), where first value is used to color the step function and the second color is used for the fitted normal
lwd	set the line width, same as in plot
lty	set the line type, same as in plot
element.color	specify the color of chart elements. Default is "darkgray"
xaxis	if true, draws the x axis
yaxis	if true, draws the y axis
...	any other passthru parameters to plot

Details

The empirical cumulative distribution function (ECDF for short) calculates the fraction of observations less or equal to a given value. The resulting plot is a step function of that fraction at each observation. This function uses ecdf and overlays the CDF for a fitted normal function as well. Inspired by a chart in Ruppert (2004).

Author(s)

Peter Carl

References

Ruppert, David. *Statistics and Finance, an Introduction*. Springer. 2004. Ch. 2 Fig. 2.5

See Also

plot, ecdf

Examples

```
data(edhec)
chart.ECDF(edhec[, 1, drop=FALSE])
```

---

chart.Events *Plots a time series with event dates aligned*

Description

Creates a time series plot where events given by a set of dates are aligned, with the adjacent prior and posterior time series data plotted in order. The x-axis is time, but relative to the date specified, e.g., number of months preceeding or following the events.

Usage

```
chart.Events(R, dates, prior = 12, post = 12, main = NULL, xlab = NULL, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
dates	a list of dates (e.g., c("09/03","05/06")) formatted the same as in R. This function matches the re-formatted row or index names (dates) with the given list, so to get a match the formatting needs to be correct.
prior	the number of periods to plot prior to the event. Interpreted as a positive number.
post	the number of periods to plot following to the event. Interpreted as a positive number.
main	set the chart title, same as in plot
xlab	set the x-axis label, same as in plot
...	any other passthru parameters to the plot function

Details

This is a chart that is commonly used for event studies in econometrics, usually with recession dates, to demonstrate the path of a time series going into and coming out of an event. The time axis is simply the number of periods prior and following the event, and each line represents a different event. Note that if time periods are close enough together and the window shown is wide enough, the data will appear to repeat. That can be confusing, but the function does not currently allow for different windows around each event.

Author(s)

Peter Carl

See Also

chart.TimeSeries,
plot,
par

Examples

```
## Not run:
data(managers)
n = table.Drawdowns(managers[,2,drop=FALSE])
chart.Events(Drawdowns(managers[,2,drop=FALSE]),
dates = n$Trough,
prior=max(na.omit(n$"To Trough")),
post=max(na.omit(n$Recovery)),
lwd=2, colorset=redfocus, legend.loc=NULL,
main = "Worst Drawdowns")

## End(Not run)
```

chart.Histogram *histogram of returns*

Description

Create a histogram of returns, with optional curve fits for density and normal. This is a wrapper function for hist, see the help for that function for additional arguments you may wish to pass in.

Usage

```
chart.Histogram(
  R,
  breaks = "FD",
  main = NULL,
  xlab = "Returns",
  ylab = "Frequency",
  methods = c("none", "add.density", "add.normal", "add.centered", "add.cauchy",
    "add.sst", "add.rug", "add.risk", "add.qqplot"),
  show.outliers = TRUE,
  colorset = c("lightgray", "#00008F", "#005AFF", "#23FFDC", "#ECFF13", "#FF4A00",
    "#800000"),
  border.col = "white",
  lwd = 2,
  xlim = NULL,
  ylim = NULL,
  element.color = "darkgray",
  note.lines = NULL,
  note.labels = NULL,
  note.cex = 0.7,
  note.color = "darkgray",
  probability = FALSE,
  p = 0.95,
  cex.axis = 0.8,
  cex.legend = 0.8,
  cex.lab = 1,
  cex.main = 1,
  xaxis = TRUE,
  yaxis = TRUE,
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
breaks	one of:

- a vector giving the breakpoints between histogram cells,

- a single number giving the number of cells for the histogram,
- a character string naming an algorithm to compute the number of cells (see 'Details'),
- a function to compute the number of cells.

For the last three the number is a suggestion only. see `hist` for details, default "FD"

`main` set the chart title, same as in `plot`

`xlab` set the x-axis label, same as in `plot`

`ylab` set the y-axis label, same as in `plot`

`methods` what to graph, one or more of:

- add.density to display the density plot
- add.normal to display a fitted normal distibution line over the mean
- add.centered to display a fitted normal line over zero
- add.rug to display a rug of the observations
- add.risk to display common risk metrics
- add.qqplot to display a small qqplot in the upper corner of the histogram plot

`show.outliers` logical; if TRUE (the default), the histogram will show all of the data points. If FALSE, it will show only the first through the fourth quartile and will exclude outliers.

`colorset` color palette to use, set by default to rational choices

`border.col` color to use for the border

`lwd` set the line width, same as in `plot`

`xlim` set the x-axis limit, same as in `plot`

`ylim` set the y-axis limits, same as in `plot`

`element.color` provides the color for drawing chart elements, such as the box lines, axis lines, etc. Default is "darkgray"

`note.lines` draws a vertical line through the value given.

`note.labels` adds a text label to vertical lines specified for note.lines.

`note.cex` The magnification to be used for note line labels relative to the current setting of 'cex'.

`note.color` specifies the color(s) of the vertical lines drawn.

`probability` logical; if TRUE, the histogram graphic is a representation of frequencies, the counts component of the result; if FALSE, probability densities, component density, are plotted (so that the histogram has a total area of one). Defaults to TRUE if and only if breaks are equidistant (and probability is not specified). see `hist`

`p` confidence level for calculation, default p=.99

`cex.axis` The magnification to be used for axis annotation relative to the current setting of 'cex', same as in `plot`.

`cex.legend` The magnification to be used for sizing the legend relative to the current setting of 'cex'.

cex.lab	The magnification to be used for x- and y-axis labels relative to the current setting of 'cex'.
cex.main	The magnification to be used for the main title relative to the current setting of 'cex'.
xaxis	if true, draws the x axis
yaxis	if true, draws the y axis
...	any other passthru parameters to plot

Details

The default for breaks is "FD". Other names for which algorithms are supplied are "Sturges" (see nclass.Sturges), "Scott", and "FD" / "Freedman-Diaconis" (with corresponding functions nclass.scott and nclass.FD). Case is ignored and partial matching is used. Alternatively, a function can be supplied which will compute the intended number of breaks as a function of R.

Note

Code inspired by a chart on: dead website zoonek2.free.fr /UNIX/48_R/03.html

Author(s)

Peter Carl

See Also

hist

Examples

```
    data(edhec)
    chart.Histogram(edhec[,'Equity Market Neutral',drop=FALSE])

    # version with more breaks and the
   # standard close fit density distribution
    chart.Histogram(edhec[,'Equity Market Neutral',drop=FALSE],
breaks=40, methods = c("add.density", "add.rug") )

    chart.Histogram(edhec[,'Equity Market Neutral',drop=FALSE],
methods = c( "add.density", "add.normal") )

    # version with just the histogram and
    # normal distribution centered on 0
    chart.Histogram(edhec[,'Equity Market Neutral',drop=FALSE],
methods = c( "add.density", "add.centered") )

    # add a rug to the previous plot
   # for more granularity on precisely where the distribution fell
    chart.Histogram(edhec[,'Equity Market Neutral',drop=FALSE],
methods = c( "add.centered", "add.density", "add.rug") )
```

```
    # now show a qqplot to give us another view
    # on how normal the data are
    chart.Histogram(edhec[,'Equity Market Neutral',drop=FALSE],
methods = c("add.centered","add.density","add.rug","add.qqplot"))

    # add risk measure(s) to show where those are
   # in relation to observed returns
    chart.Histogram(edhec[,'Equity Market Neutral',drop=FALSE],
methods = c("add.density","add.centered","add.rug","add.risk"))
```

`chart.QQPlot` *Plot a QQ chart*

Description

Plot the return data against any theoretical distribution.

Usage

```
chart.QQPlot(
  R,
  distribution = "norm",
  ylab = NULL,
  xlab = paste(distribution, "Quantiles"),
  main = NULL,
  las = par("las"),
  envelope = FALSE,
  labels = FALSE,
  col = c(1, 4),
  lwd = 2,
  pch = 1,
  cex = 1,
  line = c("quartiles", "robust", "none"),
  element.color = "darkgray",
  cex.axis = 0.8,
  cex.legend = 0.8,
  cex.lab = 1,
  cex.main = 1,
  xaxis = TRUE,
  yaxis = TRUE,
  ylim = NULL,
  distributionParameter = NULL,
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
distribution	root name of comparison distribution - e.g., 'norm' for the normal distribution; 't' for the t-distribution. See examples for other ideas.
ylab	set the y-axis label, as in plot
xlab	set the x-axis label, as in plot
main	set the chart title, same as in plot
las	set the direction of axis labels, same as in plot
envelope	confidence level for point-wise confidence envelope, or FALSE for no envelope.
labels	vector of point labels for interactive point identification, or FALSE for no labels.
col	color for points and lines; the default is the *second* entry in the current color palette (see 'palette' and 'par').
lwd	set the line width, as in plot
pch	symbols to use, see also plot
cex	symbols to use, see also plot
line	'quartiles' to pass a line through the quartile-pairs, or 'robust' for a robust-regression line; the latter uses the 'rlm' function in the 'MASS' package. Specifying 'line = "none"' suppresses the line.
element.color	provides the color for drawing chart elements, such as the box lines, axis lines, etc. Default is "darkgray"
cex.axis	The magnification to be used for axis annotation relative to the current setting of 'cex'
cex.legend	The magnification to be used for sizing the legend relative to the current setting of 'cex'
cex.lab	The magnification to be used for x- and y-axis labels relative to the current setting of 'cex'
cex.main	The magnification to be used for the main title relative to the current setting of 'cex'.
xaxis	if true, draws the x axis
yaxis	if true, draws the y axis
ylim	set the y-axis limits, same as in plot
distributionParameter	a string of the parameters of the distribution e.g., distributionParameter = 'location = 1, scale = 2, shape = 3, df = 4' for skew-T distribution
...	any other passthru parameters to the distribution function

Details

A Quantile-Quantile (QQ) plot is a scatter plot designed to compare the data to the theoretical distributions to visually determine if the observations are likely to have come from a known population. The empirical quantiles are plotted to the y-axis, and the x-axis contains the values of the theorical model. A 45-degree reference line is also plotted. If the empirical data come from the population with the choosen distribution, the points should fall approximately along this reference line. The larger the departure from the reference line, the greater the evidence that the data set have come from a population with a different distribution.

Author(s)

John Fox, ported by Peter Carl

References

main code forked/borrowed/ported from the excellent:
Fox, John (2007) *car: Companion to Applied Regression*
https://socserv.socsci.mcmaster.ca/jfox/

See Also

qqplot
qq.plot
plot

Examples

```
# you'll need lots of extra packages to run these examples of different distributions
## Not run:  # these examples require multiple packages from 'Suggests', so don't test on CRAN
library(MASS)
library(PerformanceAnalytics)
data(managers)
x = checkData(managers[,2, drop = FALSE], na.rm = TRUE, method = "vector")

# Panel 1: Normal distribution
chart.QQPlot(x, main = "Normal Distribution",
line=c("quartiles"), distribution = 'norm',
envelope=0.95)


# Panel 2, Log-Normal distribution
fit = fitdistr(1+x, 'lognormal')
chart.QQPlot(1+x, main = "Log-Normal Distribution", envelope=0.95,
    distribution='lnorm',distributionParameter='meanlog = fit$estimate[[1]],
    sdlog = fit$estimate[[2]]')

# Panel 3: Mixture Normal distribution
# library(nor1mix)
obj = norMixEM(x,m=2)
chart.QQPlot(x, main = "Normal Mixture Distribution",
line=c("quartiles"), distribution = 'norMix',  distributionParameter='obj',
envelope=0.95)


# Panel 4: Symmetric t distribution
library(sn)
n = length(x)
fit.tSN = st.mple(as.matrix(rep(1,n)),x,symmetr = TRUE)
names(fit.tSN$dp) = c("location","scale","dof")
round(fit.tSN$dp,3)
```

```
chart.QQPlot(x, main = "MO Symmetric t-Distribution QQPlot",
xlab = "quantilesSymmetricTdistEst",line = c("quartiles"),
envelope = .95, distribution = 't',
distributionParameter='df=fit.tSN$dp[3]',pch = 20)

# Panel 5: Skewed t distribution
fit.st = st.mple(as.matrix(rep(1,n)),x)
# fit.st = st.mple(y=x)  Produces same result as line above
names(fit.st$dp) = c("location","scale","skew","dof")
round(fit.st$dp,3)

chart.QQPlot(x, main = "MO Returns Skewed t-Distribution QQPlot",
xlab = "quantilesSkewedTdistEst",line = c("quartiles"),
envelope = .95, distribution = 'st',
distributionParameter = 'xi = fit.st$dp[1],
omega = fit.st$dp[2],alpha = fit.st$dp[3],
nu=fit.st$dp[4]',
pch = 20)

# Panel 6: Stable Parietian
library(fBasics)
fit.stable = stableFit(x,doplot=FALSE)
chart.QQPlot(x, main = "Stable Paretian Distribution", envelope=0.95,
             distribution = 'stable',
             distributionParameter = 'alpha = fit(stable.fit)$estimate[[1]],
                 beta = fit(stable.fit)$estimate[[2]],
                 gamma = fit(stable.fit)$estimate[[3]],
                 delta = fit(stable.fit)$estimate[[4]], pm = 0')

## End(Not run)
#end examples
```

`chart.Regression` *Takes a set of returns and relates them to a market benchmark in a scatterplot*

Description

Uses a scatterplot to display the relationship of a set of returns to a market benchmark. Fits a linear model and overlays the resulting model. Also overlays a Loess line for comparison.

Usage

```
chart.Regression(
  Ra,
  Rb,
  Rf = 0,
  excess.returns = FALSE,
  reference.grid = TRUE,
```

```
  main = "Title",
  ylab = NULL,
  xlab = NULL,
  xlim = NA,
  colorset = 1:12,
  symbolset = 1:12,
  element.color = "darkgray",
  legend.loc = NULL,
  ylog = FALSE,
  fit = c("loess", "linear", "conditional", "quadratic"),
  span = 2/3,
  degree = 1,
  family = c("symmetric", "gaussian"),
  ylim = NA,
  evaluation = 50,
  legend.cex = 0.8,
  cex = 0.8,
  lwd = 2,
  ...
)
```

Arguments

Ra	a vector of returns to test, e.g., the asset to be examined
Rb	a matrix, data.frame, or timeSeries of benchmark(s) to test the asset against
Rf	risk free rate, in same period as the returns
excess.returns	logical; should excess returns be used?
reference.grid	if true, draws a grid aligned with the points on the x and y axes
main	set the chart title, same as in plot
ylab	set the y-axis title, same as in plot
xlab	set the x-axis title, same as in plot
xlim	set the x-axis limit, same as in plot
colorset	color palette to use
symbolset	symbols to use, see also 'pch' in plot
element.color	provides the color for drawing chart elements, such as the box lines, axis lines, etc. Default is "darkgray"
legend.loc	places a legend into one of nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center.
ylog	Not used
fit	for values of "loess", "linear", or "conditional", plots a line to fit the data. Conditional lines are drawn separately for positive and negative benchmark returns. "Quadratic" is not yet implemented.
span	passed to loess line fit, as in loess.smooth
degree	passed to loess line fit, as in loess.smooth

family	passed to loess line fit, as in loess.smooth
ylim	set the y-axis limit, same as in plot
evaluation	passed to loess line fit, as in loess.smooth
legend.cex	set the legend size
cex	set the cex size, same as in plot
lwd	set the line width for fits, same as in lines
...	any other passthru parameters to plot

Author(s)

Peter Carl

References

Chapter 7 of Ruppert(2004) gives an extensive overview of CAPM, its assumptions and deficiencies.

See Also

plot

Examples

```
data(managers)
chart.Regression(managers[, 1:2, drop = FALSE],
managers[, 8, drop = FALSE],
Rf = managers[, 10, drop = FALSE],
excess.returns = TRUE, fit = c("loess", "linear"),
legend.loc = "topleft")
```

---

chart.RelativePerformance

relative performance chart between multiple return series

Description

Plots a time series chart that shows the ratio of the cumulative performance for two assets at each point in time and makes periods of under- or out-performance easier to see.

Usage

```
chart.RelativePerformance(
  Ra,
  Rb,
  main = "Relative Performance",
  xaxis = TRUE,
  colorset = (1:12),
  legend.loc = NULL,
  ylog = FALSE,
  elementcolor = "darkgray",
  lty = 1,
  cex.legend = 0.7,
  ...
)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
main	set the chart title, same as in plot
xaxis	if true, draws the x axis
colorset	color palette to use, set by default to rational choices
legend.loc	places a legend into one of nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center.
ylog	TRUE/FALSE set the y-axis to logarithmic scale, similar to plot, default FALSE
elementcolor	provides the color for drawing less-important chart elements, such as the box lines, axis lines, etc. replaces darken
lty	set the line type, same as in plot
cex.legend	the magnification to be used for sizing the legend relative to the current setting of 'cex'.
...	any other passthru parameters

Details

To show under- and out-performance through different periods of time, a time series view is more helpful. The value of the chart is less important than the slope of the line. If the slope is positive, the first asset (numerator) is outperforming the second, and vice versa. May be used to look at the returns of a fund relative to each member of the peer group and the peer group index. Alternatively, it might be used to assess the peers individually against an asset class or peer group index.

Author(s)

Peter Carl

See Also

Return.relative

Examples

```
data(managers)
chart.RelativePerformance(managers[, 1:6, drop=FALSE],
managers[, 8, drop=FALSE],
colorset=rich8equal, legend.loc="bottomright",
main="Relative Performance to S&P")
```

`chart.RiskReturnScatter` *scatter chart of returns vs risk for comparing multiple instruments*

Description

A wrapper to create a scatter chart of annualized returns versus annualized risk (standard deviation) for comparing manager performance. Also puts a box plot into the margins to help identify the relative performance quartile.

Usage

```
chart.RiskReturnScatter(
  R,
  Rf = 0,
  main = "Annualized Return and Risk",
  add.names = TRUE,
  xlab = "Annualized Risk",
  ylab = "Annualized Return",
  method = "calc",
  geometric = TRUE,
  scale = NA,
  add.sharpe = c(1, 2, 3),
  add.boxplots = FALSE,
  colorset = 1,
  symbolset = 1,
  element.color = "darkgray",
  legend.loc = NULL,
  xlim = NULL,
  ylim = NULL,
  cex.legend = 1,
  cex.axis = 0.8,
  cex.main = 1,
  cex.lab = 1,
  na.fill = FALSE,
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rf	risk free rate, in same period as your returns
main	set the chart title, same as in plot
add.names	plots the row name with the data point. default TRUE. Can be removed by setting it to NULL
xlab	set the x-axis label, as in plot
ylab	set the y-axis label, as in plot
method	if set as "calc", then the function will calculate values from the set of returns passed in. If method is set to "nocalc" then we assume that R is a column of return and a column of risk (e.g., annualized returns, annualized risk), in that order. Other method cases may be set for different risk/return calculations.
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
scale	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
add.sharpe	this draws a Sharpe ratio line that indicates Sharpe ratio levels of c(1,2,3). Lines are drawn with a y-intercept of the risk free rate and the slope of the appropriate Sharpe ratio level. Lines should be removed where not appropriate (e.g., sharpe.ratio = NULL).
add.boxplots	TRUE/FALSE adds a boxplot summary of the data on the axis
colorset	color palette to use, set by default to rational choices
symbolset	from pch in plot, submit a set of symbols to be used in the same order as the data sets submitted
element.color	provides the color for drawing chart elements, such as the box lines, axis lines, etc. Default is "darkgray"
legend.loc	places a legend into one of nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center.
xlim	set the x-axis limit, same as in plot
ylim	set the y-axis limit, same as in plot
cex.legend	The magnification to be used for sizing the legend relative to the current setting of 'cex'.
cex.axis	The magnification to be used for axis annotation relative to the current setting of 'cex', same as in plot.
cex.main	The magnification to be used for sizing the title relative to the current setting of 'cex'.
cex.lab	The magnification to be used for x and y labels relative to the current setting of 'cex'.
na.fill	optional numeric value to replace NAs, e.g. 0.
...	any other passthru parameters to plot

Note

Code inspired by a chart on dead website zoonek2.free.fr /UNIX/48_R/03.html

Author(s)

Peter Carl

Examples

```
data(edhec)
chart.RiskReturnScatter(edhec, Rf = .04 / 12)
chart.RiskReturnScatter(edhec, Rf = .04 / 12, add.boxplots = TRUE)
```

---

chart.RollingCorrelation

chart rolling correlation fo multiple assets

Description

A wrapper to create a chart of rolling correlation metrics in a line chart

Usage

```
chart.RollingCorrelation(
  Ra,
  Rb,
  width = 12,
  xaxis = TRUE,
  legend.loc = NULL,
  colorset = (1:12),
  ...,
  fill = NA
)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
width	number of periods to apply rolling function window over
xaxis	if true, draws the x axis
legend.loc	places a legend into one of nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center.
colorset	color palette to use, set by default to rational choices
...	any other passthru parameters
fill	a three-component vector or list (recycled otherwise) providing filling values at the left/within/to the right of the data range. See the fill argument of na.fill for details.

Details

The previous parameter na.pad has been replaced with fill; use fill = NA instead of na.pad = TRUE, or fill = NULL instead of na.pad = FALSE.

Author(s)

Peter Carl

Examples

```
# First we get the data
data(managers)
chart.RollingCorrelation(managers[, 1:6, drop=FALSE],
managers[, 8, drop=FALSE],
colorset=rich8equal, legend.loc="bottomright",
width=24, main = "Rolling 12-Month Correlation")
```

---

chart.RollingMean *chart the rolling mean return*

Description

A wrapper to create a rolling mean return chart with 95

Usage

```
chart.RollingMean(
  R,
  width = 12,
  xaxis = TRUE,
  ylim = NULL,
  lwd = c(2, 1, 1),
  ...,
  fill = NA
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
width	number of periods to apply rolling function window over
xaxis	if true, draws the x axis
ylim	set the y-axis limit, same as in plot
lwd	set the line width, same as in plot. Specified in order of the main line and the two confidence bands.
...	any other passthru parameters

fill	a three-component vector or list (recycled otherwise) providing filling values at the left/within/to the right of the data range. See the fill argument of na.fill for details.

Details

The previous parameter na.pad has been replaced with fill; use fill = NA instead of na.pad = TRUE, or fill = NULL instead of na.pad = FALSE.

Author(s)

Peter Carl

Examples

```
data(edhec)
chart.RollingMean(edhec[, 9, drop = FALSE])
```

---

chart.RollingPerformance

wrapper to create a chart of rolling performance metrics in a line chart

---

Description

A wrapper to create a chart of rolling performance metrics in a line chart

Usage

```
chart.RollingPerformance(
  R,
  width = 12,
  FUN = "Return.annualized",
  ...,
  ylim = NULL,
  main = NULL,
  fill = NA
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
width	number of periods to apply rolling function window over
FUN	any function that can be evaluated using a single set of returns (e.g., rolling CAPM.beta won't work, but Return.annualized will)
...	any other passthru parameters to plot or the function specified
ylim	set the y-axis limit, same as in plot

main	set the chart title, same as in plot
fill	a three-component vector or list (recycled otherwise) providing filling values at the left/within/to the right of the data range. See the fill argument of na.fill for details.

Details

The parameter na.pad has been deprecated; use fill = NA instead of na.pad = TRUE, or fill = NULL instead of na.pad = FALSE.

Author(s)

Peter Carl

See Also

charts.RollingPerformance, rollapply

Examples

```
data(edhec)
chart.RollingPerformance(edhec[, 1:3], width = 24)
chart.RollingPerformance(edhec[, 1:3],
FUN = 'mean', width = 24, colorset = rich8equal,
lwd = 2, legend.loc = "topleft",
main = "Rolling 24-Month Mean Return")
chart.RollingPerformance(edhec[, 1:3],
FUN = 'SharpeRatio.annualized', width = 24,
colorset = rich8equal, lwd = 2, legend.loc = "topleft",
main = "Rolling 24-Month Sharpe Ratio")
```

---

chart.RollingQuantileRegression

A wrapper to create charts of relative regression performance through time

---

Description

A wrapper to create a chart of relative regression performance through time

Usage

```
chart.RollingQuantileRegression(
  Ra,
  Rb,
  width = 12,
  Rf = 0,
```

```
  attribute = c("Beta", "Alpha", "R-Squared"),
  main = NULL,
  na.pad = TRUE,
  ...
)

chart.RollingRegression(
  Ra,
  Rb,
  width = 12,
  Rf = 0,
  attribute = c("Beta", "Alpha", "R-Squared"),
  main = NULL,
  na.pad = TRUE,
  ...
)

charts.RollingRegression(
  Ra,
  Rb,
  width = 12,
  Rf = 0,
  main = NULL,
  legend.loc = NULL,
  event.labels = NULL,
  ...
)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
width	number of periods to apply rolling function window over
Rf	risk free rate, in same period as your returns
attribute	one of "Beta","Alpha","R-Squared" for which attribute to show
main	set the chart title, same as in plot
na.pad	TRUE/FALSE If TRUE it adds any times that would not otherwise have been in the result with a value of NA. If FALSE those times are dropped.
...	any other passthru parameters to chart.TimeSeries
legend.loc	used to set the position of the legend
event.labels	if not null and event.lines is not null, this will apply a list of text labels to the vertical lines drawn

Details

A group of charts in charts.RollingRegression displays alpha, beta, and R-squared estimates in three aligned charts in a single device.

The attribute parameter is probably the most confusing. In mathematical terms, the different choices yield the following:

Alpha - shows the y-intercept
Beta - shows the slope of the regression line
R-Squared - shows the degree of fit of the regression to the data

`chart.RollingQuantileRegression` uses `rq` rather than `lm` for the regression, and may be more robust to outliers in the data.

Note

Most inputs are the same as "`plot`" and are principally included so that some sensible defaults could be set.

Author(s)

Peter Carl

See Also

`lm`
`rq`

Examples

```
# First we load the data
data(managers)
chart.RollingRegression(managers[, 1, drop=FALSE],
managers[, 8, drop=FALSE], Rf = .04/12)
charts.RollingRegression(managers[, 1:6],
managers[, 8, drop=FALSE], Rf = .04/12,
colorset = rich6equal, legend.loc="topleft")
dev.new()
chart.RollingQuantileRegression(managers[, 1, drop=FALSE],
managers[, 8, drop=FALSE], Rf = .04/12)
# not implemented yet
#charts.RollingQuantileRegression(managers[, 1:6],
# managers[, 8, drop=FALSE], Rf = .04/12,
# colorset = rich6equal, legend.loc="topleft")
```

---

chart.Scatter *wrapper to draw scatter plot with sensible defaults*

Description

Draws a scatter chart. This is another chart "primitive", since it only contains a set of sensible defaults.

Usage

```
chart.Scatter(
  x,
  y,
  reference.grid = TRUE,
  main = "Title",
  ylab = NULL,
  xlab = NULL,
  xlim = NULL,
  ylim = NULL,
  colorset = 1,
  symbolset = 1,
  element.color = "darkgray",
  cex.axis = 0.8,
  cex.legend = 0.8,
  cex.lab = 1,
  cex.main = 1,
  ...
)
```

Arguments

x	data for the x axis, can take matrix,vector, or timeseries
y	data for the y axis, can take matrix,vector, or timeseries
reference.grid	if true, draws a grid aligned with the points on the x and y axes
main	set the chart title, same as in plot
ylab	set the y-axis label, as in plot
xlab	set the x-axis label, as in plot
xlim	set the x-axis limit, same as in plot
ylim	set the y-axis limit, same as in plot
colorset	color palette to use, set by default to rational choices
symbolset	from pch in plot, submit a set of symbols to be used in the same order as the data sets submitted
element.color	provides the color for drawing chart elements, such as the box lines, axis lines, etc. Default is "darkgray"
cex.axis	The magnification to be used for axis annotation relative to the current setting of 'cex', same as in plot.
cex.legend	The magnification to be used for sizing the legend relative to the current setting of 'cex'.
cex.lab	The magnification to be used for x- and y-axis labels relative to the current setting of 'cex'
cex.main	The magnification to be used for the main title relative to the current setting of 'cex'.
...	any other passthru parameters

Note

Most inputs are the same as "plot" and are principally included so that some sensible defaults could be set.

Author(s)

Peter Carl

See Also

plot

Examples

```
## Not run:
  data(edhec)
  chart.Scatter(edhec[,1],edhec[,2])

## End(Not run)
```

---

chart.SFM *Compare SFM estimated using robust estimators with that estimated by OLS*

Description

This function for single factor models (SFM's) with a slope and intercept allows the user to easily make a scatter plot of asset returns versus benchmark returns, such as the SP500, with two overlaid straight-line fits, one obtained using least squares (LS), which can be very adversely influenced by outliers, and one obtained using a highly robust regression estimate that is not much influenced by outliers. The plot allows the user to see immediately whether or not any outliers result in a distorted LS fit that does not fit the bulk of the data, while the robust estimator results In a good fit to the bulk of the data. The plot contains a legend with LS slope estimate and the robust slope estimate, with estimate standard errors in parentheses.

Usage

```
chart.SFM(
  Ra,
  Rb,
  Rf = 0,
  main = NULL,
  ylim = NULL,
  xlim = NULL,
  family = "mopt",
  xlab = NULL,
```

```
  ylab = NULL,
  legend.loc = "topleft",
  makePct = FALSE
)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
Rf	risk free rate, in same period as your returns
main	Title of the generated plot. Defaults to "lm vs lmRobdetMM"
ylim	Limits on the y-axis of the plots. Defaults to min-max
xlim	Limits on the x-axis of the plots. Defaults to min-max
family	(Optional): This is a string specifying the name of the family of loss function to be used (current valid options are "bisquare", "opt" and "mopt"). Incomplete entries will be matched to the current valid options. Defaults to "mopt".
xlab	Title of the x-axis of the plots. Defaults to "Benchmark Returns"
ylab	Title of the y-axis of the plots. Defaults to "Asset Returns"
legend.loc	Position of legends. See plot() function for more info.
makePct	If Returns should be converted to percentage. Defaults to False

Details

The function chart.SFM computes the robust fit with the function lmrobdetMM contained in the package RobStatTM available at CRAN. The function lmrobdetMM has a default robust regression estimate called the mopt estimate, which is used by chart.SFM. For details on lmrobdetMM, see reference [1]. The plot made by chart.SFM has two parallel dotted lines that define a strip in the asset returns versus benchmark returns space, and data points that fall outside that strip are defined as outliers, and as such are rejected, i.e., deleted by the lmrobdetMM estimator.

Author(s)

Dhairya Jain, Doug Martin, Dan Xia

References

Martin, R. D. and Xia, D. Z. (2021). Robust Time Series Factor Models, SSRN: https://www.ssrn.com/abstract=3905345. *To appear in the Journal of Asset Management in 2022*
Ruppert, David. *Statistics and Finance, an Introduction*. Springer. 2004.
Sharpe, W.F. Capital Asset Prices: A theory of market equilibrium under conditions of risk. *Journal of finance*, vol 19, 1964, 425-442.

Examples

```
 # CRAN tests if Suggested packages are loaded
   data(managers)

mgrs <- managers["2002/"]  # So that all have managers have complete history
   names(mgrs)[7:10] <- c("LSEQ","SP500","Bond10Yr","RF") # Short names for last 3
   plot.zoo(mgrs)

   chart.SFM(mgrs$HAM1, mgrs$SP500, Rf=mgrs$RF)

   for(k in 1:7){
        chart.SFM(mgrs[,k],mgrs$SP500,mgrs$RF,makePct = TRUE,
                       main = names(mgrs[,k]))
   }
```

---

chart.SnailTrail *chart risk versus return over rolling time periods*

Description

A chart that shows rolling calculations of annualized return and annualized standard deviation have proceeded through time. Lines and dots are darker for more recent time periods.

Usage

```
chart.SnailTrail(
  R,
  Rf = 0,
  main = "Annualized Return and Risk",
  add.names = c("all", "lastonly", "firstandlast", "none"),
  xlab = "Annualized Risk",
  ylab = "Annualized Return",
  add.sharpe = c(1, 2, 3),
  colorset = 1:12,
  symbolset = 16,
  legend.loc = NULL,
  xlim = NULL,
  ylim = NULL,
  width = 12,
  stepsize = 12,
  lty = 1,
  lwd = 2,
  cex.axis = 0.8,
  cex.main = 1,
  cex.lab = 1,
  cex.text = 0.8,
```

```
  cex.legend = 0.8,
  element.color = "darkgray",
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rf	risk free rate, in same period as your returns
main	set the chart title, same as in plot
add.names	plots the row name with the data point. default TRUE. Can be removed by setting it to NULL
xlab	set the x-axis label, as in plot
ylab	set the y-axis label, as in plot
add.sharpe	this draws a Sharpe ratio line that indicates Sharpe ratio levels of c(1,2,3). Lines are drawn with a y-intercept of the risk free rate and the slope of the appropriate Sharpe ratio level. Lines should be removed where not appropriate (e.g., sharpe.ratio = NULL).
colorset	color palette to use, set by default to rational choices
symbolset	from pch in plot, submit a set of symbols to be used in the same order as the data sets submitted
legend.loc	places a legend into one of nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center.
xlim	set the x-axis limit, same as in plot
ylim	set the y-axis limit, same as in plot
width	number of periods to apply rolling calculations over, sometimes referred to as a 'window'
stepsize	the frequency with which to make the rolling calculation
lty	set the line type, same as in plot
lwd	set the line width, same as in plot
cex.axis	The magnification to be used for sizing the axis text relative to the current setting of 'cex', similar to plot.
cex.main	The magnification to be used for sizing the main chart relative to the current setting of 'cex', as in plot.
cex.lab	The magnification to be used for sizing the label relative to the current setting of 'cex', similar to plot.
cex.text	The magnification to be used for sizing the text relative to the current setting of 'cex', similar to plot.
cex.legend	The magnification to be used for sizing the legend relative to the current setting of 'cex'.
element.color	provides the color for drawing chart elements, such as the box lines, axis lines, etc. Default is "darkgray"
...	any other passthru parameters

Author(s)

Peter Carl

References

~put references to the literature/web site here ~

See Also

chart.RiskReturnScatter

Examples

```
data(managers)
chart.SnailTrail(managers[, c("HAM2", "SP500 TR"), drop = FALSE],
  width = 36, stepsize = 12,
  colorset = c("red", "orange"),
  add.names = "firstandlast",
  rf = .04 / 12,
  main = "Trailing 36-month Performance Calc'd Every 12 Months"
)
```

---

chart.StackedBar *create a stacked bar plot*

Description

This creates a stacked column chart with time on the horizontal axis and values in categories. This kind of chart is commonly used for showing portfolio 'weights' through time, although the function will plot any values by category.

Usage

```
chart.StackedBar(
  w,
  colorset = NULL,
  space = 0.2,
  cex.axis = 0.8,
  cex.legend = 0.8,
  cex.lab = 1,
  cex.labels = 0.8,
  cex.main = 1,
  xaxis = TRUE,
  legend.loc = "under",
  element.color = "darkgray",
  unstacked = TRUE,
  xlab = "Date",
```

```
  ylab = "Value",
  ylim = NULL,
  date.format = "%b %y",
  major.ticks = "auto",
  minor.ticks = TRUE,
  las = 0,
  xaxis.labels = NULL,
  ...
)
```

Arguments

w	a matrix, data frame or zoo object of values to be plotted. Rownames should contain dates or period labels; column names should indicate categories. See examples for detail.
colorset	color palette to use, set by default to rational choices
space	the amount of space (as a fraction of the average bar width) left before each bar, as in barplot. Default is 0.2.
cex.axis	The magnification to be used for sizing the axis text relative to the current setting of 'cex', similar to plot.
cex.legend	The magnification to be used for sizing the legend relative to the current setting of 'cex', similar to plot.
cex.lab	The magnification to be used for x- and y-axis labels relative to the current setting of 'cex'.
cex.labels	The magnification to be used for event line labels relative to the current setting of 'cex'.
cex.main	The magnification to be used for the chart title relative to the current setting of 'cex'.
xaxis	If true, draws the x axis
legend.loc	places a legend into a location on the chart similar to chart.TimeSeries. The default, "under," is the only location currently implemented for this chart. Use 'NULL' to remove the legend.
element.color	provides the color for drawing less-important chart elements, such as the box lines, axis lines, etc.
unstacked	logical. If set to 'TRUE' *and* only one row of data is submitted in 'w', then the chart creates a normal column chart. If more than one row is submitted, then this is ignored. See examples below.
xlab	the x-axis label, which defaults to 'NULL'.
ylab	Set the y-axis label, same as in plot
ylim	set the y-axis limit, same as in plot
date.format	Re-format the dates for the xaxis; the default is "%m/%y"
major.ticks	Should major tickmarks be drawn and labeled, default 'auto'
minor.ticks	Should minor tickmarks be drawn, default TRUE

las	sets the orientation of the axis labels, as described in par. Defaults to '3'.
xaxis.labels	Allows for non-date labeling of date axes, default is NULL
...	arguments to be passed to barplot.

Details

This function is a wrapper for barplot but adds three additional capabilities. First, it calculates and sets a bottom margin for long column names that are rotated vertically. That doesn't always result in the prettiest chart, but it does ensure readable labels.

Second, it places a legend "under" the graph rather than within the bounds of the chart (which would obscure the data). The legend is created from the column names. The default is to create the legend when there's more than one row of data being presented. If there is one row of data, the chart may be "unstacked" and the legend removed.

Third, it plots or stacks negative values from an origin of zero, similar to the behavior of barchart from the 'lattice' package.

Note

The "w" attribute is so named because this is a popular way to show portfolio weights through time. That being said, this function is not limited to portfolio weight values and does not provide any normalization so that the chart can be used more generally with different data.

The principal drawback of stacked column charts is that it is very difficult for the reader to judge size of 2nd, 3rd, etc., data series because they do not have common baseline. Another is that with a large number of series, the colors may be difficult to discern. As alternatives, Cleveland advocates the use of trellis like displays, and Tufte advocates the use of small multiple charts.

Author(s)

Peter Carl

References

Cleveland, W.S. (1994), The Elements of Graphing Data, Summit, NJ: Hobart Press.

Tufte, Edward R. (2001) The Visual Display of Quantitative Information, 2nd edition. The Graphics Press, Cheshire, Connecticut. See https://www.edwardtufte.com for this and other references.

See Also

barplot, par

Examples

```
data(weights)
head(weights)

# With the legend "under" the chart
chart.StackedBar(weights, date.format = "%Y", cex.legend = 0.7, colorset = rainbow12equal)
```

```
# Without the legend
chart.StackedBar(weights, colorset = rainbow12equal, legend.loc = NULL)

# for one row of data, use 'unstacked' for a better chart
chart.StackedBar(weights[1, , drop = FALSE], unstacked = TRUE, las = 3)
```

---

chart.TimeSeries *Creates a time series chart with some extensions.*

Description

Draws a line chart and labels the x-axis with the appropriate dates. This is really a "primitive", since it extends the base plot and standardizes the elements of a chart. Adds attributes for shading areas of the timeline or aligning vertical lines along the timeline. This function is intended to be used inside other charting functions.

Usage

```
chart.TimeSeries(
  R,
  ...,
  auto.grid = TRUE,
  xaxis = TRUE,
  yaxis = TRUE,
  yaxis.right = FALSE,
  type = "l",
  lty = 1,
  lwd = 1,
  las = par("las"),
  main = "",
  ylab = "",
  xlab = "",
  date.format.in = "%Y-%m-%d",
  date.format = NULL,
  xlim = NULL,
  ylim = NULL,
  element.color = "darkgray",
  event.lines = NULL,
  event.labels = NULL,
  period.areas = NULL,
  event.color = "darkgray",
  period.color = "aliceblue",
  colorset = (1:12),
  pch = (1:12),
  legend.loc = NULL,
  ylog = FALSE,
  cex.axis = 0.8,
```

```
    cex.legend = 0.8,
    cex.lab = 1,
    cex.labels = 0.8,
    cex.main = 1,
    major.ticks = "auto",
    minor.ticks = TRUE,
    grid.color = "lightgray",
    grid.lty = "dotted",
    xaxis.labels = NULL,
    plot.engine = "default",
    yaxis.pct = FALSE
  )

  chart.TimeSeries.base(
    R,
    auto.grid,
    xaxis,
    yaxis,
    yaxis.right,
    type,
    lty,
    lwd,
    las,
    main,
    ylab,
    xlab,
    date.format.in,
    date.format,
    xlim,
    ylim,
    element.color,
    event.lines,
    event.labels,
    period.areas,
    event.color,
    period.color,
    colorset,
    pch,
    legend.loc,
    ylog,
    cex.axis,
    cex.legend,
    cex.lab,
    cex.labels,
    cex.main,
    major.ticks,
    minor.ticks,
    grid.color,
```

```
  grid.lty,
  xaxis.labels,
  plot.engine,
  yaxis.pct,
  ...
)

chart.TimeSeries.builtin(
  R,
  auto.grid,
  xaxis,
  yaxis,
  yaxis.right,
  type,
  lty,
  lwd,
  las,
  main,
  ylab,
  xlab,
  date.format.in,
  date.format,
  xlim,
  ylim,
  element.color,
  event.lines,
  event.labels,
  period.areas,
  event.color,
  period.color,
  colorset,
  pch,
  legend.loc,
  ylog,
  cex.axis,
  cex.legend,
  cex.lab,
  cex.labels,
  cex.main,
  major.ticks,
  minor.ticks,
  grid.color,
  grid.lty,
  xaxis.labels,
  yaxis.pct,
  ...
)
```

```
chart.TimeSeries.dygraph(R, type = "l", ylog = FALSE)

chart.TimeSeries.ggplot2(
  R,
  auto.grid,
  xaxis,
  yaxis,
  yaxis.right,
  type,
  lty,
  lwd,
  las,
  main,
  ylab,
  xlab,
  date.format.in,
  date.format,
  xlim,
  ylim,
  element.color,
  event.lines,
  event.labels,
  period.areas,
  event.color,
  period.color,
  colorset,
  pch,
  legend.loc,
  ylog,
  cex.axis,
  cex.legend,
  cex.lab,
  cex.labels,
  cex.main,
  major.ticks,
  minor.ticks,
  grid.color,
  grid.lty,
  xaxis.labels,
  plot.engine,
  yaxis.pct
)

chart.TimeSeries.googlevis(R, xlab, ylab, main, type = "l", ylog = FALSE)

chart.TimeSeries.plotly(R, main, type = "l", ylog = FALSE, ...)

charts.TimeSeries(R, space = 0, main = "Returns", ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters
auto.grid	if true, draws a grid aligned with the points on the x and y axes
xaxis	if true, draws the x axis
yaxis	if true, draws the y axis
yaxis.right	if true, draws the y axis on the right-hand side of the plot
type	set the chart type, same as in plot
lty	set the line type, same as in plot
lwd	set the line width, same as in plot
las	set the axis label rotation, same as in plot
main	set the chart title, same as in plot
ylab	set the y-axis label, same as in plot
xlab	set the x-axis label, same as in plot
date.format.in	allows specification of other date formats in the data object, defaults to "%Y-%m-%d"
date.format	re-format the dates for the xaxis; the default is "%m/%y"
xlim	set the x-axis limit, same as in plot
ylim	set the y-axis limit, same as in plot
element.color	provides the color for drawing chart elements, such as the box lines, axis lines, etc. Default is "darkgray"
event.lines	if not null, vertical lines will be drawn to indicate that an event happened during that time period. event.lines should be a list of dates (e.g., c("09/03","05/06")) formatted the same as date.format. This function matches the re-formatted row names (dates) with the events.list, so to get a match the formatting needs to be correct.
event.labels	if not null and event.lines is not null, this will apply a list of text labels (e.g., c("This Event", "That Event") to the vertical lines drawn. See the example below.
period.areas	these are shaded areas described by start and end dates in a vector of xts date rangees, e.g., c("1926-10::1927-11","1929-08::1933-03") See the examples below.
event.color	draws the event described in event.labels in the color specified
period.color	draws the shaded region described by period.areas in the color specified
colorset	color palette to use, set by default to rational choices
pch	symbols to use, see also plot
legend.loc	places a legend into one of nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center.
ylog	TRUE/FALSE set the y-axis to logarithmic scale, similar to plot, default FALSE

cex.axis	The magnification to be used for axis annotation relative to the current setting of 'cex', same as in plot.
cex.legend	(deprecated) The magnification to be used for sizing the legend relative to the current setting of 'cex'.
cex.lab	The magnification to be used for x- and y-axis labels relative to the current setting of 'cex'.
cex.labels	The magnification to be used for event line labels relative to the current setting of 'cex'.
cex.main	The magnification to be used for the chart title relative to the current setting of 'cex'.
major.ticks	Should major tickmarks be drawn and labeled, default 'auto'
minor.ticks	Should minor tickmarks be drawn, default TRUE
grid.color	sets the color for the reference grid
grid.lty	defines the line type for the grid
xaxis.labels	Allows for non-date labeling of date axes, default is NULL
plot.engine	choose the plot engine you wish to use: ggplot2, plotly,dygraph,googlevis and default
yaxis.pct	if TRUE, scales the y axis labels by 100
space	(deprecated) default 0

Author(s)

Peter Carl

See Also

plot, par, axTicksByTime

Examples

```
# These are start and end dates, formatted as xts ranges.
## https://www.nber.org-cycles.html
cycles.dates <- c(
  "1857-06/1858-12",
  "1860-10/1861-06",
  "1865-04/1867-12",
  "1869-06/1870-12",
  "1873-10/1879-03",
  "1882-03/1885-05",
  "1887-03/1888-04",
  "1890-07/1891-05",
  "1893-01/1894-06",
  "1895-12/1897-06",
  "1899-06/1900-12",
  "1902-09/1904-08",
  "1907-05/1908-06",
```

```
  "1910-01/1912-01",
  "1913-01/1914-12",
  "1918-08/1919-03",
  "1920-01/1921-07",
  "1923-05/1924-07",
  "1926-10/1927-11",
  "1929-08/1933-03",
  "1937-05/1938-06",
  "1945-02/1945-10",
  "1948-11/1949-10",
  "1953-07/1954-05",
  "1957-08/1958-04",
  "1960-04/1961-02",
  "1969-12/1970-11",
  "1973-11/1975-03",
  "1980-01/1980-07",
  "1981-07/1982-11",
  "1990-07/1991-03",
  "2001-03/2001-11",
  "2007-12/2009-06"
)
# Event lists - FOR BEST RESULTS, KEEP THESE DATES IN ORDER
risk.dates <- c(
  "Oct 87",
  "Feb 94",
  "Jul 97",
  "Aug 98",
  "Oct 98",
  "Jul 00",
  "Sep 01"
)
risk.labels <- c(
  "Black Monday",
  "Bond Crash",
  "Asian Crisis",
  "Russian Crisis",
  "LTCM",
  "Tech Bubble",
  "Sept 11"
)
data(edhec)

R <- edhec[, "Funds of Funds", drop = FALSE]
Return.cumulative <- cumprod(1 + R) - 1
chart.TimeSeries(Return.cumulative)
chart.TimeSeries(Return.cumulative,
  colorset = "darkblue",
  legend.loc = "bottomright",
  period.areas = cycles.dates,
  period.color = rgb(204 / 255, 204 / 255, 204 / 255, alpha = 0.25),
  event.lines = risk.dates,
  event.labels = risk.labels,
  event.color = "red", lwd = 2
```

```
)
```

`chart.VaRSensitivity` *show the sensitivity of Value-at-Risk or Expected Shortfall estimates*

Description

Creates a chart of Value-at-Risk and/or Expected Shortfall estimates by confidence interval for multiple methods.

Usage

```
chart.VaRSensitivity(
  R,
 methods = c("GaussianVaR", "ModifiedVaR", "HistoricalVaR", "GaussianES", "ModifiedES",
    "HistoricalES"),
  clean = c("none", "boudt", "geltner"),
  elementcolor = "darkgray",
  reference.grid = TRUE,
  xlab = "Confidence Level",
  ylab = "Value at Risk",
  type = "l",
  lty = c(1, 2, 4),
  lwd = 1,
  colorset = (1:12),
  pch = (1:12),
  legend.loc = "bottomleft",
  cex.legend = 0.8,
  main = NULL,
  ylim = NULL,
  ...
)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`methods`	one or more calculation methods indicated "GaussianVaR", "ModifiedVaR", "HistoricalVaR", "GaussianES", "ModifiedES", "HistoricalES". See `VaR` or `ES` for more detail.
`clean`	method for data cleaning through `Return.clean`. Current options are "none" or "boudt" or "geltner".
`elementcolor`	the color used to draw chart elements. The default is "darkgray"
`reference.grid`	if true, draws a grid aligned with the points on the x and y axes
`xlab`	set the x-axis label, same as in `plot`

ylab	set the y-axis label, same as in plot
type	set the chart type, same as in plot
lty	set the line type, same as in plot
lwd	set the line width, same as in plot
colorset	color palette to use, set by default to rational choices
pch	symbols to use, see also plot
legend.loc	places a legend into one of nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center.
cex.legend	The magnification to be used for sizing the legend relative to the current setting of 'cex'.
main	set the chart title, same as in plot
ylim	set the y-axis dimensions, same as in plot
...	any other passthru parameters

Details

This chart shows estimated VaR along a series of confidence intervals for selected calculation methods. Useful for comparing a method to the historical VaR calculation.

Author(s)

Peter Carl

References

Boudt, K., Peterson, B. G., Croux, C., 2008. Estimation and Decomposition of Downside Risk for Portfolios with Non-Normal Returns. Journal of Risk, forthcoming.

See Also

VaR
ES

Examples

```
data(managers)
chart.VaRSensitivity(managers[,1,drop=FALSE],
methods=c("HistoricalVaR", "ModifiedVaR", "GaussianVaR"),
colorset=bluefocus, lwd=2)
```

`charts.PerformanceSummary` *Create combined wealth index, period performance, and drawdown chart*

Description

For a set of returns, create a wealth index chart, bars for per-period performance, and underwater chart for drawdown.

Usage

```
charts.PerformanceSummary(
  R,
  Rf = 0,
  main = NULL,
  geometric = TRUE,
  methods = "none",
  width = 0,
  event.labels = NULL,
  ylog = FALSE,
  wealth.index = FALSE,
  gap = 12,
  begin = c("first", "axis"),
  legend.loc = "topleft",
  p = 0.95,
  plot.engine = "default",
  ...
)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rf`	risk free rate, in same period as your returns
`main`	set the chart title, as in `plot`
`geometric`	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
`methods`	Used to select the risk parameter of trailing `width` returns to use in the `chart.BarVaR` panel: May be any of: • None - does not add a line, • ModifiedVaR - uses Cornish-Fisher modified VaR, • GaussianVaR - uses traditional Value at Risk, • HistoricalVaR - calculates historical Value at Risk, • ModifiedES - uses Cornish-Fisher modified Expected Shortfall, • GaussianES - uses traditional Expected Shortfall,

- HistoricalES - calculates historical Expected Shortfall,
- StdDev - per-period standard deviation

`width` number of periods to apply rolling function window over

`event.labels` TRUE/FALSE whether or not to display lines and labels for historical market shock events

`ylog` TRUE/FALSE set the y-axis to logarithmic scale, similar to `plot`, default FALSE

`wealth.index` if `wealth.index` is `TRUE`, shows the "value of $1", starting the cumulation of returns at 1 rather than zero

`gap` numeric number of periods from start of series to use to train risk calculation

`begin` Align shorter series to:

- first - prior value of the first column given for the reference or longer series or,
- axis - the initial value (1 or zero) of the axis.

passthru to `chart.CumReturns`

`legend.loc` sets the legend location in the top chart. Can be set to NULL or nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center.

`p` confidence level for calculation, default p=.95

`plot.engine` choose the plot engine you wish to use" ggplot2, plotly, and default

`...` any other passthru parameters

Note

Most inputs are the same as "`plot`" and are principally included so that some sensible defaults could be set.

Author(s)

Peter Carl

See Also

`chart.CumReturns`
`chart.BarVaR`
`chart.Drawdown`

Examples

```
data(edhec)
charts.PerformanceSummary(edhec[, c(1, 13)])
```

charts.RollingPerformance *rolling performance chart*

Description

A wrapper to create a rolling annualized returns chart, rolling annualized standard deviation chart, and a rolling annualized sharpe ratio chart.

Usage

```
charts.RollingPerformance(
  R,
  width = 12,
  Rf = 0,
  main = NULL,
  event.labels = NULL,
  legend.loc = NULL,
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
width	number of periods to apply rolling function over
Rf	risk free rate, in same period as your returns
main	set the chart title, same as in plot
event.labels	TRUE/FALSE whether or not to display lines and labels for historical market shock events
legend.loc	places a legend into one of nine locations on the chart: bottomright, bottom, bottomleft, left, topleft, top, topright, right, or center.
...	any other passthru parameters

Author(s)

Peter Carl

See Also

chart.RollingPerformance

Examples

```
data(managers)
charts.RollingPerformance(managers[,1:8],
                          Rf=managers[,10,drop=FALSE],
                          colorset=tim8equal,
                          main="Rolling 12-Month Performance",
                          legend.loc="topleft")
```

---

checkData *check input data type and format and coerce to the desired output type*

Description

This function was created to make the different kinds of data classes at least *seem* more fungible. It allows the user to pass in a data object without being concerned that the function requires a matrix, data.frame, vector, xts, or timeSeries object. By using checkData, the function "knows" what data format it has to work with.

Usage

```
checkData(
  x,
  method = c("xts", "zoo", "data.frame", "matrix", "vector"),
  na.rm = TRUE,
  quiet = TRUE,
  ...
)
```

Arguments

x	a vector, matrix, data.frame, xts, timeSeries or zoo object to be checked and coerced
method	type of coerced data object to return, one of c("xts", "zoo", "data.frame", "matrix", "vector"), default "xts"
na.rm	TRUE/FALSE Remove NA's from the data? used only with 'vector'
quiet	TRUE/FALSE if false, it will throw warnings when errors are noticed, default TRUE
...	any other passthru parameters

Author(s)

Peter Carl

Examples

```
data(edhec)
x <- checkData(edhec)
class(x)
head(x)
tail(x)
# Note that for some data types (like data.frame or matrix), passing
# in a single column without drop=FALSE loses the row and column names
# although xts objects (like edhec) preserve them by default.
x <- checkData(edhec[, 1])
class(x)
head(x)
# Include the "drop" attribute to keep row and column names
x <- checkData(edhec[, 1, drop = FALSE])
class(x)
head(x)
x <- checkData(edhec, method = "matrix")
class(x)
head(x)
x <- checkData(edhec[, 1], method = "vector")
class(x)
head(x)
```

---

checkSeedValue *Check 'seedValue' to ensure it is compatible with coredata_content attribute of 'R' (an xts object)*

Description

Check 'seedValue' to ensure it is compatible with coredata_content attribute of 'R' (an xts object)

Usage

```
checkSeedValue(R, seedValue)
```

Arguments

R	an xts object
seedValue	a numeric scalar indicating the (usually initial) index level or price of the series

Author(s)

Erol Biceroglu

`clean.boudt` *clean extreme observations in a time series to to provide more robust risk estimates*

Description

Robustly clean a time series to reduce the magnitude, but not the number or direction, of observations that exceed the $1 - \alpha\%$ risk threshold.

Usage

```
clean.boudt(R, alpha = 0.01, trim = 0.001)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`alpha`	probability to filter at 1-alpha, defaults to .01 (99%)
`trim`	where to set the "extremeness" of the Mahalanobis distance

Details

Many risk measures are calculated by using the first two (four) moments of the asset or portfolio return distribution. Portfolio moments are extremely sensitive to data spikes, and this sensitivity is only exacerbated in a multivariate context. For this reason, it seems appropriate to consider estimates of the multivariate moments that are robust to return observations that deviate extremely from the Gaussian distribution.

There are two main approaches in defining robust alternatives to estimate the multivariate moments by their sample means (see e.g. Maronna[2006]). One approach is to consider a more robust estimator than the sample means. Another one is to first clean (in a robust way) the data and then take the sample means and moments of the cleaned data.

Our cleaning method follows the second approach. It is designed in such a way that, if we want to estimate downside risk with loss probability α, it will never clean observations that belong to the $1 - \alpha$ least extreme observations. Suppose we have an n-dimensional vector time series of length T: $r_1, ..., r_T$. We clean this time series in three steps.

1. *Ranking the observations in function of their extremeness.* Denote μ and Σ the mean and covariance matrix of the bulk of the data and let $\lfloor \cdot \rfloor$ be the operator that takes the integer part of its argument. As a measure of the extremeness of the return observation r_t, we use its squared Mahalanobis distance $d_t^2 = (r_t - \mu)' \Sigma^{-1} (r_t - \mu)$. We follow Rousseeuw(1985) by estimating μ and Σ as the mean vector and covariance matrix (corrected to ensure consistency) of the subset of size $\lfloor (1 - \alpha)T \rfloor$ for which the determinant of the covariance matrix of the elements in that subset is the smallest. These estimates will be robust against the α most extreme returns. Let $d_{(1)}^2, ..., d_{(T)}^2$ be the ordered sequence of the estimated squared Mahalanobis distances such that $d_{(i)}^2 \leq d_{(i+1)}^2$.

2. *Outlier identification.* Return observations are qualified as outliers if their estimated squared Mahalanobis distance d_t^2 is greater than the empirical $1-\alpha$ quantile $d^2_{(\lfloor(1-\alpha)T\rfloor)}$ and exceeds a very extreme quantile of the Chi squared distribution function with n degrees of freedom, which is the distribution function of d_t^2 when the returns are normally distributed. In this application we take the 99.9% quantile, denoted $\chi^2_{n,0.999}$.
3. *Data cleaning.* Similarly to Khan(2007) we only clean the returns that are identified as outliers in step 2 by replacing these returns r_t with

$$r_t\sqrt{\frac{\max(d^2_{(\lfloor(1-\alpha)T)\rfloor},\chi^2_{n,0.999})}{d_t^2}}$$

The cleaned return vector has the same orientation as the original return vector, but its magnitude is smaller. Khan(2007) calls this procedure of limiting the value of d_t^2 to a quantile of the χ_n^2 distribution, "multivariate Winsorization'.

Note that the primary value of data cleaning lies in creating a more robust and stable estimation of the distribution describing the large majority of the return data. The increased robustness and stability of the estimated moments utilizing cleaned data should be used for portfolio construction. If a portfolio manager wishes to have a more conservative risk estimate, cleaning may not be indicated for risk monitoring. It is also important to note that the robust method proposed here does not remove data from the series, but only decreases the magnitude of the extreme events. It may also be appropriate in practice to use a cleaning threshold somewhat outside the VaR threshold that the manager wishes to consider. In actual practice, it is probably best to back-test the results of both cleaned and uncleaned series to see what works best with the particular combination of assets under consideration.

Value

cleaned data matrix

Note

This function and much of this text was originally written for Boudt, et. al, 2008

Author(s)

Kris Boudt, Brian G. Peterson

References

Boudt, K., Peterson, B. G., Croux, C., 2008. Estimation and Decomposition of Downside Risk for Portfolios with Non-Normal Returns. Journal of Risk, forthcoming.

Khan, J. A., S. Van Aelst, and R. H. Zamar (2007). Robust linear model selection based on least angle regression. Journal of the American Statistical Association 102.

Maronna, R. A., D. R. Martin, and V. J. Yohai (2006). Robust Statistics: Theory and Methods. Wiley.

Rousseeuw, P. J. (1985). Multivariate estimation with high breakdown point. In W. Grossmann, G. Pflug, I. Vincze, and W. Wertz (Eds.), Mathematical Statistics and Its Applications, Volume B, pp. 283?297. Dordrecht-Reidel.

See Also

Return.clean

CoMoments *Functions for calculating comoments of financial time series*

Description

calculates coskewness and cokurtosis as the skewness and kurtosis of two assets with reference to one another.

Usage

```
CoSkewnessMatrix(R, ...)

CoKurtosisMatrix(R, ...)

CoVariance(Ra, Rb)

CoSkewness(Ra, Rb)

CoKurtosis(Ra, Rb)

M3.MM(R, unbiased = FALSE, as.mat = TRUE, ...)

M4.MM(R, as.mat = TRUE, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters
Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	an xts, vector, matrix, data frame, timeSeries or zoo object of index, benchmark, portfolio, or secondary asset returns to compare against
unbiased	TRUE/FALSE whether to use a correction to have an unbiased estimator, default FALSE
as.mat	TRUE/FALSE whether to return the full moment matrix or only the vector with the unique elements (the latter is advised for speed), default TRUE

Details

Ranaldo and Favre (2005) define coskewness and cokurtosis as the skewness and kurtosis of a given asset analysed with the skewness and kurtosis of the reference asset or portfolio. Adding an asset to a portfolio, such as a hedge fund with a significant level of coskewness to the portfolio, can increase or decrease the resulting portfolio's skewness. Similarly, adding a hedge fund with a positive cokurtosis coefficient will add kurtosis to the portfolio.

The co-moments are useful for measuring the marginal contribution of each asset to the portfolio's resulting risk. As such, comoments of asset return distribution should be useful as inputs for portfolio optimization in addition to the covariance matrix. Martellini and Ziemann (2007) point out that the problem of portfolio selection becomes one of selecting tangency points in four dimensions, incorporating expected return, second, third and fourth centered moments of asset returns.

Even outside of the optimization problem, measuring the co-moments should be a useful tool for evaluating whether or not an asset is likely to provide diversification potential to a portfolio, not only in terms of normal risk (i.e. volatility) but also the risk of assymetry (skewness) and extreme events (kurtosis).

Author(s)

Kris Boudt, Peter Carl, Dries Cornilly, Brian Peterson

References

Boudt, Kris, Brian G. Peterson, and Christophe Croux. 2008. Estimation and Decomposition of Downside Risk for Portfolios with Non-Normal Returns. Journal of Risk. Winter.

Boudt, Kris, Dries Cornilly and Tim Verdonck. 2020 A Coskewness Shrinkage Approach for Estimating the Skewness of Linear Combinations of Random Variables. Journal of Financial Econometrics, 18(1), 1-23.

Martellini, Lionel, and Volker Ziemann. 2010. Improved estimates of higher-order comoments and implications for portfolio selection. Review of Financial Studies, 23(4), 1467-1502.

Ranaldo, Angelo, and Laurent Favre Sr. 2005. How to Price Hedge Funds: From Two- to Four-Moment CAPM. SSRN eLibrary.

Scott, Robert C., and Philip A. Horvath. 1980. On the Direction of Preference for Moments of Higher Order than the Variance. Journal of Finance 35(4):915-919.

See Also

BetaCoSkewness
BetaCoKurtosis
BetaCoMoments
ShrinkageMoments
EWMAMoments
StructuredMoments
MCA
NCE

Examples

```
data(managers)
CoVariance(managers[, "HAM2", drop=FALSE], managers[, "SP500 TR", drop=FALSE])
CoSkewness(managers[, "HAM2", drop=FALSE], managers[, "SP500 TR", drop=FALSE])
CoKurtosis(managers[, "HAM2", drop=FALSE], managers[, "SP500 TR", drop=FALSE])
```

DownsideDeviation *downside risk (deviation, variance) of the return distribution*

Description

Downside deviation, semideviation, and semivariance are measures of downside risk.

Usage

```
DownsideDeviation(
  R,
  MAR = 0,
  method = c("full", "subset"),
  ...,
  potential = FALSE,
  SE = FALSE,
  SE.control = NULL
)

DownsidePotential(R, MAR = 0)

SemiDeviation(R, SE = FALSE, SE.control = NULL, ...)

SemiSD(R, SE = FALSE, SE.control = NULL, ...)

SemiVariance(R)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
MAR	Minimum Acceptable Return, in the same periodicity as your returns
method	one of "full" or "subset", indicating whether to use the length of the full series or the length of the subset of the series below the MAR as the denominator, defaults to "full"
...	any other passthru parameters
potential	if TRUE, calculate downside potential instead, default FALSE
SE	TRUE/FALSE whether to ouput the standard errors of the estimates of the risk measures, default FALSE. Only available for SemiDeviation and SemiSD
SE.control	Control parameters for the computation of standard errors. Should be done using the RPESE.control function.

Details

Downside deviation, similar to semi deviation, eliminates positive returns when calculating risk. Instead of using the mean return or zero, it uses the Minimum Acceptable Return as proposed by

Sharpe (which may be the mean historical return or zero). It measures the variability of underperformance below a minimum targer rate. The downside variance is the square of the downside potential.

To calculate it, we take the subset of returns that are less than the target (or Minimum Acceptable Returns (MAR)) returns and take the differences of those to the target. We sum the squares and divide by the total number of returns to get a below-target semi-variance.

$$DownsideDeviation(R, MAR) = \delta_{MAR} = \sqrt{\sum_{t=1}^{n} \frac{min[(R_t - MAR), 0]^2}{n}}$$

$$DownsideVariance(R, MAR) = \sum_{t=1}^{n} \frac{min[(R_t - MAR), 0]^2}{n}$$

$$DownsidePotential(R, MAR) = \sum_{t=1}^{n} \frac{min[(R_t - MAR), 0]}{n}$$

where n is either the number of observations of the entire series or the number of observations in the subset of the series falling below the MAR.

SemiDeviation or SemiVariance is a popular alternative downside risk measure that may be used in place of standard deviation or variance. SemiDeviation and SemiVariance are implemented as a wrapper of DownsideDeviation with MAR=mean(R).

In many functions like Markowitz optimization, semideviation may be substituted directly, and the covariance matrix may be constructed from semideviation or the vector of returns below the mean rather than from variance or the full vector of returns.

In semideviation, by convention, the value of n is set to the full number of observations. In semivariance the the value of n is set to the subset of returns below the mean. It should be noted that while this is the correct mathematical definition of semivariance, this result doesn't make any sense if you are also going to be using the time series of returns below the mean or below a MAR to construct a semi-covariance matrix for portfolio optimization.

Sortino recommends calculating downside deviation utilizing a continuous fitted distribution rather than the discrete distribution of observations. This would have significant utility, especially in cases of a small number of observations. He recommends using a lognormal distribution, or a fitted distribution based on a relevant style index, to construct the returns below the MAR to increase the confidence in the final result. Hopefully, in the future, we'll add a fitted option to this function, and would be happy to accept a contribution of this nature.

Author(s)

Peter Carl, Brian G. Peterson, Matthieu Lestel

References

Sortino, F. and Price, L. Performance Measurement in a Downside Risk Framework. *Journal of Investing*. Fall 1994, 59-65.
Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008

Plantinga, A., van der Meer, R. and Sortino, F. The Impact of Downside Risk on Risk-Adjusted Performance of Mutual Funds in the Euronext Markets. July 19, 2001. Available at SSRN: https://www.ssrn.com/abstract=277352

https://www.sortino.com/htm/performance.htm see especially end note 10

https://en.wikipedia.org/wiki/Semivariance

Examples

```
#with data used in Bacon 2008

data(portfolio_bacon)
MAR = 0.005
DownsideDeviation(portfolio_bacon[,1], MAR) #expected 0.0255
DownsidePotential(portfolio_bacon[,1], MAR) #expected 0.0137

#with data of managers

data(managers)
apply(managers[,1:6], 2, sd, na.rm=TRUE)
DownsideDeviation(managers[,1:6])  # MAR 0%
DownsideDeviation(managers[,1:6], MAR = .04/12) #MAR 4%
SemiDeviation(managers[,1,drop=FALSE])
SemiDeviation(managers[,1:6])
SemiVariance (managers[,1,drop=FALSE])
SemiVariance (managers[,1:6]) #calculated using method="subset"
```

DownsideFrequency *downside frequency of the return distribution*

Description

To calculate Downside Frequency, we take the subset of returns that are less than the target (or Minimum Acceptable Returns (MAR)) returns and divide the length of this subset by the total number of returns.

Usage

```
DownsideFrequency(R, MAR = 0, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
MAR	Minimum Acceptable Return, in the same periodicity as your returns
...	any other passthru parameters

Details

$$DownsideFrequency(R, MAR) = \sum_{t=1}^{n} \frac{min[(R_t - MAR), 0]}{R_t * n}$$

where n is the number of observations of the entire series

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.94

Examples

```
data(portfolio_bacon)
MAR = 0.005
print(DownsideFrequency(portfolio_bacon[,1], MAR)) #expected 0.458

data(managers)
print(DownsideFrequency(managers['1996']))
print(DownsideFrequency(managers['1996',1])) #expected 0.25
```

DownsideSharpeRatio *Downside Sharpe Ratio*

Description

DownsideSharpeRatio computation with standard errors

Usage

```
DownsideSharpeRatio(R, rf = 0, SE = FALSE, SE.control = NULL, ...)
```

Arguments

R	Data of returns for one or multiple assets or portfolios.
rf	Risk-free interest rate.
SE	TRUE/FALSE whether to ouput the standard errors of the estimates of the risk measures, default FALSE.
SE.control	Control parameters for the computation of standard errors. Should be done using the RPESE.control function.
...	Additional parameters.

Details

The Downside Sharpe Ratio (DSR) is a short name for what Ziemba (2005) called the "Symmetric Downside Risk Sharpe Ratio" and is defined as the ratio of the mean excess return to the square root of lower semivariance:

$$\frac{\overline{(R_a - R_f)}}{\sqrt{2} SemiSD(R_a)}$$

.

Value

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

Author(s)

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

References

Ziemba, W. T. (2005). The symmetric downside-risk Sharpe ratio. The Journal of Portfolio Management, 32(1), 108-122.

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")
# Compute Rachev ratio for managers data
DownsideSharpeRatio(edhec)
```

`DRatio` *d ratio of the return distribution*

Description

The d ratio is similar to the Bernado Ledoit ratio but inverted and taking into account the frequency of positive and negative returns.

Usage

```
DRatio(R, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters

Details

It has values between zero and infinity. It can be used to rank the performance of portfolios. The lower the d ratio the better the performance, a value of zero indicating there are no returns less than zero and a value of infinity indicating there are no returns greater than zero.

$$DRatio(R) = \frac{n_d * \sum_{t=1}^{n} max(-R_t, 0)}{n_u * \sum_{t=1}^{n} max(R_t, 0)}$$

where n is the number of observations of the entire series, n_d is the number of observations less than zero, n_u is the number of observations greater than zero

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.95

Examples

```
data(portfolio_bacon)
print(DRatio(portfolio_bacon[,1])) #expected 0.401

data(managers)
print(DRatio(managers['1996']))
print(DRatio(managers['1996',1])) #expected 0.0725
```

DrawdownDeviation *Calculates a standard deviation-type statistic using individual drawdowns.*

Description

DD = sqrt(sum[j=1,2,...,d](D_j^2/n)) where D_j = jth drawdown over the entire period d = total number of drawdowns in entire period n = number of observations

Usage

```
DrawdownDeviation(R, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters

DrawdownPeak *Drawdawn peak of the return distribution*

Description

Drawdawn peak is for each return its drawdown since the previous peak

Usage

```
DrawdownPeak(R, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns (Note: expects decimal returns, e.g. 0.05 for 5%, not percentage units).
...	any other passthru parameters

Author(s)

Matthieu Lestel

Drawdowns *Find the drawdowns and drawdown levels in a timeseries.*

Description

`findDrawdowns` will find the starting period, the ending period, and the amount and length of the drawdown.

Usage

```
Drawdowns(R, geometric = TRUE, ...)

findDrawdowns(R, geometric = TRUE, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
...	any other passthru parameters

Details

Often used with `sortDrawdowns` to get the largest drawdowns.

`Drawdowns` will calculate the drawdown levels as percentages, for use in `chart.Drawdown`.

Returns an unordered list:

- return depth of drawdown
- from starting period
- to ending period
- length length in periods

Author(s)

Peter Carl

`findDrawdowns` modified with permission from function by Sankalp Upadhyay

References

Bacon, C. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 3rd Edition. 2023. p. 194-196

See Also

`sortDrawdowns`
`maxDrawdown`
`sortDrawdowns`
`table.Drawdowns`
`table.DownsideRisk`
`chart.Drawdown`

Examples

```
data(edhec)
findDrawdowns(edhec[, "Funds of Funds", drop = FALSE])
sortDrawdowns(findDrawdowns(edhec[, "Funds of Funds", drop = FALSE]))
```

edhec *EDHEC-Risk Hedge Fund Style Indices*

Description

EDHEC composite hedge fund style index returns.

Usage

```
data(edhec)
```

Format

CSV loaded into R and saved as an xts object with monthly observations. NOTE: In the era of CoVid-19, a few observations in the 'Short Selling' index have not been reported. We chose to zero fill them at this time. These are observations on 2020-04-30, 2020-05-31, 2020-11-30, 2020-12-31, 2021-01-31, 2021-04-30, and 2021-05-31.

Details

EDHEC Data used in PerformanceAnalytics and related publications with the kind permission of the EDHEC Risk and Asset Management Research Center.

The 'edhec' data set included with PerformanceAnalytics will be periodically updated (typically annually) to include additional observations. If you intend to use this data set in automated tests, please be sure to subset your data like `edhec[1:120,]` to use the first ten years of observations.

From the EDHEC website: "The EDHEC Risk and Asset Management Research Centre plays a noted role in furthering applied financial research and systematically highlighting its practical uses. As part of its philosophy, the centre maintains a dialogue with professionals which benefits the industry as a whole. At the same time, its proprietary R&D provides sponsors with an edge over competition and joint ventures allow selected partners to develop new business opportunities.

To further assist financial institutions and investors implement the latest research advances in order to meet the challenges of the changing asset management landscape, the centre has spawned two consultancies and an executive education arm. Clients of these derivative activities include many of the leading organisations throughout Europe."

Source

https://climateinstitute.edhec.edu/

References

About EDHEC Alternative Indexes. December 16, 2003. EDHEC-Risk.

Vaissie Mathieu. A Detailed Analysis of the Construction Methods and Management Principles of Hedge Fund Indices. October 2003. EDHEC.

Examples

```
data(edhec)

#preview the data
head(edhec)

#summary period statistics
summary(edhec)

#cumulative index returns
tail(cumprod(1+edhec),1)
```

ETL *calculates Expected Shortfall(ES) (or Conditional Value-at-Risk(CVaR) for univariate and component, using a variety of analytical methods.*

Description

Calculates Expected Shortfall(ES) (also known as) Conditional Value at Risk(CVaR) or Expected Tail Loss (ETL) for univariate, component, and marginal cases using a variety of analytical methods.

Usage

```
ETL(
  R = NULL,
  p = 0.95,
  ...,
  method = c("modified", "gaussian", "historical"),
  clean = c("none", "boudt", "geltner", "locScaleRob"),
  portfolio_method = c("single", "component"),
  weights = NULL,
  mu = NULL,
  sigma = NULL,
  m3 = NULL,
  m4 = NULL,
  invert = TRUE,
  operational = TRUE,
  SE = FALSE,
  SE.control = NULL,
  p.tr = 0.97,
  init = c(1, 0.3),
  nsim = 10000
)
```

Arguments

R	a vector, matrix, data frame, timeSeries or zoo object of asset returns
p	confidence level for calculation, default p=.95
...	any other passthru parameters
method	one of "modified","gaussian","historical", "kernel", "gpd", "lognormal", "montecarlo", see Details.
clean	method for data cleaning through Return.clean. Current options are "none", "boudt", "geltner", or "locScaleRob".
portfolio_method	one of "single","component","marginal" defining whether to do univariate, component, or marginal calc, see Details.
weights	portfolio weighting vector, default NULL, see Details
mu	If univariate, mu is the mean of the series. Otherwise mu is the vector of means of the return series, default NULL, see Details
sigma	If univariate, sigma is the variance of the series. Otherwise sigma is the covariance matrix of the return series, default NULL, see Details
m3	If univariate, m3 is the skewness of the series. Otherwise m3 is the coskewness matrix (or vector with unique coskewness values) of the returns series, default NULL, see Details
m4	If univariate, m4 is the excess kurtosis of the series. Otherwise m4 is the cokurtosis matrix (or vector with unique cokurtosis values) of the return series, default NULL, see Details
invert	TRUE/FALSE whether to invert the VaR measure, see Details.
operational	TRUE/FALSE, default TRUE, see Details.
SE	TRUE/FALSE whether to ouput the standard errors of the estimates of the risk measures, default FALSE.
SE.control	Control parameters for the computation of standard errors. Should be done using the RPESE.control function.
p.tr	probability threshold for method="gpd", default 0.97
init	initial parameter estimate for method="gpd", default c(1.00, 0.3)
nsim	the number of simulations used for method="montecarlo", default 10000

Value

ES measures are evaluated and returned based on the portfolio_method and SE arguments:

- portfolio_method = "single": Returns a scalar or matrix (depending on the number of asset columns) of ES estimates.
- portfolio_method = "component": Returns a list with three components: the univariate portfolio ES, the scalar contribution of each component to the portfolio ES, and a percentage risk contribution.

If SE = TRUE, the output will also include standard errors or confidence intervals:

- **Standard Methods (`"modified"`, `"gaussian"`, `"historical"`):** Leverages the `RPESE` package to return standard error estimates, appending rows such as `se` or `IFiid` to the ES matrix.
- **Extreme Value Theory (`method = "gpd"`):** Uses Profile Log-Likelihood ratio tests to calculate asymmetric confidence bounds, appending rows for the Lower Confidence Limit (`LCL`) and Upper Confidence Limit (`UCL`).

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.

Univariate estimation of ES

The ES at a probability level p (e.g. 95%) is the negative value of the expected value of the return when the return is less than its $c = 1-p$ quantile. In a set of returns for which sufficently long history exists, the per-period ES can be estimated by the negative value of the sample average of all returns below the quantile. This method is also sometimes called "historical ES", as it is by definition *ex post* analysis of the return distribution, and may be accessed with `method="historical"`.

When you don't have a sufficiently long set of returns to use non-parametric or historical ES, or wish to more closely model an ideal distribution, it is common to us a parmetric estimate based on the distribution. Parametric ES does a better job of accounting for the tails of the distribution by more precisely estimating shape of the distribution tails of the risk quantile. The most common estimate is a normal (or Gaussian) distribution $R \sim N(\mu, \sigma)$ for the return series. In this case, estimation of ES requires the mean return $\bar{R}$, the return distribution and the variance of the returns σ. In the most common case, parametric VaR is thus calculated by

$$\sigma = variance(R)$$

$$ES = -\bar{R} + \sqrt{\sigma} \cdot \frac{1}{c}\phi(z_c)$$

where z_c is the c-quantile of the standard normal distribution. Represented in R by `qnorm(c)`, and may be accessed with `method="gaussian"`. The function ϕ is the Gaussian density function.

Other forms of parametric estimation utilize a different distribution for the distribution of losses to better account for the possible fat-tailed nature of downside risk. The now-archived package `VaR`

by Talgat Daniyarov contained methods for simulating and estimating lognormal and generalized Pareto distributions. You may access the lognormal estimate using `method="lognormal"`. A Generalized Pareto Distribution estimate is available using `method="gpd"`. A Monte Carlo simulation estimate is available using `method="montecarlo"`.

The limitations of Gaussian ES are well covered in the literature, since most financial return series are non-normal. Boudt, Peterson and Croux (2008) provide a modified ES calculation that takes the higher moments of non-normal distributions (skewness, kurtosis) into account through the use of a Cornish-Fisher expansion, and collapses to standard (traditional) Gaussian ES if the return stream follows a standard distribution. More precisely, for a loss probability c, modified ES is defined as the negative of the expected value of all returns below the c Cornish-Fisher quantile and where the expectation is computed under the second order Edgeworth expansion of the true distribution function.

Modified expected shortfall should always be larger than modified Value at Risk. Due to estimation problems, this might not always be the case. Set Operational = TRUE to replace modified ES with modified VaR in the (exceptional) case where the modified ES is smaller than modified VaR.

Component ES

By setting `portfolio_method="component"` you may calculate the ES contribution of each element of the portfolio. The return from the function in this case will be a list with three components: the univariate portfolio ES, the scalar contribution of each component to the portfolio ES (these will sum to the portfolio ES), and a percentage risk contribution (which will sum to 100%).

Both the numerical and percentage component contributions to ES may contain both positive and negative contributions. A negative contribution to Component ES indicates a portfolio risk diversifier. Increasing the position weight will reduce overall portoflio ES.

If a weighting vector is not passed in via `weights`, the function will assume an equal weighted (neutral) portfolio.

Multiple risk decomposition approaches have been suggested in the literature. A naive approach is to set the risk contribution equal to the stand-alone risk. This approach is overly simplistic and neglects important diversification effects of the units being exposed differently to the underlying risk factors. An alternative approach is to measure the ES contribution as the weight of the position in the portfolio times the partial derivative of the portfolio ES with respect to the component weight.

$$C_i\text{ES} = w_i \frac{\partial \text{ES}}{\partial w_i}.$$

Because the portfolio ES is linear in position size, we have that by Euler's theorem the portfolio VaR is the sum of these risk contributions. Scaillet (2002) shows that for ES, this mathematical decomposition of portfolio risk has a financial meaning. It equals the negative value of the asset's expected contribution to the portfolio return when the portfolio return is less or equal to the negative portfolio VaR:

$$C_i\text{ES} == -E\left[w_i r_i | r_p \leq -\text{VaR}\right]$$

For the decomposition of Gaussian ES, the estimated mean and covariance matrix are needed. For the decomposition of modified ES, also estimates of the coskewness and cokurtosis matrices are needed. If r denotes the $Nx1$ return vector and mu is the mean vector, then the $N \times N^2$ coskewness matrix is

$$m3 = E\left[(r-\mu)(r-\mu)' \otimes (r-\mu)'\right]$$

The $N \times N^3$ co-kurtosis matrix is

$$m_4 = E\left[(r-\mu)(r-\mu)' \otimes (r-\mu)' \otimes (r-\mu)'\right]$$

where $\otimes$ stands for the Kronecker product. The matrices can be estimated through the functions `skewness.MM` and `kurtosis.MM`. More efficient estimators were proposed by Martellini and Ziemann (2007) and will be implemented in the future.

As discussed among others in Cont, Deguest and Scandolo (2007), it is important that the estimation of the ES measure is robust to single outliers. This is especially the case for modified VaR and its decomposition, since they use higher order moments. By default, the portfolio moments are estimated by their sample counterparts. If `clean="boudt"` then the $1-p$ most extreme observations are winsorized if they are detected as being outliers. For more information, see Boudt, Peterson and Croux (2008) and `Return.clean`. If your data consist of returns for highly illiquid assets, then `clean="geltner"` may be more appropriate to reduce distortion caused by autocorrelation, see `Return.Geltner` for details.

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)

Brian G. Peterson and Kris Boudt, Talgat Daniyarov

References

Daniyarov, T. *VaR: Value at Risk estimation*. CRAN Archive. 2004. https://cran.r-project.org/src/contrib/Archive/VaR/

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, L. and Ziemann, V., 2010. Improved estimates of higher-order comoments and implications for portfolio selection. Review of Financial Studies, 23(4):1467-1502.

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.

See Also

VaR
SharpeRatio
chart.VaRSensitivity
Return.clean

Examples

```
data(edhec)

# first do normal ES calc
ES(edhec, p = .95, method = "historical")

# now use Gaussian
ES(edhec, p = .95, method = "gaussian")

# now use modified Cornish Fisher calc to take non-normal distribution into account
ES(edhec, p = .95, method = "modified")

# now use p=.99
ES(edhec, p = .99)
# or the equivalent alpha=.01
ES(edhec, p = .01)

# CRAN (questionably(ahem) requires these methods to not run if you don't have Suggests loaded)
if (requireNamespace("robustbase", quietly = TRUE)) {
  #        now with outliers squished
  ES(edhec, clean = "boudt")

  # add Component ES for the equal weighted portfolio
  ES(edhec, clean = "boudt", portfolio_method = "component")
} # end CRAN workaround


# end CRAN Windows check
```

EWMAMoments *Functions for calculating EWMA comoments of financial time series*

Description

calculates exponentially weighted moving average covariance, coskewness and cokurtosis matrices

Usage

```
M2.ewma(R, lambda = 0.97, last.M2 = NULL, ...)
```

```
M3.ewma(R, lambda = 0.97, last.M3 = NULL, as.mat = TRUE, ...)

M4.ewma(R, lambda = 0.97, last.M4 = NULL, as.mat = TRUE, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns (with mean zero)
lambda	decay coefficient
last.M2	last estimated covariance matrix before the observed returns R
...	any other passthru parameters
last.M3	last estimated coskewness matrix before the observed returns R
as.mat	TRUE/FALSE whether to return the full moment matrix or only the vector with the unique elements (the latter is advised for speed), default TRUE
last.M4	last estimated cokurtosis matrix before the observed returns R

Details

The coskewness and cokurtosis matrices are defined as the matrices of dimension p x p^2 and p x p^3 containing the third and fourth order central moments. They are useful for measuring nonlinear dependence between different assets of the portfolio and computing modified VaR and modified ES of a portfolio.

EWMA estimation of the covariance matrix was popularized by the RiskMetrics report in 1996. The M3.ewma and M4.ewma are straightforward extensions to the setting of third and fourth order central moments

Author(s)

Dries Cornilly

References

JP Morgan. Riskmetrics technical document. 1996.

See Also

CoMoments
ShrinkageMoments
StructuredMoments
MCA
NCE

Examples

```
data(edhec)

# EWMA estimation
# 'as.mat = F' would speed up calculations in higher dimensions
```

```
sigma <- M2.ewma(edhec, 0.94)
m3 <- M3.ewma(edhec, 0.94)
m4 <- M4.ewma(edhec, 0.94)

# compute equal-weighted portfolio modified ES
mu <- colMeans(edhec)
p <- length(mu)
ES(p = 0.95, portfolio_method = "component", weights = rep(1 / p, p), mu = mu,
    sigma = sigma, m3 = m3, m4 = m4)

# compare to sample method
sigma <- cov(edhec)
m3 <- M3.MM(edhec)
m4 <- M4.MM(edhec)
ES(p = 0.95, portfolio_method = "component", weights = rep(1 / p, p), mu = mu,
    sigma = sigma, m3 = m3, m4 = m4)
```

`FamaBeta` *Fama beta of the return distribution*

Description

Fama beta is a beta used to calculate the loss of diversification. It is made so that the systematic risk is equivalent to the total portfolio risk.

Usage

```
FamaBeta(Ra, Rb, ...)
```

Arguments

`Ra`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rb`	return vector of the benchmark asset
`...`	any other passthru parameters

Details

$$\beta_F = \frac{\sigma_P}{\sigma_M}$$

where σ_P is the portfolio standard deviation and σ_M is the market risk

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.78

Examples

```
data(portfolio_bacon)
print(FamaBeta(portfolio_bacon[,1], portfolio_bacon[,2])) #expected 1.03

data(managers)
print(FamaBeta(managers['1996',1], managers['1996',8]))
print(FamaBeta(managers['1996',1:5], managers['1996',8]))
```

Frequency *Frequency of the return distribution*

Description

Gives the period of the return distribution (ie 12 if monthly return, 4 if quarterly return)

Usage

```
Frequency(R, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters

Author(s)

Matthieu Lestel

Examples

```
data(portfolio_bacon)
print(Frequency(portfolio_bacon[,1])) #expected 12
data(managers)
print(Frequency(managers['1996',1:5]))
```

HurstIndex *calculate the Hurst Index The Hurst index can be used to measure whether returns are mean reverting, totally random, or persistent.*

Description

Hurst obtained a dimensionless statistical exponent by dividing the range of the cumulative sum of deviations from the mean by the standard deviation of the observations, so this approach is commonly referred to as rescaled range (R/S) analysis.

Usage

```
HurstIndex(R, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters

Details

$$H = log(m)/log(n)$$

where $m = [max(z_i) - min(z_i)]/\sigma_r$, $z_i = \sum_{i=1}^{n}(r_i - \bar{r})$, and $n = numberofobservations$ A Hurst index between 0.5 and 1 suggests that the returns are persistent. At 0.5, the index suggests returns are totally random. Between 0 and 0.5 it suggests that the returns are mean reverting.

H.E. Hurst originally developed the Hurst index to help establish optimal water storage along the Nile. Nile floods are extremely persistent, measuring a Hurst index of 0.9. Peters (1991) notes that Equity markets have a Hurst index in excess of 0.5, with typical values of around 0.7. That appears to be anomalous in the context of the mainstream 'rational behaviour' theories of economics, and suggests existence of a powerful 'long-term memory' causal dependence. Clarkson (2001) suggests that an 'over-reaction bias' could be expected to generate a powerful 'long-term memory' effect in share prices.

References

Clarkson, R. (2001) FARM: a financial actuarial risk model. In Chapter 12 of Managing Downside Risk in Financial Markets, ed. Sortino, F. and Satchel, S. Woburn MA. Butterworth-Heinemann Finance.

Peters, E.E (1991) Chaos and Order in Capital Markets, New York: Wiley.

Bacon, Carl. (2008) Practical Portfolio Performance Measurement and Attribution, 2nd Edition. London: John Wiley & Sons.

See Also

hurstexp for more robust estimators of the Hurst exponent (e.g., corrected R/S, empirical, and theoretical).

`InformationRatio` *InformationRatio = ActivePremium/TrackingError*

Description

The Active Premium divided by the Tracking Error.

Usage

```
InformationRatio(Ra, Rb, scale = NA, ...)
```

Arguments

`Ra`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rb`	return vector of the benchmark asset
`scale`	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
`...`	any other passthru parameters to ActivePremium and Return.annualized (e.g., `geometric=FALSE`)

Details

InformationRatio = ActivePremium/TrackingError

This relates the degree to which an investment has beaten the benchmark to the consistency with which the investment has beaten the benchmark.

Note

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

Author(s)

Peter Carl

References

Sharpe, W.F. The Sharpe Ratio,*Journal of Portfolio Management*,Fall 1994, 49-58.

See Also

TrackingError
ActivePremium
SharpeRatio

Examples

```
data(managers)
InformationRatio(managers[,"HAM1",drop=FALSE], managers[, "SP500 TR", drop=FALSE])
InformationRatio(managers[,1:6], managers[,8,drop=FALSE])
InformationRatio(managers[,1:6], managers[,8:7])
```

Kappa *Kappa of the return distribution*

Description

Introduced by Kaplan and Knowles (2004), Kappa is a generalized downside risk-adjusted performance measure.

Usage

```
Kappa(R, MAR, l, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
MAR	Minimum Acceptable Return, in the same periodicity as your returns
l	the coefficient of the Kappa
...	any other passthru parameters

Details

To calculate it, we take the difference of the mean of the distribution to the target and we divide it by the l-root of the lth lower partial moment. To calculate the lth lower partial moment we take the subset of returns below the target and we sum the differences of the target to these returns. We then return return this sum divided by the length of the whole distribution.

$$Kappa(R, MAR, l) = \frac{r_p - MAR}{\sqrt[l]{\frac{1}{n} * \sum_{t=1}^{n} max(MAR - R_t, 0)^l}}$$

For l=1 kappa is the Sharpe-omega ratio and for l=2 kappa is the sortino ratio.

Kappa should only be used to rank portfolios as it is difficult to interpret the absolute differences between kappas. The higher the kappa is, the better.

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.96

Examples

```
l = 2

data(portfolio_bacon)
MAR = 0.005
print(Kappa(portfolio_bacon[,1], MAR, l)) #expected 0.157

data(managers)
MAR = 0
print(Kappa(managers['1996'], MAR, l))
print(Kappa(managers['1996',1], MAR, l)) #expected 1.493
```

`KellyRatio` *calculate Kelly criterion ratio (leverage or bet size) for a strategy*

Description

Kelly criterion ratio (leverage or bet size) for a strategy.

Usage

```
KellyRatio(R, Rf = 0, method = "half")
```

Arguments

`R`	a vector of returns to perform a mean over
`Rf`	risk free rate, in same period as your returns
`method`	method=half will use the half-Kelly, this is the default

Details

The Kelly Criterion was identified by Bell Labs scientist John Kelly, and applied to blackjack and stock strategy sizing by Ed Thorpe.

The Kelly ratio can be simply stated as: "bet size is the ratio of edge over odds." Mathematically, you are maximizing log-utility. As such, the Kelly criterion is equal to the expected excess return of the strategy divided by the expected variance of the excess return, or

$$leverage = \frac{(\overline{R}_s - R_f)}{StdDev(R)^2}$$

As a performance metric, the Kelly Ratio is calculated retrospectively on a particular investment as a measure of the edge that investment has over the risk free rate. It may be use as a stack ranking method to compare investments in a manner similar to the various ratios related to the Sharpe ratio.

Author(s)

Brian G. Peterson

References

Thorp, Edward O. (1997; revised 1998). The Kelly Criterion in Blackjack, Sports Betting, and the Stock Market. #https://archive.ph/h1LuS no longer online, but can be found at archive.ph /h1LuS

Examples

```
data(managers)
KellyRatio(managers[,1,drop=FALSE], Rf=.04/12)
KellyRatio(managers[,1,drop=FALSE], Rf=managers[,10,drop=FALSE])
KellyRatio(managers[,1:6], Rf=managers[,10,drop=FALSE])
```

---

kurtosis *Kurtosis*

Description

compute kurtosis of a univariate distribution

Usage

```
kurtosis(
  x,
  na.rm = FALSE,
  method = c("excess", "moment", "fisher", "sample", "sample_excess"),
  ...
)
```

Arguments

x	a numeric vector or object.
na.rm	a logical. Should missing values be removed?
method	a character string which specifies the method of computation. These are either "moment", "fisher", or "excess". If "excess" is selected, then the value of the kurtosis is computed by the "moment" method and a value of 3 will be subtracted. The "moment" method is based on the definitions of kurtosis for distributions; these forms should be used when resampling (bootstrap or jackknife). The "fisher" method correspond to the usual "unbiased" definition of sample variance, although in the case of kurtosis exact unbiasedness is not possible. The "sample" method gives the sample kurtosis of the distribution.
...	arguments to be passed.

Details

This function was ported from the RMetrics package fUtilities to eliminate a dependency on fUtilities being loaded every time. This function is identical except for the addition of checkData and additional labeling.

$$Kurtosis(moment) = \frac{1}{n} * \sum_{i=1}^{n} (\frac{r_i - \overline{r}}{\sigma_P})^4$$

$$Kurtosis(excess) = \frac{1}{n} * \sum_{i=1}^{n} (\frac{r_i - \overline{r}}{\sigma_P})^4 - 3$$

$$Kurtosis(sample) = \frac{n * (n+1)}{(n-1) * (n-2) * (n-3)} * \sum_{i=1}^{n} (\frac{r_i - \overline{r}}{\sigma_{S_P}})^4$$

$$Kurtosis(fisher) = \frac{(n+1) * (n-1)}{(n-2) * (n-3)} * (\frac{\sum_{i=1}^{n} (r_i - \overline{r})^4 / n}{(\sum_{i=1}^{n} (r_i - \overline{r})^2 / n)^2} - \frac{3 * (n-1)}{n+1})$$

$$Kurtosis(sampleexcess) = \frac{n * (n+1)}{(n-1) * (n-2) * (n-3)} * \sum_{i=1}^{n} (\frac{r_i - \overline{r}}{\sigma_{S_P}})^4 - \frac{3 * (n-1)^2}{(n-2) * (n-3)}$$

where n is the number of return, $\overline{r}$ is the mean of the return distribution, σ_P is its standard deviation and σ_{S_P} is its sample standard deviation

Author(s)

Diethelm Wuertz, Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.84-85

See Also

skewness.

Examples

```
## mean -
## var -
# Mean, Variance:
r <- rnorm(100)
mean(r)
var(r)

## kurtosis -
kurtosis(r)

data(managers)
kurtosis(managers[, 1:8])
```

```
data(portfolio_bacon)
print(kurtosis(portfolio_bacon[, 1], method = "sample")) # expected 3.03
print(kurtosis(portfolio_bacon[, 1], method = "sample_excess")) # expected -0.41
print(kurtosis(managers["1996"], method = "sample"))
print(kurtosis(managers["1996", 1], method = "sample"))
```

`Level.calculate` *Calculate appropriate cumulative return series or asset level using xts attribute information*

Description

This function calculates the time varying index level over the entire period of available data. It works with arithmetic, log, and difference returns, natively reading the `coredata_content` attribute of the `xts` object to determine the correct calculation logic.

Usage

```
Level.calculate(R, seedValue = NULL, initial = TRUE)
```

Arguments

`R`	an xts object
`seedValue`	a numeric scalar indicating the (usually initial) index level or price of the series
`initial`	(default TRUE) a TRUE/FALSE flag associated with 'seedValue', indicating if this value is at the begginning of the series (TRUE) or at the end of the series (FALSE)

Details

The function relies on the `coredata_content` attribute, which is automatically set by `Return.calculate`. If this attribute is missing, it gracefully defaults to `"discreteReturn"`.

If the first value in the left-most column is `NA`, it will be populated with the `seedValue` (which defaults to `1`). However, if the first value is not `NA`, the previous date will be estimated based on the periodicity of the time series and populated with the `seedValue`.

This is designed so that information is not lost if levels are converted back to returns (where the first value results in an `NA`). Note: the estimated previous date does not consider weekdays or holidays; it simply calculates the previous calendar day. If users run `Return.calculate()` from this package, this will be a non-issue as it prepends the `NA` row automatically.

For arithmetic (`discreteReturn`) returns:

$$(1 + r_1)(1 + r_2)(1 + r_3) \dots (1 + r_n) = cumprod(1 + R)$$

For `logReturn` returns:

$$exp(r_1 + r_2 + r_3 + \dots + r_n) = exp(cumsum(R))$$

For `difference` returns:

$$r_1 + r_2 + r_3 + \ldots + r_n = cumsum(R)$$

#'

Value

An xts object containing the calculated price level or cumulative return series.

Author(s)

Erol Biceroglu

See Also

`Return.calculate`

Examples

```
# Using a price series
data(prices)

# Calculate discrete returns
ret <- Return.calculate(as.xts(prices), method = "discrete")

# Recover the price level
# The first row of returns is NA, which will be populated with seedValue (1 by default)
prices_recovered <- Level.calculate(ret, seedValue = 100)
head(prices_recovered)

# Compare to Return.cumulative
data(managers)
mgr_eq <- managers[, 1:6] # equities only
xtsAttributes(mgr_eq) <- list(coredata_content = "discreteReturn")
mgr_level <- Level.calculate(mgr_eq)

# Here they are equal
Return.cumulative(mgr_eq)
tail(mgr_level - 1, 1)
```

---

lpm *calculate a lower partial moment for a time series*

Description

Caclulate a Lower Partial Moment around the mean or a specified threshold.

Usage

```
lpm(
  R,
  n = 2,
  threshold = 0,
  about_mean = FALSE,
  SE = FALSE,
  SE.control = NULL,
  ...
)
```

Arguments

R	xts data
n	the n-th moment to return
threshold	threshold can be the mean or any point as desired
about_mean	TRUE/FALSE calculate LPM about the mean under the threshold or use the threshold to calculate the LPM around (if FALSE)
SE	TRUE/FALSE whether to ouput the standard errors of the estimates of the risk measures, default FALSE.
SE.control	Control parameters for the computation of standard errors. Should be done using the RPESE.control function.
...	Additional parameters.

Details

Lower partial moments capture negative deviation from a reference point. That reference point may be the mean, or some specified threshold that has other meaning for the investor.

Author(s)

Kyle Balkissoon <kylebalkisoon@gmail.com>

References

Huffman S.P. & Moll C.R., "The impact of Asymmetry on Expected Stock Returns: An Investigation of Alternative Risk Measures", Algorithmic Finance 1, 2011 p. 79-93

M2Sortino *M squared for Sortino of the return distribution*

Description

M squared for Sortino is a M^2 calculated for Downside risk instead of Total Risk

Usage

```
M2Sortino(Ra, Rb, MAR = 0, ...)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset return
Rb	return vector of the benchmark asset
MAR	the minimum acceptable return
...	any other passthru parameters

Details

$$M_S^2 = r_P + Sortinoratio * (\sigma_{DM} - \sigma_D)$$

where M_S^2 is MSquared for Sortino, r_P is the annualised portfolio return, σ_{DM} is the benchmark annualised downside risk and D is the portfolio annualised downside risk

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.102-103

Examples

```
data(portfolio_bacon)
MAR = 0.005
print(M2Sortino(portfolio_bacon[,1], portfolio_bacon[,2], MAR)) #expected 0.1035

data(managers)
MAR = 0
print(MSquaredExcess(managers['1996',1], managers['1996',8], MAR))
print(MSquaredExcess(managers['1996',1:5], managers['1996',8], MAR))
```

managers *Hypothetical Alternative Asset Manager and Benchmark Data*

Description

A xts object that contains columns of monthly returns for six hypothetical asset managers (HAM1 through HAM6), the EDHEC Long-Short Equity hedge fund index, the S&P 500 total returns, and total return series for the US Treasury 10-year bond and 3-month bill. Monthly returns for all series end in December 2006 and begin at different periods starting from January 1996. Note that all the EDHEC indices are available in edhec.

Usage

```
managers
```

Format

CSV conformed into an xts object with monthly observations

Details

Please note that the 'managers' data set included with PerformanceAnalytics will be periodically updated with new managers and information. If you intend to use this data set in automated tests, please be sure to subset your data like `managers[1:120,1:6]` to use the first ten years of observations on HAM1-HAM6.

Examples

```
data(managers)
#preview the data
head(managers)
#cumulative returns
tail(cumprod(1+managers),1)
```

MarketTiming *Market timing models*

Description

Allows to estimate Treynor-Mazuy or Merton-Henriksson market timing model. The Treynor-Mazuy model is essentially a quadratic extension of the basic CAPM. It is estimated using a multiple regression. The second term in the regression is the value of excess return squared. If the gamma coefficient in the regression is positive, then the estimated equation describes a convex upward-sloping regression "line". The quadratic regression is:

$$R_p - R_f = \alpha + \beta(R_b - R_f) + \gamma(R_b - R_f)^2 + \varepsilon_p$$

γ is a measure of the curvature of the regression line. If γ is positive, this would indicate that the manager's investment strategy demonstrates market timing ability.

Usage

```
MarketTiming(Ra, Rb, Rf = 0, method = c("TM", "HM"), ...)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of the asset returns
Rb	an xts, vector, matrix, data frame, timeSeries or zoo object of the benchmark asset return
Rf	risk free rate, in same period as your returns
method	used to select between Treynor-Mazuy and Henriksson-Merton models. May be any of: • TM - Treynor-Mazuy model, • HM - Henriksson-Merton model By default Treynor-Mazuy is selected
...	any other passthrough parameters

Details

The basic idea of the Merton-Henriksson test is to perform a multiple regression in which the dependent variable (portfolio excess return and a second variable that mimics the payoff to an option). This second variable is zero when the market excess return is at or below zero and is 1 when it is above zero:

$$R_p - R_f = \alpha + \beta(R_b - R_f) + \gamma D + \varepsilon_p$$

where all variables are familiar from the CAPM model, except for the up-market return $D = max(0, R_f - R_b)$ and market timing abilities γ

Author(s)

Andrii Babii, Peter Carl

References

J. Christopherson, D. Carino, W. Ferson. *Portfolio Performance Measurement and Benchmarking*. 2009. McGraw-Hill, p. 127-133.
J. L. Treynor and K. Mazuy, "Can Mutual Funds Outguess the Market?" *Harvard Business Review*, vol44, 1966, pp. 131-136
Roy D. Henriksson and Robert C. Merton, "On Market Timing and Investment Performance. II. Statistical Procedures for Evaluating Forecast Skills," *Journal of Business*, vol.54, October 1981, pp.513-533

See Also

CAPM.beta

Examples

```
data(managers)
MarketTiming(managers[,1], managers[,8], Rf=.035/12, method = "HM")
MarketTiming(managers[80:120,1:6], managers[80:120,7], managers[80:120,10])
MarketTiming(managers[80:120,1:6], managers[80:120,8:7], managers[80:120,10], method = "TM")
```

`MartinRatio` *Martin ratio of the return distribution*

Description

To calculate Martin ratio we divide the difference of the portfolio return and the risk free rate by the Ulcer index

Usage

```
MartinRatio(R, Rf = 0, ...)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rf`	risk free rate, in same period as your returns
`...`	any other passthru parameters

Details

$$Martinratio = \frac{r_P - r_F}{\sqrt{\sum_{i=1}^{n} \frac{D_i'^2}{n}}}$$

where r_P is the annualized portfolio return, r_F is the risk free rate, n is the number of observations of the entire series, D_i' is the drawdown since previous peak in period i

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.91

Examples

```
data(portfolio_bacon)
print(MartinRatio(portfolio_bacon[,1])) #expected 1.70

data(managers)
print(MartinRatio(managers['1996']))
print(MartinRatio(managers['1996',1]))
```

---

maxDrawdown *caclulate the maximum drawdown from peak equity*

Description

To find the maximum drawdown in a return series, we need to first calculate the cumulative returns and the maximum cumulative return to that point. Any time the cumulative returns dips below the maximum cumulative returns, it's a drawdown. Drawdowns are measured as a percentage of that maximum cumulative return, in effect, measured from peak equity.

Usage

```
maxDrawdown(R, weights = NULL, geometric = TRUE, invert = TRUE, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
weights	portfolio weighting vector, default NULL, see Details
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
invert	TRUE/FALSE whether to invert the drawdown measure. see Details.
...	any other passthru parameters

Details

The option to `invert` the measure should appease both academics and practitioners. The default option `invert=TRUE` will provide the drawdown as a positive number. This should be useful for optimization (which usually seeks to minimize a value), and for tables (where having negative signs in front of every number may be considered clutter). Practitioners will argue that drawdowns denote losses, and should be internally consistent with the quantile (a negative number), for which `invert=FALSE` will provide the value they expect. Individually, different preferences may apply for clarity and compactness. As such, we provide the option, but make no value judgment on which approach is preferable.

Author(s)

Peter Carl

References

Bacon, C. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 3rd Edition. 2023. p. 194-196

See Also

findDrawdowns
sortDrawdowns
table.Drawdowns
table.DownsideRisk
chart.Drawdown

Examples

```
data(edhec)
t(round(maxDrawdown(edhec[, "Funds of Funds"]), 4))
data(managers)
t(round(maxDrawdown(managers), 4))
```

MCA *Functions for doing Moment Component Analysis (MCA) of financial time series*

Description

calculates MCA coskewness and cokurtosis matrices

Usage

```
M3.MCA(R, k = 1, as.mat = TRUE, ...)

M4.MCA(R, k = 1, as.mat = TRUE, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
k	the number of components to use
as.mat	TRUE/FALSE whether to return the full moment matrix or only the vector with the unique elements (the latter is advised for speed), default TRUE
...	any other passthru parameters

Details

The coskewness and cokurtosis matrices are defined as the matrices of dimension p x p^2 and p x p^3 containing the third and fourth order central moments. They are useful for measuring nonlinear dependence between different assets of the portfolio and computing modified VaR and modified ES of a portfolio.

MCA is a generalization of PCA to higher moments. The principal components in MCA are the ones that maximize the coskewness and cokurtosis present when projecting onto these directions. It was introduced by Lim and Morton (2007) and applied to financial returns data by Jondeau and Rockinger (2017)

If a coskewness matrix (argument M3) or cokurtosis matrix (argument M4) is passed in using ..., then MCA is performed on the given comoment matrix instead of the sample coskewness or cokurtosis matrix.

Author(s)

Dries Cornilly

References

Lim, Hek-Leng and Morton, Jason. 2007. Principal Cumulant Component Analysis. working paper

Jondeau, Eric and Jurczenko, Emmanuel. 2018. Moment Component Analysis: An Illustration With International Stock Markets. Journal of Business and Economic Statistics, 36(4), 576-598.

See Also

CoMoments
ShrinkageMoments
StructuredMoments
EWMAMoments
NCE

Examples

```
data(edhec)

# coskewness matrix based on two components
M3mca <- M3.MCA(edhec, k = 2)$M3mca

# screeplot MCA
M3dist <- M4dist <- rep(NA, ncol(edhec))
M3S <- M3.MM(edhec)  # sample coskewness estimator
M4S <- M4.MM(edhec)  # sample cokurtosis estimator
for (k in 1:ncol(edhec)) {
  M3MCA_list <- M3.MCA(edhec, k)
  M4MCA_list <- M4.MCA(edhec, k)

  M3dist[k] <- sqrt(sum((M3S - M3MCA_list$M3mca)^2))
  M4dist[k] <- sqrt(sum((M4S - M4MCA_list$M4mca)^2))
}
```

```
par(mfrow = c(2, 1))
plot(1:ncol(edhec), M3dist)
plot(1:ncol(edhec), M4dist)
par(mfrow = c(1, 1))
```

`mean.geometric` *calculate attributes relative to the mean of the observation series given, including geometric, stderr, LCL and UCL*

Description

`mean.geometric`	geometric mean
`mean.stderr`	standard error of the mean (S.E. mean)
`mean.LCL`	lower confidence level (LCL) of the mean
`mean.UCL`	upper confidence level (UCL) of the mean

Usage

```
## S3 method for class 'geometric'
mean(x, ...)

## S3 method for class 'arithmetic'
mean(x, SE = FALSE, SE.control = NULL, ...)

## S3 method for class 'stderr'
mean(x, ...)

## S3 method for class 'LCL'
mean(x, ci = 0.95, ...)

## S3 method for class 'UCL'
mean(x, ci = 0.95, ...)
```

Arguments

`x`	a vector, matrix, data frame, or time series to calculate the modified mean statistic over
`...`	any other passthru parameters
`SE`	TRUE/FALSE whether to ouput the standard errors of the estimates of the risk measures, default FALSE. Only available for `mean.arithmetic`.
`SE.control`	Control parameters for the computation of standard errors. Should be done using the `RPESE.control` function. Only available for `mean.arithmetic`.
`ci`	the confidence interval to use

Author(s)

Peter Carl

See Also

sd
mean

Examples

```
data(edhec)
mean.geometric(edhec[,"Funds of Funds"])
mean.stderr(edhec[,"Funds of Funds"])
mean.UCL(edhec[,"Funds of Funds"])
mean.LCL(edhec[,"Funds of Funds"])
```

MeanAbsoluteDeviation *Mean absolute deviation of the return distribution*

Description

To calculate Mean absolute deviation we take the sum of the absolute value of the difference between the returns and the mean of the returns and we divide it by the number of returns.

Usage

```
MeanAbsoluteDeviation(R, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters

Details

$$MeanAbsoluteDeviation = \frac{\sum_{i=1}^{n} \mid r_i - \overline{r} \mid}{n}$$

where n is the number of observations of the entire series, r_i is the return in month i and $\overline{r}$ is the mean return

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.62

Examples

```
data(portfolio_bacon)
print(MeanAbsoluteDeviation(portfolio_bacon[,1])) #expected 0.0310

data(managers)
print(MeanAbsoluteDeviation(managers['1996']))
print(MeanAbsoluteDeviation(managers['1996',1]))
```

MinTrackRecord *Minimum Track Record Length*

Description

The Minimum Track Record Length responds to the following question: "How long should a track record be in order to have a p-level statistical confidence that its Sharpe ratio is above a given threshold?". Obviously, the main assumption is the returns will continue displaying the same statistical properties out-of-sample. For example, if the input contains fifty observations and the Minimum Track Record is forty, then for the next ten observations the relevant measures (sharpe ratio, skewness and kyrtosis) need to remain the same as the input so to achieve statistical significance after exactly ten time points.

Usage

```
MinTrackRecord(
  R = NULL,
  Rf = 0,
  refSR,
  p = 0.95,
  weights = NULL,
  n = NULL,
  sr = NULL,
  sk = NULL,
  kr = NULL,
  ignore_skewness = FALSE,
  ignore_kurtosis = TRUE
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of the returns input
Rf	the risk free rate

refSR	a single value or a vector when R is multicolumn. It defines the reference Sharpe Ratio and should be in the same periodicity as the returns (non-annualized).
p	the confidence level
weights	(if R is multicolumn and the underlying assets form a portfolio) the portfolio weights
n	(if R is NULL) the track record length of the returns
sr	(if R is NULL) the sharpe ratio of the returns
sk	(if R is NULL) the skewness of the returns
kr	(if R is NULL) the kurtosis of the returns
ignore_skewness	If TRUE, it ignores the effects of skewness in the calculations
ignore_kurtosis	If TRUE, it ignores the effects of kurtosis in the calculations

Value

A list containing the below

- min_TRL: The minimum track record length value (periodicity follows R)
- IS_SR_SIGNIFICANT: TRUE if the sharpe ratio is statistically significant, FALSE otherwise
- num_of_extra_obs_needed: If the sharpe ratio is not statistically significant, how many more observations are needed so as to achieve this

Author(s)

Tasos Grivas <tasos@openriskcalculator.com>, Pulkit Mehrotra

References

Bailey, David H. and Lopez de Prado, Marcos, The Sharpe Ratio Efficient Frontier (July 1, 2012). Journal of Risk, Vol. 15, No. 2, Winter 2012/13

See Also

ProbSharpeRatio

Examples

```
data(edhec)
MinTrackRecord(edhec[,1],refSR = 0.23)
MinTrackRecord(refSR = 1/12^0.5,Rf = 0,p=0.95,sr = 2/12^0.5,sk=-0.72,kr=5.78,n=59)

### Higher moments are data intensive, kurtosis shouldn't be used for short timeseries
MinTrackRecord(edhec[,1:2],refSR = c(0.28,0.24), ignore_skewness = FALSE, ignore_kurtosis = FALSE)
MinTrackRecord(edhec[,1:2],refSR = c(0.28,0.24), ignore_skewness = FALSE, ignore_kurtosis = TRUE)
MinTrackRecord(edhec[,1:2],refSR = c(0.28,0.24), ignore_skewness = TRUE, ignore_kurtosis = TRUE)

MinTrackRecord(edhec[,1:2],refSR = 0.26,weights = c(0.5,0.5),
```

```
ignore_skewness = FALSE, ignore_kurtosis = FALSE)
```

`Modigliani` *Modigliani-Modigliani measure*

Description

The Modigliani-Modigliani measure is the portfolio return adjusted upward or downward to match the benchmark's standard deviation. This puts the portfolio return and the benchmark return on 'equal footing' from a standard deviation perspective.

$$MM_p = \frac{E[R_p - R_f]}{\sigma_p} = SR_p * \sigma_b + E[R_f]$$

where SR_p - Sharpe ratio, σ_b - benchmark standard deviation

Usage

```
Modigliani(Ra, Rb, Rf = 0, ...)
```

Arguments

`Ra`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rb`	return vector of the benchmark asset
`Rf`	risk free rate, in same period as your returns
`...`	any other passthrough parameters

Details

This is also analogous to some approaches to 'risk parity' portfolios, which use (presumably costless) leverage to increase the portfolio standard deviation to some target.

Author(s)

Andrii Babii, Brian G. Peterson

References

J. Christopherson, D. Carino, W. Ferson. *Portfolio Performance Measurement and Benchmarking*. 2009. McGraw-Hill, p. 97-99.
Franco Modigliani and Leah Modigliani, "Risk-Adjusted Performance: How to Measure It and Why," *Journal of Portfolio Management*, vol.23, no., Winter 1997, pp.45-54

See Also

`SharpeRatio`, `TreynorRatio`

Examples

```
data(managers)
Modigliani(managers[,1,drop=FALSE], managers[,8,drop=FALSE], Rf=.035/12)
Modigliani(managers[,1:6], managers[,8,drop=FALSE], managers[,8,drop=FALSE])
Modigliani(managers[,1:6], managers[,8:7], managers[,8,drop=FALSE])
```

MSquared *M squared of the return distribution*

Description

M squared is a risk adjusted return useful to judge the size of relative performance between differents portfolios. With it you can compare portfolios with different levels of risk.

Usage

```
MSquared(Ra, Rb, Rf = 0, ...)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset return
Rb	return vector of the benchmark asset
Rf	risk free rate, in same period as your returns
...	any other passthru parameters

Details

$$M^2 = r_P + SR * (\sigma_M - \sigma_P) = (r_P - r_F) * \frac{\sigma_M}{\sigma_P} + r_F$$

where r_P is the portfolio return annualized, σ_M is the market risk and σ_P is the portfolio risk

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.67-68

Examples

```
data(portfolio_bacon)
print(MSquared(portfolio_bacon[,1], portfolio_bacon[,2])) #expected 0.10062

data(managers)
print(MSquared(managers['1996',1], managers['1996',8]))
print(MSquared(managers['1996',1:5], managers['1996',8]))
```

`MSquaredExcess` *M squared excess of the return distribution*

Description

M squared excess is the quantity above the standard M. There is a geometric excess return which is better for Bacon and an arithmetic excess return

Usage

```
MSquaredExcess(Ra, Rb, Rf = 0, Method = c("geometric", "arithmetic"), ...)
```

Arguments

`Ra`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset return
`Rb`	return vector of the benchmark asset
`Rf`	risk free rate, in same period as your returns
`Method`	one of "geometric" or "arithmetic" indicating the method to use to calculate MSquareExcess
`...`	any other passthru parameters

Details

$$M^2 excess(geometric) = \frac{1 + M^2}{1 + b} - 1$$

$$M^2 excess(arithmetic) = M^2 - b$$

where M^2 is MSquared and b is the benchmark annualised return.

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.68

Examples

```
data(portfolio_bacon)
MSquaredExcess(portfolio_bacon[,1], portfolio_bacon[,2]) #expected -0.00998

MSquaredExcess(portfolio_bacon[,1], portfolio_bacon[,2], Method="arithmetic") #expected -0.011

data(managers)
MSquaredExcess(managers['1996',1], managers['1996',8])
MSquaredExcess(managers['1996',1:5], managers['1996',8])
```

NCE *Functions for calculating the nearest comoment estimator for financial time series*

Description

calculates NCE covariance, coskewness and cokurtosis matrices

Usage

```
MM.NCE(R, as.mat = TRUE, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns (with mean zero)
as.mat	TRUE/FALSE whether to return the full moment matrix or only the vector with the unique elements (the latter is advised for speed), default TRUE
...	any other passthru parameters, see details.

Details

The coskewness and cokurtosis matrices are defined as the matrices of dimension p x p^2 and p x p^3 containing the third and fourth order central moments. They are useful for measuring nonlinear dependence between different assets of the portfolio and computing modified VaR and modified ES of a portfolio.

The nearest comoment estimator is a way to estimate the covariance, coskewness and cokurtosis matrix by means of a latent multi-factor model. The method is proposed in Boudt, Cornilly and Verdonck (2018).

The optional arguments include the number of factors, given by ‘k‘ and the weight matrix ‘W‘, see the examples.

Author(s)

Dries Cornilly

References

Boudt, Kris, Dries Cornilly and Tim Verdonck. 2020. Nearest comoment estimation with unobserved factors. Journal of Econometrics, 217(2), 381-397.

See Also

CoMoments
ShrinkageMoments
StructuredMoments
EWMAMoments
MCA

Examples

```
## Not run:
# CRAN doesn't like how long this takes (>5 secs)
data(edhec)

# default estimator
est_nc <- MM.NCE(edhec[, 1:3] * 100)

# scree plot to determine number of factors
obj <- rep(NA, 5)
for (ii in 1:5) {
  est_nc <- MM.NCE(edhec[, 1:5] * 100, k = ii - 1)
  obj[ii] <- est_nc$optim.sol$objective * nrow(edhec[, 1:5])
}
plot(0:4, obj,
  type = "b", xlab = "number of factors",
  ylab = "objective value", las = 1
)

# bootstrapped estimator
est_nc <- MM.NCE(edhec[, 1:2] * 100, W = list(
  "Wid" = "RidgeD",
  "alpha" = NULL, "nb" = 250, "alphavec" = seq(0.2, 1, by = 0.2)
))

# ridge weight matrix with alpha = 0.5
est_nc <- MM.NCE(edhec[, 1:2] * 100, W = list("Wid" = "RidgeD", "alpha" = 0.5))
# end

## End(Not run)
```

`NetSelectivity` *Net selectivity of the return distribution*

Description

Net selectivity is the remaining selectivity after deducting the amount of return require to justify not being fully diversified

Usage

```
NetSelectivity(Ra, Rb, Rf = 0, ...)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
Rf	risk free rate, in same period as your returns
...	any other passthru parameters

Details

If net selectivity is negative the portfolio manager has not justified the loss of diversification

$$Netselectivity = \alpha - d$$

where α is the selectivity and d is the diversification

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.78

Examples

```
data(portfolio_bacon)
print(NetSelectivity(portfolio_bacon[,1], portfolio_bacon[,2])) #expected -0.017

data(managers)
print(NetSelectivity(managers['1996',1], managers['1996',8]))
print(NetSelectivity(managers['1996',1:5], managers['1996',8]))
```

Omega *calculate Omega for a return series*

Description

Keating and Shadwick (2002) proposed Omega (referred to as Gamma in their original paper) as a way to capture all of the higher moments of the returns distribution.

Usage

```
Omega(
  R,
  L = 0,
  method = c("simple", "interp", "binomial", "blackscholes"),
  output = c("point", "full"),
  Rf = 0,
  SE = FALSE,
  SE.control = NULL,
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
L	L is the loss threshold that can be specified as zero, return from a benchmark index, or an absolute rate of return - any specified level
method	one of: simple, interp, binomial, blackscholes
output	one of: point (in time), or full (distribution of Omega)
Rf	risk free rate, as a single number
SE	TRUE/FALSE whether to ouput the standard errors of the estimates of the risk measures, default FALSE.
SE.control	Control parameters for the computation of standard errors. Should be done using the RPESE.control function.
...	any other passthru parameters

Details

Mathematically, Omega is: integral[L to b](1 - F(r))dr / integral[a to L](F(r))dr

where the cumulative distribution F is defined on the interval (a,b). L is the loss threshold that can be specified as zero, return from a benchmark index, or an absolute rate of return - any specified level. When comparing alternatives using Omega, L should be common.

Input data can be transformed prior to calculation, which may be useful for introducing risk aversion.

This function returns a vector of Omega, useful for plotting. The steeper, the less risky. Above it's mean, a steeply sloped Omega also implies a very limited potential for further gain.

Omega has a value of 1 at the mean of the distribution.

Omega is sub-additive. The ratio is dimensionless.

Kazemi, Schneeweis, and Gupta (2003), in "Omega as a Performance Measure" show that Omega can be written as: Omega(L) = C(L)/P(L) where C(L) is essentially the price of a European call option written on the investment and P(L) is essentially the price of a European put option written on the investment. The maturity for both options is one period (e.g., one month) and L is the strike price of both options.

The numerator and the denominator can be expressed as: exp(-Rf=0) * E[max(x - L, 0)] exp(-Rf=0) * E[max(L - x, 0)] with exp(-Rf=0) calculating the present values of the two, where rf is the per-period riskless rate.

The first three methods implemented here focus on that observation. The first method takes the simplification described above. The second uses the Black-Scholes option pricing as implemented in fOptions. The third uses the binomial pricing model from fOptions. The second and third methods are not implemented here.

The fourth method, "interp", creates a linear interpolation of the cdf of returns using `Ecdf`, and evaluates Omega via mathematically rigorous trapezoidal integration over the area of that empirical CDF. The "interp" method effectively assumes the continuous CDF is linearly interpolated between empirical data points, returning a highly accurate continuous curve, whereas the "simple" method evaluates the exact empirical discrete Expected Values (which acts as a rectangle integration over the step-function CDF).

For both "simple" and "interp" methods, if the threshold L is strictly outside the observed domain of returns (e.g. all returns are strictly above or below L), the ratio will mathematically evaluate to `Inf` or `0` respectively.

Author(s)

Peter Carl

References

Keating, J. and Shadwick, W.F. The Omega Function. working paper. Finance Development Center, London. 2002. Kazemi, Schneeweis, and Gupta. Omega as a Performance Measure. 2003.

See Also

`Ecdf`

Examples

```
data(edhec)
Omega(edhec)

# CRAN (questionably(ahem) requires these methods to not run if you don't have Suggests loaded)
if (requireNamespace("Hmisc", quietly = TRUE)) {
  Omega(edhec[, 13], method = "interp", output = "point")
  Omega(edhec[, 13], method = "interp", output = "full")
```

```
} # end  spurious CRAN check
```

`OmegaExcessReturn` *Omega excess return of the return distribution*

Description

Omega excess return is another form of downside risk-adjusted return. It is calculated by multiplying the downside variance of the style benchmark by 3 times the style beta.

Usage

```
OmegaExcessReturn(Ra, Rb, MAR = 0, ...)
```

Arguments

`Ra`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rb`	return vector of the benchmark asset
`MAR`	the minimum acceptable return
`...`	any other passthru parameters

Details

$$\omega = r_P - 3 * \beta_S * \sigma^2_{MD}$$

where ω is omega excess return, β_S is style beta, σ_D is the portfolio annualised downside risk and σ_{MD} is the benchmark annualised downside risk.

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.103

Examples

```
data(portfolio_bacon)
MAR = 0.005
print(OmegaExcessReturn(portfolio_bacon[,1], portfolio_bacon[,2], MAR)) #expected 0.0805

data(managers)
MAR = 0
print(OmegaExcessReturn(managers['1996',1], managers['1996',8], MAR))
print(OmegaExcessReturn(managers['1996',1:5], managers['1996',8], MAR))
```

`OmegaSharpeRatio` *Omega-Sharpe ratio of the return distribution*

Description

The Omega-Sharpe ratio is a conversion of the omega ratio to a ranking statistic in familiar form to the Sharpe ratio.

Usage

```
OmegaSharpeRatio(R, MAR = 0, ...)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`MAR`	Minimum Acceptable Return, in the same periodicity as your returns
`...`	any other passthru parameters

Details

To calculate the Omega-Sharpe ration we subtract the target (or Minimum Acceptable Returns (MAR)) return from the portfolio return and we divide it by the opposite of the Downside Deviation.

$$OmegaSharpeRatio(R, MAR) = \frac{r_p - r_t}{\sum_{t=1}^{n} \frac{max(r_t - r_i, 0)}{n}}$$

where n is the number of observations of the entire series

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008, p.95

Examples

```
data(portfolio_bacon)
MAR = 0.005
print(OmegaSharpeRatio(portfolio_bacon[,1], MAR)) #expected 0.29

MAR = 0
data(managers)
print(OmegaSharpeRatio(managers['1996'], MAR))
print(OmegaSharpeRatio(managers['1996',1], MAR)) #expected 3.60
```

PainIndex *Pain index of the return distribution*

Description

The pain index is the mean value of the drawdowns over the entire analysis period. The measure is similar to the Ulcer index except that the drawdowns are not squared. Also, it's different than the average drawdown, in that the numerator is the total number of observations rather than the number of drawdowns.

Usage

```
PainIndex(R, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns (Note: expects decimal returns, e.g. 0.05 for 5%, not percentage units).
...	any other passthru parameters

Details

Visually, the pain index is the area of the region that is enclosed by the horizontal line at zero percent and the drawdown line in the Drawdown chart.

$$Painindex = \sum_{i=1}^{n} \frac{\mid D_i' \mid}{n}$$

where n is the number of observations of the entire series, D_i' is the drawdown since previous peak in period i

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.89, Becker, Thomas (2006) Zephyr Associates

Examples

```
data(portfolio_bacon)
print(PainIndex(portfolio_bacon[, 1])) # expected 0.04

data(managers)
print(PainIndex(100 * managers["1996"]))
print(PainIndex(100 * managers["1996", 1]))
```

PainRatio *Pain ratio of the return distribution*

Description

To calculate Pain ratio we divide the difference of the portfolio return and the risk free rate by the Pain index

Usage

```
PainRatio(R, Rf = 0, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rf	risk free rate, in same period as your returns
...	any other passthru parameters

Details

$$Painratio = \frac{r_P - r_F}{\sum_{i=1}^{n} \frac{|D_i'|}{n}}$$

where r_P is the annualized portfolio return, r_F is the risk free rate, n is the number of observations of the entire series, D_i' is the drawdown since previous peak in period i

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.91

Examples

```
data(portfolio_bacon)
print(PainRatio(portfolio_bacon[,1])) #expected 2.66

data(managers)
print(PainRatio(managers['1996']))
print(PainRatio(managers['1996',1]))
```

`portfolio-moments` *Portfolio moments*

Description

computes the portfolio second, third and fourth central moments given the multivariate comoments and the portfolio weights. The gradient functions compute the gradient of the portfolio central moment with respect to the portfolio weights, evaluated in the portfolio weights.

Usage

```
portm2(w, sigma)

derportm2(w, sigma)

portm3(w, M3)

derportm3(w, M3)

portm4(w, M4)

derportm4(w, M4)
```

Arguments

`w`	vector of length p with portfolio weights
`sigma`	portfolio covariance matrix of dimension p x p
`M3`	matrix of dimension p x p^2, or a vector with (p * (p + 1) * (p + 2) / 6) unique coskewness elements
`M4`	matrix of dimension p x p^3, or a vector with (p * (p + 1) * (p + 2) * (p + 3) / 12) unique coskewness elements

Details

For documentation on the coskewness and cokurtosis matrices, we refer to ?CoMoments. Both the full matrices and reduced form can be the used as input for the function related to the portfolio third and fourth central moments.

Author(s)

Kris Boudt, Peter Carl, Dries Cornilly, Brian Peterson

See Also

CoMoments
ShrinkageMoments
EWMAMoments
StructuredMoments
MCA

Examples

```
data(managers)

# equal weighted portfolio of managers
p <- ncol(edhec)
w <- rep(1 / p, p)

# portfolio variance and its gradient with respect to the portfolio weights
sigma <- cov(edhec)
pm2 <- portm2(w, sigma)
dpm2 <- derportm2(w, sigma)

# portfolio third central moment and its gradient with respect to the portfolio weights
m3 <- M3.MM(edhec)
pm3 <- portm3(w, m3)
dpm3 <- derportm3(w, m3)

# portfolio fourth central moment and its gradient with respect to the portfolio weights
m4 <- M4.MM(edhec)
pm4 <- portm4(w, m4)
dpm4 <- derportm4(w, m4)
```

---

portfolio_bacon *Bacon(2008) Data*

Description

A xts object that contains columns of monthly returns for an example of portfolio and its benchmark

Usage

```
portfolio_bacon
```

Format

CSV conformed into an xts object with monthly observations

Examples

```
data(portfolio_bacon)

#preview the data
head(portfolio_bacon)

#summary period statistics
summary(portfolio_bacon)

#cumulative returns
tail(cumprod(1+portfolio_bacon),1)
```

---

prices *Selected Price Series Example Data*

Description

A object returned by get.hist.quote of price data for use in the example for `Return.calculate`

Usage

```
prices
```

Format

R variable 'prices'

Examples

```
data(prices)

#preview the data
head(prices)
```

ProbSharpeRatio *Probabilistic Sharpe Ratio*

Description

Given a predefined benchmark Sharpe ratio,the observed Sharpe Ratio can be expressed in probabilistic terms known as the Probabilistic Sharpe Ratio. PSR provides an adjusted estimate of SR, by removing the inflationary effect caused by short series with skewed and/or fat-tailed returns and is defined as the probability of the observed sharpe ratio being higher than the reference sharpe ratio.

Usage

```
ProbSharpeRatio(
  R = NULL,
  Rf = 0,
  refSR,
  p = 0.95,
  weights = NULL,
  n = NULL,
  sr = NULL,
  sk = NULL,
  kr = NULL,
  ignore_skewness = FALSE,
  ignore_kurtosis = TRUE
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of the returns input
Rf	the risk free rate
refSR	a single value or a vector when R is multicolumn. It defines the reference Sharpe Ratio and should be in the same periodicity as the returns (non-annualized).
p	the confidence level
weights	(if R is multicolumn and the underlying assets form a portfolio) the portfolio weights
n	(if R is NULL) the track record length of the returns
sr	(if R is NULL) the sharpe ratio of the returns
sk	(if R is NULL) the skewness of the returns
kr	(if R is NULL) the kurtosis of the returns
ignore_skewness	If TRUE, it ignores the effects of skewness in the calculations
ignore_kurtosis	If TRUE, it ignores the effects of kurtosis in the calculations

Value

A list containing the below

- The probability that the observed Sharpe Ratio is higher than the reference one
- The p-level confidence interval of the Sharpe Ratio

Author(s)

Tasos Grivas <tasos@openriskcalculator.com>, Pulkit Mehrotra

References

Marcos Lopez de Prado. 2018. Advances in Financial Machine Learning (1st ed.). Wiley Publishing.

Examples

```
data(edhec)
ProbSharpeRatio(edhec[, 1], refSR = 0.23)
ProbSharpeRatio(
  refSR = 1 / 12^0.5, Rf = 0, p = 0.95, sr = 2 / 12^0.5,
  sk = -0.72, kr = 5.78, n = 59
)

### Higher moments are data intensive, kurtosis shouldn't be used for short timeseries
ProbSharpeRatio(edhec[, 1:2],
  refSR = c(0.28, 0.24),
  ignore_skewness = FALSE, ignore_kurtosis = FALSE
)
ProbSharpeRatio(edhec[, 1:2],
  refSR = c(0.28, 0.24),
  ignore_skewness = FALSE, ignore_kurtosis = TRUE
)
ProbSharpeRatio(edhec[, 1:2],
  refSR = c(0.28, 0.24),
  ignore_skewness = TRUE, ignore_kurtosis = TRUE
)

ProbSharpeRatio(edhec[, 1:2],
  refSR = 0.26, weights = c(0.5, 0.5),
  ignore_skewness = FALSE, ignore_kurtosis = FALSE
)
```

ProspectRatio *Prospect ratio of the return distribution*

Description

Prospect ratio is a ratio used to penalise loss since most people feel loss greater than gain

Usage

```
ProspectRatio(R, MAR, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
MAR	the minimum acceptable return
...	any other passthru parameters

Details

$$ProspectRatio(R) = \frac{\frac{1}{n} * \sum_{i=1}^{n}(Max(r_i, 0) + 2.25 * Min(r_i, 0) - MAR)}{\sigma_D}$$

where n is the number of observations of the entire series, MAR is the minimum acceptable return and σ_D is the downside risk

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.100

Examples

```
data(portfolio_bacon)
MAR = 0.05
print(ProspectRatio(portfolio_bacon[,1], MAR)) #expected -0.134

data(managers)
MAR = 0
print(ProspectRatio(managers['1996'], MAR))
print(ProspectRatio(managers['1996',1], MAR))
```

RachevRatio *Rachev Ratio*

Description

RachevRatio computation with standard errors.

Usage

```
RachevRatio(
  R,
  alpha = 0.1,
  beta = 0.1,
  rf = 0,
  SE = FALSE,
  SE.control = NULL,
  ...
)
```

Arguments

R	Data of returns for one or multiple assets or portfolios.
alpha	Lower tail probability.
beta	Upper tail probability.
rf	Risk-free interest rate.
SE	TRUE/FALSE whether to ouput the standard errors of the estimates of the risk measures, default FALSE.
SE.control	Control parameters for the computation of standard errors. Should be done using the RPESE.control function.
...	Additional parameters.

Details

The Rachev ratio, introducted in Rachev et al. (2008), is a non-parametric estimator of the upper tail reward potential relative to the lower tail risk in a non-Gaussian setting, and as such, it is particularly useful when returns have a fat-tailed and possibly skewed distribution. For small α and β, it is a measure of the potential of extreme positive returns to risk of extremel negative returns.

For lower tail parameter α and lower tail parameter β, the Rachev ratio is given by

$$\frac{ETL_{\alpha}(R_f - R_a)}{ETL_{\beta}(R_a - R_f)}$$

.

Value

A vector or a list depending on se.method.

Author(s)

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

References

Rachev, Svetlozar T. et al. (2008). Advanced Stochastic Models, Risk Assessment, and Portfolio Optimization (1st ed.)

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")
# Compute Rachev ratio for managers data
RachevRatio(edhec)
```

`replaceTabs.inner` *Display text information in a graphics plot.*

Description

This function displays text output in a graphics window. It is the equivalent of 'print' except that the output is displayed as a plot.

Usage

```
replaceTabs.inner(text, width = 8)

replaceTabs(text, width = 8)

textplot(
  object,
  halign = "center",
  valign = "center",
  cex,
  max.cex = 1,
  cmar = 2,
  rmar = 0.5,
  show.rownames = TRUE,
  show.colnames = TRUE,
  hadj = 1,
  vadj = NULL,
  row.valign = "center",
  heading.valign = "bottom",
  mar = c(0, 0, 0, 0) + 0.1,
  col.data = par("col"),
  col.rownames = par("col"),
  col.colnames = par("col"),
  wrap = TRUE,
  wrap.colnames = 10,
  wrap.rownames = 10,
  ...
)

## Default S3 method:
textplot(
  object,
  halign = c("center", "left", "right"),
  valign = c("center", "top", "bottom"),
  cex,
  max.cex,
  cmar,
  rmar,
```

```
  show.rownames,
  show.colnames,
  hadj,
  vadj,
  row.valign,
  heading.valign,
  mar,
  col.data,
  col.rownames,
  col.colnames,
  wrap,
  wrap.colnames,
  wrap.rownames,
  ...
)

## S3 method for class 'data.frame'
textplot(
  object,
  halign = c("center", "left", "right"),
  valign = c("center", "top", "bottom"),
  cex,
  max.cex = 1,
  cmar = 2,
  rmar = 0.5,
  show.rownames = TRUE,
  show.colnames = TRUE,
  hadj = 1,
  vadj = NULL,
  row.valign = "center",
  heading.valign = "bottom",
  mar = c(0, 0, 0, 0) + 0.1,
  col.data = par("col"),
  col.rownames = par("col"),
  col.colnames = par("col"),
  wrap = TRUE,
  wrap.colnames = 10,
  wrap.rownames = 10,
  ...
)

## S3 method for class 'matrix'
textplot(
  object,
  halign = c("center", "left", "right"),
  valign = c("center", "top", "bottom"),
  cex,
  max.cex = 1,
```

```
  cmar = 2,
  rmar = 0.5,
  show.rownames = TRUE,
  show.colnames = TRUE,
  hadj = 1,
  vadj = NULL,
  row.valign = "center",
  heading.valign = "bottom",
  mar = c(0, 0, 0, 0) + 0.1,
  col.data = par("col"),
  col.rownames = par("col"),
  col.colnames = par("col"),
  wrap = TRUE,
  wrap.colnames = 10,
  wrap.rownames = 10,
  ...
)

## S3 method for class 'character'
textplot(
  object,
  halign = c("center", "left", "right"),
  valign = c("center", "top", "bottom"),
  cex,
  max.cex = 1,
  cmar = 2,
  rmar = 0.5,
  show.rownames = TRUE,
  show.colnames = TRUE,
  hadj = 1,
  vadj = NULL,
  row.valign = "center",
  heading.valign = "bottom",
  mar = c(0, 0, 3, 0) + 0.1,
  col.data = par("col"),
  col.rownames = par("col"),
  col.colnames = par("col"),
  wrap = TRUE,
  wrap.colnames = 10,
  wrap.rownames = 10,
  fixed.width = TRUE,
  cspace = 1,
  lspace = 1,
  tab.width = 8,
  ...
)
```

Arguments

text	in the function 'replaceTabs', the text string to be processed
width	in the function 'replaceTabs', the number of spaces to replace tabs with
object	Object to be displayed.
halign	Alignment in the x direction, one of "center", "left", or "right".
valign	Alignment in the y direction, one of "center", "top" , or "bottom"
cex	Character size, see par for details. If unset, the code will attempt to use the largest value which allows the entire object to be displayed.
max.cex	Sets the largest text size as a ceiling
rmar, cmar	Space between rows or columns, in fractions of the size of the letter 'M'.
show.rownames, show.colnames	Logical value indicating whether row or column names will be displayed.
hadj, vadj	Vertical and horizontal location of elements within matrix cells. These have the same meaning as the adj graphics paramter (see par).
row.valign	Sets the vertical alignment of the row as "top", "bottom", or (default) "center".
heading.valign	Sets the vertical alignment of the heading as "top", (default) "bottom", or "center".
mar	Figure margins, see the documentation for par.
col.data	Colors for data elements. If a single value is provided, all data elements will be the same color. If a matrix matching the dimensions of the data is provided, each data element will receive the specified color.
col.rownames, col.colnames	Colors for row names and column names, respectively. Either may be specified as a scalar or a vector of appropriate length.
wrap	If TRUE (default), will wrap column names and rownames
wrap.colnames	The number of characters after which column labels will be wrapped. Default is 10.
wrap.rownames	The number of characters after which row headings will be wrapped. Default is 10.
...	Optional arguments passed to the text plotting command or specialized object methods
fixed.width	default is TRUE
cspace	default is 1
lspace	default is 1
tab.width	default is 8

Details

A new plot is created and the object is displayed using the largest font that will fit on in the plotting region. The halign and valign parameters can be used to control the location of the string within the plotting region.

For matrixes and vectors a specialized textplot function is available, which plots each of the cells individually, with column widths set according to the sizes of the column elements. If present, row and column labels will be displayed in a bold font.

textplot also uses replaceTabs, a function to replace all tabs in a string with an appropriate number of spaces. That function was also written by Gregory R. Warnes and included in the 'gplots' package.

Author(s)

Originally written by Gregory R. Warnes <warnes@bst.rochester.edu> for the package 'gplots', modified by Peter Carl

See Also

plot,
text,
capture.output,
textplot

Examples

```
 # don't test on CRAN, since it requires Suggested packages
# Also see the examples in the original gplots textplot function
data(managers)
textplot(table.AnnualizedReturns(managers[,1:6]))

# This was really nice before Hmisc messed up 'format' from R-base
# prettify with format.df in hmisc package
# require("Hmisc")
# result = t(table.CalendarReturns(managers[,1:8]))[-1:-12,]

#  textplot(Hmisc::format.df(result, na.blank=TRUE, numeric.dollar=FALSE,
#           cdec=rep(1,dim(result)[2])), rmar = 0.8, cmar = 1,  max.cex=.9,
#           halign = "center", valign = "top", row.valign="center", wrap.rownames=20,
#           wrap.colnames=10, col.rownames=c("red", rep("darkgray",5),
#           rep("orange",2)), mar = c(0,0,4,0)+0.1)
#
# title(main="Calendar Returns")
```

Return.annualized *calculate an annualized return for comparing instruments with different length history*

Description

An average annualized return is convenient for comparing returns.

Usage

```
Return.annualized(R, scale = NA, geometric = TRUE, na.rm = TRUE)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
scale	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
na.rm	TRUE/FALSE whether to remove NA values before calculation, default TRUE

Details

Annualized returns are useful for comparing two assets. To do so, you must scale your observations to an annual scale by raising the compound return to the number of periods in a year, and taking the root to the number of total observations:

$$prod(1 + R_a)^{\frac{scale}{n}} - 1 = \sqrt[n]{prod(1 + R_a)^{scale}} - 1$$

where scale is the number of periods in a year, and n is the total number of periods for which you have observations.

For simple returns (geometric=FALSE), the formula is:

$$\overline{R_a} \cdot scale$$

Author(s)

Peter Carl

References

Bacon, Carl. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004. p. 6

See Also

Return.cumulative,

Examples

```
data(managers)
round(Return.annualized(managers[, 1, drop = FALSE]), 4)
round(Return.annualized(managers[, 1:8]), 4)
round(Return.annualized(managers[, 1:8], geometric = FALSE), 4)
```

`Return.annualized.excess` *calculates an annualized excess return for comparing instruments with different length history*

Description

An average annualized excess return is convenient for comparing excess returns.

Usage

```
Return.annualized.excess(Rp, Rb, scale = NA, geometric = TRUE)
```

Arguments

`Rp`	an xts, vector, matrix, data frame, timeSeries or zoo object of portfolio returns
`Rb`	an xts, vector, matrix, data frame, timeSeries or zoo object of benchmark returns
`scale`	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
`geometric`	generate geometric (TRUE) or simple (FALSE) excess returns, default TRUE

Details

Annualized returns are useful for comparing two assets. To do so, you must scale your observations to an annual scale by raising the compound return to the number of periods in a year, and taking the root to the number of total observations:

$$prod(1+R_a)^{\frac{scale}{n}} - 1 = \sqrt[n]{prod(1+R_a)^{scale}} - 1$$

where scale is the number of periods in a year, and n is the total number of periods for which you have observations.

Finally having annualized returns for portfolio and benchmark we can compute annualized excess return as difference in the annualized portfolio and benchmark returns in the arithmetic case:

$$er = R_{pa} - R_{ba}$$

and as a geometric difference in the geometric case:

$$er = \frac{(1+R_{pa})}{(1+R_{ba})} - 1$$

Author(s)

Andrii Babii

References

Bacon, Carl. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004. p. 206-207

See Also

Return.annualized,

Examples

```
data(managers)
Return.annualized.excess(Rp = managers[,1], Rb = managers[,8])
```

Return.calculate *calculate simple or compound returns from prices*

Description

calculate simple or compound returns from prices

Usage

```
Return.calculate(prices, method = c("discrete", "log", "difference"))

CalculateReturns(prices, method = c("discrete", "log"))
```

Arguments

prices	data object containing ordered price observations
method	calculate "discrete" or "log" returns, default discrete(simple)

Details

Two requirements should be made clear. First, the function Return.calculate assumes regular price data. In the example, we downloaded monthly close prices. Prices can be for any time scale, such as daily, weekly, monthly or annual, as long as the data consists of regular observations. Irregular observations require time period scaling to be comparable. Fortunately, to.period in the xts package, or aggregate.zoo in the zoo package support management and conversion of irregular time series to regular time series.

Second, if corporate actions, dividends, or other adjustments such as time- or money-weighting are to be taken into account, those calculations must be made separately. This is a simple function that assumes fully adjusted close prices as input. For the IBM timeseries in the example below, dividends and corporate actions are not contained in the "close" price series, so we end up with "price returns" instead of "total returns". This can lead to significant underestimation of the return series over longer time periods. To use adjusted returns, specify quote="AdjClose" in get.hist.quote, which is found in package tseries, or use the Ad function.

We have changed the default arguments and settings for `method` from `compound` and `simple` to `discrete` and `log` and `difference` to avoid confusion between the return type (discrete, log, difference) and the chaining method (compound or arithmetic/simple). In most of the rest of `PerformanceAnalytics`, compound and simple are used to refer to the *return chaining* method which is used with the returns. The default for this function is to use discrete returns, because most other package functions use compound chaining by default.

Other `method` argument formulations also work for clarity and backwards compatibility:

`method='discrete'` has equivalent arguments `simple` and `arithmetic`

`method='log'` has equivalent argument `compound` and `continuous`

`method='difference'` has equivalent argument `diff`

Author(s)

Peter Carl

References

Bacon, C. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004. Chapter 2

See Also

`Return.cumulative`

Examples

```
  ## Not run:  # no internet access on CRAN tests
    require(quantmod)
    prices = Cl(getSymbols("IBM", from = "1999-01-01", to = "2007-01-01"))

## End(Not run)

R.IBM = Return.calculate(xts(prices), method="discrete")
colnames(R.IBM)="IBM"
chart.CumReturns(R.IBM,legend.loc="topleft", main="Cumulative Daily Returns for IBM")
round(R.IBM,2)
```

Return.centered *calculate centered moment/co-moment return matrices*

Description

the n-th centered moment is calculated as

Usage

```
Return.centered(R, ...)

centeredmoment(R, power)

centeredcomoment(Ra, Rb, p1, p2, normalize = FALSE)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters
power	power or moment to calculate
Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	an xts, vector, matrix, data frame, timeSeries or zoo object of index, benchmark, portfolio, or secondary asset returns to compare against
p1	first power of the comoment
p2	second power of the comoment
normalize	whether to standardize the calculation to agree with common usage, or leave the default mathematical meaning

Details

.

$$\mu^{(n)}(R) = E[(R - E(R))^n]$$

These functions are used internally by PerformanceAnalytics to calculate centered moments for a multivariate distribution as well as the standardized moments of a portfolio distribution. They are exposed here for users who wish to use them directly, and we'll get more documentation written when we can.

These functions were first utilized in Boudt, Peterson, and Croux (2008), and have been subsequently used in our other research.

Author(s)

Kris Boudt and Brian Peterson

References

Boudt, Kris, Brian G. Peterson, and Christophe Croux. 2008. Estimation and Decomposition of Downside Risk for Portfolios with Non-Normal Returns. Journal of Risk. Winter.

Martellini, L. and Ziemann, V., 2010. Improved estimates of higher-order comoments and implications for portfolio selection. Review of Financial Studies, 23(4):1467-1502.

Ranaldo, Angelo, and Laurent Favre Sr. 2005. How to Price Hedge Funds: From Two- to Four-Moment CAPM. SSRN eLibrary.

Scott, Robert C., and Philip A. Horvath. 1980. On the Direction of Preference for Moments of Higher Order than the Variance. Journal of Finance 35(4):915-919.

Examples

```
data(managers)
Return.centered(managers[,1:3,drop=FALSE])
```

`Return.clean` *clean returns in a time series to to provide more robust risk estimates*

Description

A function that provides access to multiple methods for cleaning outliers from return data.

Usage

```
Return.clean(R, method = c("none", "boudt", "geltner"), alpha = 0.01, ...)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`method`	one of "none", "boudt", which applies the function `clean.boudt` or "geltner" which applies the function `Return.Geltner`to R
`alpha`	the percentage of outliers you want to clean
`...`	additional parameters passed into the underlying cleaning function

Details

This is a wrapper for offering multiple data cleaning methods for data objects containing returns.

The primary value of data cleaning lies in creating a more robust and stable estimation of the distribution generating the large majority of the return data. The increased robustness and stability of the estimated moments using cleaned data should be used for portfolio construction. If an investor wishes to have a more conservative risk estimate, cleaning may not be indicated for risk monitoring.

In actual practice, it is probably best to back-test the results of both cleaned and uncleaned series to see what works best when forecasting risk with the particular combination of assets under consideration.

In this version, only one method is supported. See `clean.boudt` for more details.

Author(s)

Peter Carl

See Also

`clean.boudt`
`Return.Geltner`

Examples

```
## Not run:  # CRAN doesn't like how long this takes (>5 secs)
data(managers)
head(Return.clean(managers[,1:4]),n=20)
chart.BarVaR(managers[,1,drop=FALSE], show.clean=TRUE, clean="boudt", lwd=2, methods="ModifiedVaR")

## End(Not run)
```

`Return.convert` *Convert coredata content from one type of return to another*

Description

This function takes an xts object, and using its attribute information, will convert information in the object into the desired output, selected by the user. For example, all combinations of moving from one of 'discrete', 'log', 'difference' and 'level', to another different data type (from the same list) are permissible.

Usage

```
Return.convert(
  R,
  destinationType = c("discrete", "log", "difference", "level"),
  seedValue = NULL,
  initial = TRUE
)
```

Arguments

`R`	an xts object
`destinationType`	one of 'discrete', 'log', 'difference' or 'level'
`seedValue`	a numeric scalar indicating the (usually initial) index level or price of the series
`initial`	(default TRUE) a TRUE/FALSE flag associated with 'seedValue', indicating if this value is at the begginning of the series (TRUE) or at the end of the series (FALSE)

Author(s)

Erol Biceroglu

See Also

`Return.calculate`

Examples

```
# TBD
```

`Return.cumulative` *calculate a compounded (geometric) cumulative return*

Description

This is a useful function for calculating cumulative return over a period of time, say a calendar year. Can produce simple or geometric return.

Usage

```
Return.cumulative(R, geometric = TRUE)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`geometric`	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE

Details

product of all the individual period returns

$$(1 + r_1)(1 + r_2)(1 + r_3)\dots(1 + r_n) - 1 = prod(1 + R) - 1$$

Note that this function calculates the total return over the entire period as a single scalar value. If you need the time series of cumulative wealth (i.e. a wealth index), you should use `cumprod(1+R)` or `cumprod(1+R)-1`.

Author(s)

Peter Carl

References

Bacon, Carl. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004. p. 6

For time series of cumulative wealth, see: `cumprod` `chart.CumReturns`

See Also

`Return.annualized`

Examples

```
data(managers)
Return.cumulative(managers[, 1, drop = FALSE])
Return.cumulative(managers[, 1:8])
Return.cumulative(managers[, 1:8], geometric = FALSE)
```

`Return.excess` *Calculates the returns of an asset in excess of the given risk free rate*

Description

Calculates the returns of an asset in excess of the given "risk free rate" for the period.

Usage

```
Return.excess(R, Rf = 0)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rf`	risk free rate, in same period as your returns, or as a single digit average

Details

Ideally, your risk free rate will be for each period you have returns observations, but a single average return for the period will work too.

Mean of the period return minus the period risk free rate

$$\overline{(R_a - R_f)}$$

OR

mean of the period returns minus a single numeric risk free rate

$$\overline{R_a} - R_f$$

Note that while we have, in keeping with common academic usage, assumed that the second parameter will be a risk free rate, you may also use any other timeseries as the second argument. A common alteration would be to use a benchmark to produce excess returns over a specific benchmark, as demonstrated in the examples below.

Author(s)

Peter Carl

References

Bacon, Carl. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004. p. 47-52

Examples

```
data(managers)
head(Return.excess(managers[, 1, drop = FALSE], managers[, 10, drop = FALSE]))
head(Return.excess(managers[, 1, drop = FALSE], .04 / 12))
head(Return.excess(managers[, 1:6], managers[, 10, drop = FALSE]))
head(Return.excess(managers[, 1, drop = FALSE], managers[, 8, drop = FALSE]))
```

Return.Geltner *calculate Geltner liquidity-adjusted return series*

Description

David Geltner developed a method to remove estimating or liquidity bias in real estate index returns. It has since been applied with success to other return series that show autocorrelation or illiquidity effects.

Usage

```
Return.Geltner(Ra, ...)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters

Details

The theory is that by correcting for autocorrelation, you are uncovering a "true" return from a series of observed returns that contain illiquidity or manual pricing effects.

The Geltner autocorrelation adjusted return series may be calculated via:

$$R_G = \frac{R_t - (R_{t-1} \cdot \rho_1)}{1 - \rho_1}$$

where ρ_1 is the first-order autocorrelation of the return series R_a and R_t is the return of R_a at time t and R_{t-1} is the one-period lagged return.

Author(s)

Brian Peterson

References

"Edhec Funds of Hedge Funds Reporting Survey : A Return-Based Approach to Funds of Hedge Funds Reporting",Edhec Risk and Asset Management Research Centre, January 2005,p. 27

Geltner, David, 1991, Smoothing in Appraisal-Based Returns, Journal of Real Estate Finance and Economics, Vol.4, p.327-345.

Geltner, David, 1993, Estimating Market Values from Appraised Values without Assuming an Efficient Market, Journal of Real Estate Research, Vol.8, p.325-345.

Examples

```
data(managers)
head(Return.Geltner(managers[,1:3]),n=20)
```

Return.locScaleRob *Robust Filter for Time Series Returns*

Description

`Return.locScaleRob` returns the data after passing through a robust location and scale filter.

Usage

```
Return.locScaleRob(R, alpha.robust = 0.05, normal.efficiency = 0.99, ...)
```

Arguments

`R`	Data of returns for assets or portfolios.
`alpha.robust`	Tuning parameter for the robust filter.
`normal.efficiency`	Normal efficiency for robust filter.
`...`	any other passthrough parameters

Value

A vector of the cleaned data.

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")
# Cleaning the returns time series for manager data
if (suppressMessages(suppressWarnings(
  requireNamespace("RobStatTM", quietly = TRUE)))) {
  # CRAN requires conditional execution
  outRob <- suppressMessages(Return.locScaleRob(edhec$CA))
}
```

Return.portfolio *Calculate weighted returns for a portfolio of assets*

Description

Using a time series of returns and any regular or irregular time series of weights for each asset, this function calculates the returns of a portfolio with the same periodicity of the returns data.

Usage

```
Return.portfolio(
  R,
  weights = NULL,
  wealth.index = FALSE,
  contribution = FALSE,
  geometric = TRUE,
  rebalance_on = c(NA, "years", "quarters", "months", "weeks", "days"),
  value = 1,
  verbose = FALSE,
  ...,
  rebal_cost = 0,
  full_investment = FALSE
)
```

Arguments

R	An xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
weights	A time series or single-row matrix/vector containing asset weights, as decimal percentages. When passed as a time series, weights are treated as end-of-period targets that take effect at the start of the subsequent period. See Details below.
wealth.index	TRUE/FALSE whether to return a wealth index. Default FALSE

contribution	if contribution is TRUE, add the weighted return contributed by the asset in a given period. Default FALSE
geometric	utilize geometric chaining (TRUE) or simple/arithmetic (FALSE) to aggregate returns. Default TRUE.
rebalance_on	Default "none"; alternatively "daily" "weekly" "monthly" "annual" to specify calendar-period rebalancing supported by endpoints. Ignored if weights is an xts object that specifies the rebalancing dates.
value	The beginning of period total portfolio value. This is used for calculating position value.
verbose	If verbose is TRUE, return a list of intermediary calculations. See Details below.
...	any other passthru parameters. Not currently used.
rebal_cost	proportional transaction cost applied to the change in weights when rebalancing. Default 0.
full_investment	if full_investment is TRUE, forces a target weight rebalance when security returns become unavailable (NA), excluding such securities and proportionately reallocating their weights to the remaining assets on the boundary day before the return is NA. Default FALSE

Details

By default, this function calculates the time series of portfolio returns given asset returns and weights. In verbose mode, the function returns a list of intermediary calculations that users may find helpful, including both asset contribution and asset value through time.

When asset return and weights are matched by period, contribution is simply the weighted return of the asset. c_i = w_i * R_i Contributions are summable across the portfolio to calculate the total portfolio return.

Contribution cannot be aggregated through time. For example, say we have an equal weighted portfolio of five assets with monthly returns. The geometric return of the portfolio over several months won't match any aggregation of the individual contributions of the assets, particularly if any rebalancing was done during the period.

To aggregate contributions through time such that they are summable to the geometric returns of the portfolio, the calculation must track changes in the notional value of the assets and portfolio. For example, contribution during a quarter will be calculated as the change in value of the position through those three months, divided by the original value of the portfolio. Approaching it this way makes the calculation robust to weight changes as well. c_pi = V_(t-p)i - V_t)/V_ti

If the user does not specify weights, an equal weight portfolio is assumed. Alternatively, a vector or single-row matrix of weights that matches the length of the asset columns may be specified. In either case, if no rebalancing period is specified, the weights will be applied at the beginning of the asset time series and no further rebalancing will take place. If a rebalancing period is specified, the portfolio will be rebalanced to the starting weights at the interval specified.

Note that if weights is an xts object, then any value passed to rebalance_on is ignored. The weights object specifies the rebalancing dates, therefore a regular rebalancing frequency provided via rebalance_on is not needed and ignored.

Return.portfolio will work only on daily or lower frequencies. If you are rebalancing intraday, you should be using a trades/prices framework like the blotter package, not a weights/returns framework.

Irregular rebalancing can be done by specifying a time series of weights. The function uses the date index of the weights for xts-style subsetting of rebalancing periods.

Weights specified for rebalancing should be thought of as "end-of-period" weights. Rebalancing periods can be thought of as taking effect immediately after the close of the bar. So, a March 31 rebalancing date will actually be in effect for April 1. A December 31 rebalancing date will be in effect on Jan 1, and so forth. This convention was chosen because it fits with common usage, and because it simplifies xts Date subsetting via endpoints.

In verbose mode, the function returns a list of data and intermediary calculations.

returns: The portfolio returns.

contribution: The per period contribution to portfolio return of each asset. Contribution is calculated as BOP weight times the period's return divided by BOP value. Period contributions are summed across the individual assets to calculate portfolio return

BOP.Weight: Beginning of Period (BOP) Weight for each asset. An asset's BOP weight is calculated using the input weights (or assumed weights, see below) and rebalancing parameters given. The next period's BOP weight is either the EOP weights from the prior period or input weights given on a rebalance period.

EOP.Weight: End of Period (BOP) Weight for each asset. An asset's EOP weight is the sum of the asset's BOP weight and contribution for the period divided by the sum of the contributions and initial weights for the portfolio.

BOP.Value: BOP Value for each asset. The BOP value for each asset is the asset's EOP value from the prior period, unless there is a rebalance event. If there is a rebalance event, the BOP value of the asset is the rebalance weight times the EOP value of the portfolio. That effectively provides a zero-transaction cost change to the position values as of that date to reflect the rebalance. Note that the sum of the BOP values of the assets is the same as the prior period's EOP portfolio value.

EOP.Value: EOP Value for each asset. The EOP value is for each asset is calculated as (1 + asset return) times the asset's BOP value. The EOP portfolio value is the sum of EOP value across assets.

To calculate BOP and EOP position value, we create an index for each position. The sum of that value across assets represents an indexed value of the total portfolio. Note that BOP and EOP position values are only computed when geometric = TRUE.

From the value calculations, we can calculate different aggregations through time for the asset contributions. Those are calculated as the EOP asset value less the BOP asset value; that quantity is divided by the BOP portfolio value. Across assets, those will sum to equal the geometric chained returns of the portfolio for that same time period. The function does not do this directly, however.

Value

returns a time series of returns weighted by the weights parameter, or a list that includes intermediate calculations

Note

This function was previously two functions: `Return.portfolio` and `Return.rebalancing`. Both function names are still exported, but the code is now common, and `Return.portfolio` is probably to be preferred.

Author(s)

Peter Carl, Ross Bennett, Brian Peterson

References

Bacon, C. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004. Chapter 2

See Also

`Return.calculate` `endpoints`

Examples

```
data(edhec)
Return.portfolio(edhec["1997", 1:5], rebalance_on = "quarters") # returns time series
Return.portfolio(edhec["1997", 1:5], rebalance_on = "quarters", verbose = TRUE) # returns list
# with a weights object
data(weights) # rebalance at the beginning of the year to various weights through time
chart.StackedBar(weights)
x <- Return.portfolio(edhec["2000::", 1:11], weights = weights, verbose = TRUE)
chart.CumReturns(x$returns)
chart.StackedBar(x$BOP.Weight)
chart.StackedBar(x$BOP.Value)

# with a rebalancing cost
Return.portfolio(edhec["1997", 1:5], rebalance_on = "quarters", rebal_cost = 0.002) # 20 bps

# add a transaction cost to the rebalancing
x_cost <- Return.portfolio(edhec["2000::", 1:11], weights = weights, verbose = TRUE,
                           rebal_cost = 0.01)
chart.CumReturns(x_cost$returns)
chart.StackedBar(x_cost$BOP.Weight)
chart.StackedBar(x_cost$BOP.Value)
```

`Return.read` *Read returns data with different date formats*

Description

A simple wrapper of read.zoo with some defaults for different date formats and xts conversion

Usage

```
Return.read(
  filename = stop("Please specify a filename"),
  frequency = c("d", "m", "q", "i", "o"),
  format.in = c("ISO8601", "excel", "oo", "gnumeric"),
  sep = ",",
  header = TRUE,
  check.names = FALSE,
  ...
)
```

Arguments

filename	the name of the file to be read
frequency	• "d" sets as a daily timeseries using as.Date, • "m" sets as monthly timeseries using as.yearmon, • "q" sets as a quarterly timeseries using as.yearqtr, and • "i" sets as irregular timeseries using as.POSIXct
format.in	says how the data being read is formatted. Although the default is set to the ISO 8601 standard (which can also be set as " most spreadsheets have less sensible date formats as defaults. See below.
sep	separator, default is ","
header	a logical value indicating whether the file contains the names of the variables as its first line.
check.names	logical. If TRUE then the names of the variables in the data frame are checked to ensure that they are syntactically valid variable names. If necessary they are adjusted (by make.names) so that they are, and also to ensure that there are no duplicates. See read.table
...	passes through any other parameters to read.zoo

Details

The parameter 'format.in' takes several values, including:

excel default date format for MS Excel spreadsheet csv format, which is "%m/%d/%Y"

oo default date format for OpenOffice spreadsheet csv format, "%m/%d/%y", although this may be operating system dependent

gnumeric default date format for Gnumeric spreadsheet, which is "%d-%b-%Y"

... alternatively, any specific format may be passed in, such as "%M/%y"

Author(s)

Peter Carl

See Also

read.zoo, read.table

Examples

```
        ## Not run:
        Return.read("managers.cvs", frequency="d")

    ## End(Not run)
```

`Return.relative` *calculate the relative return of one asset to another*

Description

Calculates the ratio of the cumulative performance for two assets through time.

Usage

```
Return.relative(Ra, Rb, ...)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return object for the benchmark asset
...	ignored

Value

xts or other time series of relative return

Author(s)

Peter Carl

See Also

chart.RelativePerformance

Examples

```
data(managers)
head(Return.relative(managers[,1:3], managers[,8,drop=FALSE]),n=20)
```

RPESE.control *Controls Function for the Computation of Standard Errors for Risk and Performance estimators*

Description

RPESE.controls sets the different control parameters used in the compuation of standard errors for risk and performance estimators.

Usage

```
RPESE.control(
 estimator = c("Mean", "SD", "VaR", "ES", "SR", "DSR", "SoR", "ESratio", "VaRratio",
    "SoR", "LPM", "OmegaRatio", "SemiSD", "RachevRatio")[1],
  se.method = NULL,
  cleanOutliers = NULL,
  fitting.method = NULL,
  freq.include = NULL,
  freq.par = NULL,
  a = NULL,
  b = NULL
)
```

Arguments

estimator	Risk or performance estimator used to set default control parameters. Default is "Mean" estimator.
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".
freq.include	Frequency domain inclusion criteria. Must be one of "All" (default), "Decimate" or "Truncate."
freq.par	Percentage of the frequency used if "freq.include" is "Decimate" or "Truncate." Default is 0.5.
a	First adaptive method parameter.
b	Second adaptive method parameter.

Value

A list of different control parameters for the computation of standard errors.

Author(s)

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

Examples

```
# Case where we want the default parameters for the ES
ES.control <- RPESE.control(estimator="ES")
# Case where we also set additional parameters manually
ES.control.2 <- RPESE.control(estimator="ES", se.method=c("IFcor", "BOOTiid"),
                              cleanOutliers=TRUE, freq.include="Decimate")
# These lists can be passed onto the functions (e.g., ES) to control the parameters
# for computing and returning standard errors.
```

Selectivity *Selectivity of the return distribution*

Description

Selectivity is the same as Jensen's alpha

Usage

```
Selectivity(Ra, Rb, Rf = 0, ...)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
Rf	risk free rate, in same period as your returns
...	any other passthru parameters

Details

$$Selectivity = r_p - r_f - \beta_p * (b - r_f)$$

where r_f is the risk free rate, β_r is the regression beta, r_p is the portfolio return and b is the benchmark return

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.78

Examples

```
data(portfolio_bacon)
print(Selectivity(portfolio_bacon[,1], portfolio_bacon[,2])) #expected -0.0141

data(managers)
print(Selectivity(managers['2002',1], managers['2002',8]))
print(Selectivity(managers['2002',1:5], managers['2002',8]))
```

SFM.alpha *Calculate single factor model (CAPM) alpha*

Description

This is a wrapper for calculating a single factor model (CAPM) alpha.

Usage

```
SFM.alpha(
  Ra,
  Rb,
  Rf = 0,
  ...,
  digits = 3,
  benchmarkCols = T,
  method = "LS",
  family = "mopt",
  warning = T
)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
Rf	risk free rate, in same period as your returns
...	Other parameters like max.it or bb specific to lmrobdetMM regression.
digits	Number of digits to round the results to. Defaults to 3.
benchmarkCols	Boolean to show the benchmarks as columns. Defaults to TRUE.
method	string representing linear regression model, "LS" for Least Squares and "Robust" for robust. Defaults to "LS"
family	If method == "Robust": This is a string specifying the name of the family of loss function to be used (current valid options are "bisquare", "opt" and "mopt"). Incomplete entries will be matched to the current valid options. Defaults to "mopt". Else: the parameter is ignored
warning	Boolean to show warnings or not. Defaults to TRUE.

Details

"Alpha" purports to be a measure of a manager's skill by measuring the portion of the managers returns that are not attributable to "Beta", or the portion of performance attributable to a benchmark.

While the classical CAPM has been almost completely discredited by the literature, it is an example of a simple single factor model, comparing an asset to any arbitrary benchmark.

Author(s)

Dhairya Jain, Peter Carl

References

Sharpe, W.F. Capital Asset Prices: A theory of market equilibrium under conditions of risk. *Journal of finance*, vol 19, 1964, 425-442.
Ruppert, David. *Statistics and Finance, an Introduction*. Springer. 2004.

See Also

CAPM.beta CAPM.utils

Examples

```
# First we load the data
    data(managers)
    SFM.alpha(managers[, "HAM1"], managers[, "SP500 TR"], Rf = managers[, "US 3m TR"])
    SFM.alpha(managers[,1:3], managers[,8:10], Rf=.035/12)
  # SFM.alpha(managers[,1], managers[,8:10], Rf=.035/12, benchmarkCols=FALSE) # other variations
if(requireNamespace("RobStatTM", quietly = TRUE)) { # CRAN requires conditional execution
   alphas <- SFM.alpha(managers[,1:6],
                  managers[,8:10],
                 Rf=.035/12, method="Robust",
                 family="opt", bb=0.25, max.it=200, digits=4)
  alphas["HAM1", ]
  alphas[, "Alpha : SP500 TR"]
} # CRAN can have issues with RobStatTM
```

SFM.beta *Calculate single factor model (CAPM) beta*

Description

The single factor model or CAPM Beta is the beta of an asset to the variance and covariance of an initial portfolio. Used to determine diversification potential.

Usage

```
SFM.beta(
  Ra,
  Rb,
  Rf = 0,
  ...,
  digits = 3,
  benchmarkCols = T,
  method = "LS",
  family = "mopt",
  warning = T
)

SFM.beta.bull(
  Ra,
  Rb,
  Rf = 0,
  ...,
  digits = 3,
  benchmarkCols = T,
  method = "LS",
  family = "mopt"
)

SFM.beta.bear(
  Ra,
  Rb,
  Rf = 0,
  ...,
  digits = 3,
  benchmarkCols = T,
  method = "LS",
  family = "mopt"
)

TimingRatio(Ra, Rb, Rf = 0, ...)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
Rf	risk free rate, in same period as your returns
...	Other parameters like max.it or bb specific to lmrobdetMM regression.
digits	(Optional): Number of digits to round the results to. Defaults to 3.
benchmarkCols	(Optional): Boolean to show the benchmarks as columns. Defaults to TRUE.
method	(Optional): string representing linear regression model, "LS" for Least Squares and "Robust" for robust. Defaults to "LS

family	(Optional): If method == "Robust": This is a string specifying the name of the family of loss function to be used (current valid options are "bisquare", "opt" and "mopt"). Incomplete entries will be matched to the current valid options. Defaults to "mopt". Else: the parameter is ignored
warning	(Optional): Boolean to show warnings or not. Defaults to TRUE.

Details

This function uses a linear intercept model to achieve the same results as the symbolic model used by BetaCoVariance

$$\beta_{a,b} = \frac{CoV_{a,b}}{\sigma_b} = \frac{\sum((R_a - \bar{R_a})(R_b - \bar{R_b}))}{\sum(R_b - \bar{R_b})^2}$$

Ruppert(2004) reports that this equation will give the estimated slope of the linear regression of R_a on R_b and that this slope can be used to determine the risk premium or excess expected return (see Eq. 7.9 and 7.10, p. 230-231).

Two other functions apply the same notion of best fit to positive and negative market returns, separately. The SFM.beta.bull is a regression for only positive market returns, which can be used to understand the behavior of the asset or portfolio in positive or 'bull' markets. Alternatively, SFM.beta.bear provides the calculation on negative market returns.

The TimingRatio may help assess whether the manager is a good timer of asset allocation decisions. The ratio, which is calculated as

$$TimingRatio = \frac{\beta^+}{\beta^-}$$

is best when greater than one in a rising market and less than one in a falling market.

While the classical CAPM has been almost completely discredited by the literature, it is an example of a simple single factor model, comparing an asset to any arbitrary benchmark.

Author(s)

Dhairya Jain, Peter Carl

References

Sharpe, W.F. Capital Asset Prices: A theory of market equilibrium under conditions of risk. *Journal of finance*, vol 19, 1964, 425-442.
Ruppert, David. *Statistics and Finance, an Introduction*. Springer. 2004.
Bacon, Carl. *Practical portfolio performance measurement and attribution*. Wiley. 2004.

See Also

BetaCoVariance SFM.alpha CAPM.utils

Examples

```
data(managers)

SFM.beta(managers[, "HAM1"], managers[, "SP500 TR"], Rf = managers[, "US 3m TR"])
SFM.beta(managers[,1:3], managers[,8:10], Rf=.035/12)
SFM.beta(managers[,1], managers[,8:10], Rf=.035/12, benchmarkCols=FALSE)

SFM.beta.bull(managers[, "HAM2"],
         managers[, "SP500 TR"],
         Rf = managers[, "US 3m TR"])

SFM.beta.bear(managers[, "HAM2"],
         managers[, "SP500 TR"],
         Rf = managers[, "US 3m TR"])

TimingRatio(managers[, "HAM2"],
       managers[, "SP500 TR"],
       Rf = managers[, "US 3m TR"])

if(requireNamespace("RobStatTM", quietly = TRUE)) { # CRAN requires conditional execution
  betas <- SFM.beta(managers[,1:6],
               managers[,8:10],
               Rf=.035/12, method="Robust",
               family="opt", bb=0.25, max.it=200, digits=4)
 betas["HAM1", ]
 betas[, "Beta : SP500 TR"]

  SFM.beta.bull(managers[, "HAM2"],
               managers[, "SP500 TR"],
          Rf = managers[, "US 3m TR"],
          method="Robust")

  SFM.beta.bear(managers[, "HAM2"],
          managers[, "SP500 TR"],
          Rf = managers[, "US 3m TR"],
          method="Robust")

 TimingRatio(managers[, "HAM2"],
          managers[, "SP500 TR"],
         Rf = managers[, "US 3m TR"],
         method="Robust", family="mopt")

} # CRAN can have issues with RobStatTM

chart.Regression(managers[, "HAM2"],
            managers[, "SP500 TR"],
            Rf = managers[, "US 3m TR"],
            fit="conditional",
            main="Conditional Beta")
```

`SFM.coefficients` *Calculate single factor model alpha and beta coefficients*

Description

The single factor model is the beta of an asset to the variance and covariance of an initial portfolio. Used to determine diversification potential. "Alpha" purports to be a measure of a manager's skill by measuring the portion of the managers returns that are not attributable to "Beta", or the portion of performance attributable to a benchmark.

Usage

```
SFM.coefficients(
  Ra,
  Rb,
  Rf = 0,
  subset = TRUE,
  ...,
  method = "Robust",
  family = "mopt",
  digits = 3,
  benchmarkCols = T,
  Model = F,
  warning = T
)
```

Arguments

`Ra`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rb`	return vector of the benchmark asset
`Rf`	risk free rate, in same period as your returns
`subset`	a logical vector representing the set of observations to be used in regression
`...`	Other parameters like max.it or bb specific to lmrobdetMM regression. Another interesting parameter is round.by.alphas, and round.by.betas which can be set to round off the returned values, otherwise the values
`method`	(Optional): string representing linear regression model, "LS" for Least Squares and "Robust" for robust. Defaults to "LS
`family`	(Optional): If method == "Robust": This is a string specifying the name of the family of loss function to be used (current valid options are "bisquare", "opt" and "mopt"). Incomplete entries will be matched to the current valid options. Defaults to "mopt". Else: the parameter is ignored
`digits`	(Optional): Number of digits to round the results to. Defaults to 3.
`benchmarkCols`	(Optional): Boolean to show the benchmarks as columns. Defaults to TRUE.
`Model`	(Optional): Boolean to return the fitted model. Defaults to FALSE
`warning`	(Optional): Boolean to show warnings or not. Defaults to TRUE.

Details

This function is designed to be used a a wrapper for SFM's regression models. Using this, one can easily fit different types of linear models like Ordinary Least Squares or Robust Estimators like Huber.

$$\beta_{a,b} = \frac{CoV_{a,b}}{\sigma_b} = \frac{\sum((R_a - \bar{R_a})(R_b - \bar{R_b}))}{\sum(R_b - \bar{R_b})^2}$$

Ruppert(2004) reports that this equation will give the estimated slope of the linear regression of R_a on R_b and that this slope can be used to determine the risk premium or excess expected return (see Eq. 7.9 and 7.10, p. 230-231).

Author(s)

Dhairya Jain

References

Sharpe, W.F. Capital Asset Prices: A theory of market equilibrium under conditions of risk. *Journal of finance*, vol 19, 1964, 425-442.
Ruppert, David. *Statistics and Finance, an Introduction*. Springer. 2004.
Bacon, Carl. *Practical portfolio performance measurement and attribution*. Wiley. 2004.

See Also

BetaCoVariance SFM.alpha CAPM.utils SFM.beta

Examples

```
if(requireNamespace("RobStatTM", quietly = TRUE)) { # CRAN requires conditional execution
  data(managers)
  SFM.coefficients(managers[,1], managers[,8])
  SFM.coefficients(managers[,1:6], managers[,8:9], Rf = managers[,10])
 SFM.coefficients(managers[,1:6], managers[,8:9],
     Rf=.035/12, method="Robust",
     family="mopt", bb=0.25,
     max.it=200)
} # CRAN requires conditional execution
```

SFM.fit.models *Compare SFM estimated using robust estimators with that estimated by OLS*

Description

This function provies a simple plug and play option for user to compare the SFM estimates by lm and lmrobdetMM functions, using the fit.models framework. This will allow for an easier comparison using charts and tables

Usage

```
SFM.fit.models(
  Ra,
  Rb,
  Rf = 0,
  family = "mopt",
  which.plots = NULL,
  plots = TRUE
)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
Rf	risk free rate, in same period as your returns
family	(Optional): If method == "Robust": This is a string specifying the name of the family of loss function to be used (current valid options are "bisquare", "opt" and "mopt"). Incomplete entries will be matched to the current valid options. Defaults to "mopt". Else: the parameter is ignored
which.plots	specify the plot numbers that you want to see. Default option is that program will ask you to choose a plot
plots	Boolean to output plots after the function. Defaults to TRUE

Author(s)

Dhairya Jain

Examples

```
## Not run:
# First we load the data
    data(managers)
    mgrs <- managers["2002/"]  # So that all have managers have complete history
    names(mgrs)[7:10] <- c("LSEQ","SP500","Bond10Yr","RF") # Short names for last 3

    # NOTE: Use plots=TRUE to be able to see plots

    fitModels <- SFM.fit.models(mgrs$HAM1,mgrs$SP500,Rf = mgrs$RF, plots=FALSE)
    coef(fitModels)
    summary(fitModels)
    class(fitModels)

    SFM.fit.models(mgrs$HAM1, mgrs$SP500, Rf = mgrs$RF, plots=FALSE)

    SFM.fit.models(mgrs[,6], mgrs[,8], Rf=.035/12, plots=FALSE)

    SFM.fit.models(mgrs$HAM6, mgrs$SP500, Rf=.035/12, family = "mopt",
                   which.plots = c(1,2), plots=FALSE)
```

```
SFM.fit.models(mgrs$HAM2, mgrs$SP500, Rf = mgrs$RF, family = "opt",
               plots=FALSE)

## End(Not run)
```

SharpeRatio *calculate a traditional or modified Sharpe Ratio of Return over StdDev or VaR or ES*

Description

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.

Usage

```
SharpeRatio(
  R,
  Rf = 0,
  p = 0.95,
  FUN = c("StdDev", "VaR", "ES", "SemiSD"),
  weights = NULL,
  annualize = FALSE,
  geometric = FALSE,
  SE = FALSE,
  SE.control = NULL,
  ...
)

SharpeRatio.modified(
  R,
  Rf = 0,
  p = 0.95,
  FUN = c("StdDev", "VaR", "ES"),
  weights = NULL,
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rf	risk free rate, in same period as your returns
p	confidence level for calculation, default p=.95
FUN	one of "StdDev" or "VaR" or "ES" to use as the denominator
weights	portfolio weighting vector, default NULL, see Details in VaR
annualize	if TRUE, annualize the measure, default FALSE

geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default FALSE
SE	TRUE/FALSE whether to ouput the standard errors of the estimates of the risk measures, default FALSE.
SE.control	Control parameters for the computation of standard errors. Should be done using the RPESE.control function.
...	any other passthru parameters to the VaR or ES functions

Details

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

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.

The Sharpe Ratio can be used to measure both 'excess return' (over a risk-free rate) and 'differential return' (excess return over a benchmark).

A number of papers now recommend using a "modified Sharpe" ratio using a Modified Cornish-Fisher VaR or CVaR/Expected Shortfall as the measure of Risk.

We have extended this concept to create multivariate modified Sharpe-like Ratios for standard deviation, Gaussian VaR, modified VaR, Gaussian Expected Shortfall, and modified Expected Shortfall. See VaR and ES. You can pass additional arguments to VaR and ES via ...The most important is probably the 'method' argument/

Most recently, we have added Downside Sharpe Ratio (DSR) (see DownsideSharpeRatio), a short name for what Ziemba (2005) called the "Symmetric Downside Risk Sharpe Ratio" and is defined as the ratio of the mean return to the square root of lower semivariance:

$$\frac{\overline{(R_a - R_f)}}{\sqrt{2} SemiSD(R_a)}$$

.

This function returns a traditional or modified Sharpe ratio for the same periodicity of the data being input (e.g., monthly data -> monthly SR)

Author(s)

Brian G. Peterson

References

Sharpe, W.F. The Sharpe Ratio,*Journal of Portfolio Management*,Fall 1994, 49-58.

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

Ziemba, W. T. (2005). The symmetric downside-risk Sharpe ratio. The Journal of Portfolio Management, 32(1), 108-122.

Jacquier, E., Kane, A., Marcus, A. [2003]. Geometric Mean or Arithmetic Mean: A Reconsideration. Financial Analysts Journal, November/December 2003, p. 46-53.

See Also

SharpeRatio.annualized
InformationRatio
TrackingError
ActivePremium
SortinoRatio
VaR
ES

Examples

```
data(managers)
SharpeRatio(managers[, 1, drop = FALSE], Rf = .035 / 12, FUN = "StdDev")
SharpeRatio(managers[, 1, drop = FALSE], Rf = managers[, 10, drop = FALSE], FUN = "StdDev")
SharpeRatio(managers[, 1:6], Rf = .035 / 12, FUN = "StdDev")
SharpeRatio(managers[, 1:6], Rf = managers[, 10, drop = FALSE], FUN = "StdDev")



data(edhec)
SharpeRatio(edhec[, 6, drop = FALSE], FUN = "VaR")
SharpeRatio(edhec[, 6, drop = FALSE], Rf = .04 / 12, FUN = "VaR")
SharpeRatio(edhec[, 6, drop = FALSE], Rf = .04 / 12, FUN = "VaR", method = "gaussian")
SharpeRatio(edhec[, 6, drop = FALSE], FUN = "ES")

# and all the methods
SharpeRatio(managers[, 1:9], Rf = managers[, 10, drop = FALSE], FUN = c("StdDev", "VaR", "ES"))
SharpeRatio(edhec, Rf = .04 / 12, FUN = c("StdDev", "VaR", "ES"))
```

SharpeRatio.annualized

calculate annualized Sharpe Ratio

Description

The Sharpe Ratio is a risk-adjusted measure of return that uses standard deviation to represent risk.

Usage

```
SharpeRatio.annualized(R, Rf = 0, scale = NA, geometric = FALSE, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rf	risk free rate, in same period as your returns
scale	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default FALSE
...	any other passthru parameters

Details

The Sharpe ratio is simply the return per unit of risk (represented by variance). The higher the Sharpe ratio, the better the combined performance of "risk" and return.

This function annualizes the number based on the scale parameter.

$$\frac{\sqrt[n]{prod(1+R_a)^{scale}}-1}{\sqrt{scale}\cdot\sqrt{\sigma}}$$

Using an annualized Sharpe Ratio is useful for comparison of multiple return streams. The annualized Sharpe ratio is computed by dividing the annualized mean monthly excess return by the annualized monthly standard deviation of excess return.

William Sharpe now recommends Information Ratio preferentially to the original Sharpe Ratio.

Author(s)

Peter Carl

References

Sharpe, W.F. The Sharpe Ratio,*Journal of Portfolio Management*,Fall 1994, 49-58.

See Also

SharpeRatio
InformationRatio
TrackingError
ActivePremium
SortinoRatio

Examples

```
data(managers)
SharpeRatio.annualized(managers[, 1, drop = FALSE], Rf = .035 / 12)
SharpeRatio.annualized(managers[, 1, drop = FALSE], Rf = managers[, 10, drop = FALSE])
SharpeRatio.annualized(managers[, 1:6], Rf = .035 / 12)
SharpeRatio.annualized(managers[, 1:6], Rf = managers[, 10, drop = FALSE])
SharpeRatio.annualized(managers[, 1:6], Rf = managers[, 10, drop = FALSE], geometric = FALSE)
```

ShrinkageMoments *Functions for calculating shrinkage-based comoments of financial time series*

Description

calculates covariance, coskewness and cokurtosis matrices using linear shrinkage between the sample estimator and a structured estimator

Usage

```
M2.shrink(R, targets = 1, f = NULL)

M3.shrink(R, targets = 1, f = NULL, unbiasedMSE = FALSE, as.mat = TRUE)

M4.shrink(R, targets = 1, f = NULL, as.mat = TRUE)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
targets	vector of integers selecting the target matrices to shrink to. The first four structures are, in order: 'independent marginals', 'independent and identical marginals', 'observed 1-factor model' and 'constant correlation'. See Details.
f	vector or matrix with observations of the factor, to be used with target 3. See Details.
unbiasedMSE	TRUE/FALSE whether to use a correction to have an unbiased estimator for the marginal skewness values, in case of targets 1 and/or 2, default FALSE
as.mat	TRUE/FALSE whether to return the full moment matrix or only the vector with the unique elements (the latter is advised for speed), default TRUE

Details

The coskewness and cokurtosis matrices are defined as the matrices of dimension p x p^2 and p x p^3 containing the third and fourth order central moments. They are useful for measuring nonlinear dependence between different assets of the portfolio and computing modified VaR and modified ES of a portfolio.

Shrinkage estimation for the covariance matrix was popularized by Ledoit and Wolf (2003, 2004). An extension to coskewness and cokurtosis matrices by Martellini and Ziemann (2010) uses the 1-factor and constant-correlation structured comoment matrices as targets. In Boudt, Cornilly and Verdonck (2017) the framework of single-target shrinkage for the coskewness and cokurtosis matrices is extended to a multi-target setting, making it possible to include several target matrices at once. Also, an option to enhance small sample performance for coskewness estimation was proposed, resulting in the option 'unbiasedMSE' present in the 'M3.shrink' function.

The first four target matrices of the 'M2.shrink', 'M3.shrink' and 'M4.shrink' correspond to the models 'independent marginals', 'independent and identical marginals', 'observed 1-factor model'

and 'constant correlation'. Coskewness shrinkage includes two more options, target 5 corresponds to the latent 1-factor model proposed in Simaan (1993) and target 6 is the coskewness matrix under central-symmetry, a matrix full of zeros. For more details on the targets, we refer to Boudt, Cornilly and Verdonck (2017) and the supplementary appendix.

If f is a matrix containing multiple factors, then the shrinkage estimator will use each factor in a seperate single-factor model and use multi-target shrinkage to all targets matrices at once.

Author(s)

Dries Cornilly

References

Boudt, Kris, Brian G. Peterson, and Christophe Croux. 2008. Estimation and Decomposition of Downside Risk for Portfolios with Non-Normal Returns. Journal of Risk. Winter.

Boudt, Kris, Dries Cornilly and Tim Verdonck. 2020 A Coskewness Shrinkage Approach for Estimating the Skewness of Linear Combinations of Random Variables. Journal of Financial Econometrics, 18(1), 1-23.

Ledoit, Olivier and Michael Wolf. 2003. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. Journal of empirical finance, 10(5), 603-621.

Ledoit, Olivier and Michael Wolf. 2004. A well-conditioned estimator for large-dimensional covariance matrices. Journal of multivariate analysis, 88(2), 365-411.

Martellini, Lionel, and Volker Ziemann. 2010. Improved estimates of higher-order comoments and implications for portfolio selection. Review of Financial Studies, 23(4), 1467-1502.

Simaan, Yusif. 1993. Portfolio selection and asset pricing: three-parameter framework. Management Science, 39(5), 68-577.

See Also

CoMoments
StructuredMoments
EWMAMoments
MCA
NCE

Examples

```
data(edhec)

# construct an underlying factor (market-factor, observed factor, PCA, ...)
f <- rowSums(edhec)

# multi-target shrinkage with targets 1, 3 and 4
# as.mat = F' would speed up calculations in higher dimensions
targets <- c(1, 3, 4)
sigma <- M2.shrink(edhec, targets, f)$M2sh
m3 <- M3.shrink(edhec, targets, f)$M3sh
m4 <- M4.shrink(edhec, targets, f)$M4sh
```

```
# compute equal-weighted portfolio modified ES
mu <- colMeans(edhec)
p <- length(mu)
ES(p = 0.95, portfolio_method = "component", weights = rep(1 / p, p), mu = mu,
    sigma = sigma, m3 = m3, m4 = m4)

# compare to sample method
sigma <- cov(edhec)
m3 <- M3.MM(edhec)
m4 <- M4.MM(edhec)
ES(p = 0.95, portfolio_method = "component", weights = rep(1 / p, p), mu = mu,
    sigma = sigma, m3 = m3, m4 = m4)
```

---

skewness *Skewness*

Description

compute skewness of a univariate distribution.

Usage

```
skewness(x, na.rm = FALSE, method = c("moment", "fisher", "sample"), ...)
```

Arguments

x	a numeric vector or object.
na.rm	a logical. Should missing values be removed?
method	a character string which specifies the method of computation. These are either "moment" or "fisher" The "moment" method is based on the definitions of skewnessfor distributions; these forms should be used when resampling (bootstrap or jackknife). The "fisher" method correspond to the usual "unbiased" definition of sample variance, although in the case of skewness exact unbiasedness is not possible. The "sample" method gives the sample skewness of the distribution.
...	arguments to be passed.

Details

This function was ported from the RMetrics package fUtilities to eliminate a dependency on fUtiltiies being loaded every time. The function is identical except for the addition of checkData and column support.

$$Skewness(moment) = \frac{1}{n} * \sum_{i=1}^{n} (\frac{r_i - \overline{r}}{\sigma_P})^3$$

$$Skewness(sample) = \frac{n}{(n-1)*(n-2)} * \sum_{i=1}^{n} (\frac{r_i - \overline{r}}{\sigma_{S_P}})^3$$

$$Skewness(fisher) = \frac{\sqrt{n*(n-1)}}{n-2} * \frac{\sum_{i=1}^{n}(x_i - \overline{x})^3/n}{(\sum_{i=1}^{n}(x_i - \overline{x})^2/n)^{3/2}}$$

where n is the number of return, $\overline{r}$ is the mean of the return distribution, σ_P is its standard deviation and σ_{S_P} is its sample standard deviation

Author(s)

Diethelm Wuertz, Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.83-84

See Also

kurtosis

Examples

```
## mean -
## var -
# Mean, Variance:
r <- rnorm(100)
mean(r)
var(r)

## skewness -
skewness(r)
data(managers)
skewness(managers)
```

`SkewnessKurtosisRatio` *Skewness-Kurtosis ratio of the return distribution*

Description

Skewness-Kurtosis ratio is the division of Skewness by Kurtosis.

Usage

```
SkewnessKurtosisRatio(R, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters

Details

It is used in conjunction with the Sharpe ratio to rank portfolios. The higher the rate the better.

$$SkewnessKurtosisRatio(R, MAR) = \frac{S}{K}$$

where S is the skewness and K is the Kurtosis

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.100

Examples

```
data(portfolio_bacon)
print(SkewnessKurtosisRatio(portfolio_bacon[,1])) #expected -0.034

data(managers)
print(SkewnessKurtosisRatio(managers['1996']))
print(SkewnessKurtosisRatio(managers['1996',1]))
```

SmoothingIndex *calculate Normalized Getmansky Smoothing Index*

Description

Proposed by Getmansky et al to provide a normalized measure of "liquidity risk."

Usage

```
SmoothingIndex(R, neg.thetas = FALSE, MAorder = 2, verbose = FALSE, ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
neg.thetas	if FALSE, function removes negative coefficients (thetas) when calculating the index
MAorder	specify the number of periods used to calculate the moving average, defaults to 2
verbose	if TRUE, return a list containing the Thetas in addition to the smoothing index/
...	any other passthru parameters

Details

To measure the effects of smoothing, Getmansky, Lo, et al (2004) define a "smoothing profile" as a vector of coefficients for an MLE fit on returns using a two-period moving-average process.

The moving-average process of order $k = 2$ (specified using `MAorder`) gives $R_t = \theta_0 R_t + \theta_1 R_{t-1} + \theta_2 R_{t-2}$, under the constraint that the sum of the coefficients is equal to 1. In , the `arima` function allows us to create an MA(2) model using an "ARIMA(p,d,q)" model, where p is the number of autoregressive terms (AR), d is the degree of differencing, and q is the number of lagged forecast errors (MA) in the prediction equation. The `order` parameter allows us to specify the three components (p, d, q) as an argument, e.g., `order = c(0, 0, 2)`. The `method` specifies how to fit the model, in this case using maximum likelihood estimation (MLE) in a fashion similar to the estimation of standard moving-average time series models, using:

```
arima(ra, order=c(0,0,2), method="ML", transform.pars=TRUE,include.mean=FALSE)
```

`include.mean`: Getmansky, et al. (2004) p 555 "By applying the above procedure to observed de-meaned returns...". To comply with this, we explicitly subtract the mean from the returns prior to the `arima` fit and set `include.mean=FALSE` to avoid intercept absorption into the MA terms.

`transform.pars`: ibid, "we impose the additional restriction that the estimated MA(k) process be invertible," so we set the parameter to 'TRUE'.

The coefficients, θ_j, are then normalized to sum to interpreted as a "weighted average of the fund's true returns over the most recent $k + 1$ periods, including the current period."

If these weights are disproportionately centered on a small number of lags, relatively little serial correlation will be induced. However, if the weights are evenly distributed among many lags, this would show higher serial correlation.

The paper notes that because $\theta_j \in [0, 1]$, ξ is also confined to the unit interval, and is minimized when all the θ_j's are identical. That implies a value of $1/(k + 1)$ for ξ, and a maximum value of $\xi = 1$ when one coefficient is 1 and the rest are 0. In the context of smoothed returns, a lower value of ξ implies more smoothing, and the upper bound of 1 implies no smoothing.

The "smoothing index", represented as ξ, is calculated the same way the Herfindahl index. The Herfindal measure is well known in the industrial organization literature as a measure of the concentration of firms in a given industry where y_j represents the market share of firm j.

This method (as well as the implementation described in the paper), does not enforce $\theta_j \in [0, 1]$, so ξ is not limited to that range either. All we can say is that lower values are "less liquid" and higher values are "more liquid" or mis-specified. In this function, setting the parameter neg.thetas = FALSE does enforce the limitation, eliminating negative autocorrelation coefficients from the calculation (the papers below do not make an economic case for eliminating negative autocorrelation, however).

Interpretation of the resulting value is difficult. All we can say is that lower values appear to have autocorrelation structure like we might expect of "less liquid" instruments. Higher values appear "more liquid" or are poorly fit or mis-specified.

Acknowledgments

Thanks to Dr. Stefan Albrecht, CFA, for invaluable input.

Author(s)

Peter Carl

References

Chan, Nicholas, Mila Getmansky, Shane M. Haas, and Andrew W. Lo. 2005. Systemic Risk and Hedge Funds. NBER Working Paper Series (11200). Getmansky, Mila, Andrew W. Lo, and Igor Makarov. 2004. An Econometric Model of Serial Correlation and Illiquidity in Hedge Fund Returns. Journal of Financial Economics (74): 529-609.

Examples

```
data(managers)
data(edhec)
SmoothingIndex(managers[, 1, drop = FALSE])
SmoothingIndex(managers[, 1:8])
SmoothingIndex(edhec)
```

---

sortDrawdowns *order list of drawdowns from worst to best*

Description

sortDrawdowns(findDrawdowns(R)) Gives the drawdowns in order of worst to best

Usage

```
sortDrawdowns(runs)
```

Arguments

runs	pass in runs array from findDrawdowns to be sorted

Details

Returns a sorted list:

- return depth of drawdown
- from starting period
- to ending period
- length length in periods

Author(s)

Peter Carl
modified with permission from prototype function by Sankalp Upadhyay

References

Bacon, C. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 3rd Edition. 2023. p. 194-196

See Also

DownsideDeviation
maxDrawdown
findDrawdowns
sortDrawdowns
chart.Drawdown
table.Drawdowns
table.DownsideRisk

Examples

```
data(edhec)
findDrawdowns(edhec[, "Funds of Funds", drop = FALSE])
sortDrawdowns(findDrawdowns(edhec[, "Funds of Funds", drop = FALSE]))
```

SortinoRatio *calculate Sortino Ratio of performance over downside risk*

Description

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.

Usage

```
SortinoRatio(R, MAR = 0, ..., weights = NULL, SE = FALSE, SE.control = NULL)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
MAR	Minimum Acceptable Return, in the same periodicity as your returns
...	any other passthru parameters
weights	portfolio weighting vector, default NULL
SE	TRUE/FALSE whether to ouput the standard errors of the estimates of the risk measures, default FALSE.
SE.control	Control parameters for the computation of standard errors. Should be done using the RPESE.control function.

Details

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.

$$SortinoRatio = \frac{(\overline{R_a - MAR})}{\delta_{MAR}}$$

where δ_{MAR} is the DownsideDeviation.

Author(s)

Brian G. Peterson

References

Sortino, F. and Price, L. Performance Measurement in a Downside Risk Framework. *Journal of Investing*. Fall 1994, 59-65.

See Also

SharpeRatio
DownsideDeviation
SemiVariance
SemiDeviation
InformationRatio

Examples

```
data(managers)
round(SortinoRatio(managers[, 1]), 4)
round(SortinoRatio(managers[, 1:8]), 4)
```

SpecificRisk *Specific risk of the return distribution*

Description

Specific risk is the standard deviation of the error term in the regression equation.

Usage

```
SpecificRisk(Ra, Rb, Rf = 0, ...)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
Rf	risk free rate, in same period as your returns
...	any other passthru parameters

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.75

Examples

```
data(portfolio_bacon)
print(SpecificRisk(portfolio_bacon[,1], portfolio_bacon[,2])) #expected 0.0329

data(managers)
print(SpecificRisk(managers['1996',1], managers['1996',8]))
print(SpecificRisk(managers['1996',1:5], managers['1996',8]))
```

StdDev *calculates Standard Deviation for univariate and multivariate series, also calculates component contribution to standard deviation of a portfolio*

Description

calculates Standard Deviation for univariate and multivariate series, also calculates component contribution to standard deviation of a portfolio

Usage

```
StdDev(
  R,
  ...,
  clean = c("none", "boudt", "geltner", "locScaleRob"),
  portfolio_method = c("single", "component"),
  weights = NULL,
  mu = NULL,
  sigma = NULL,
  use = "everything",
  method = c("pearson", "kendall", "spearman"),
  sample_method = c("unbiased", "ML"),
  SE = FALSE,
  SE.control = NULL
)
```

Arguments

R	a vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters
clean	method for data cleaning through Return.clean. Current options are "none", "boudt", "geltner", or "locScaleRob".
portfolio_method	one of "single","component" defining whether to do univariate/multivariate or component calc, see Details.
weights	portfolio weighting vector, default NULL, see Details
mu	If univariate, mu is the mean of the series. Otherwise mu is the vector of means of the return series , default NULL, , see Details
sigma	If univariate, sigma is the variance of the series. Otherwise sigma is the covariance matrix of the return series , default NULL, see Details
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs".

method	a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman", can be abbreviated.
sample_method	character string, one of "unbiased" (default) or "ML" (maximum likelihood). "unbiased" uses $n - 1$ in the denominator, while "ML" uses n. This matches the behavior of cov.wt.
SE	TRUE/FALSE whether to ouput the standard errors of the estimates of the risk measures, default FALSE.
SE.control	Control parameters for the computation of standard errors. Should be done using the RPESE.control function.

Details

TODO add more details

This wrapper function provides fast matrix calculations for univariate, multivariate, and component contributions to Standard Deviation.

It is likely that the only one that requires much description is the component decomposition. This provides a weighted decomposition of the contribution each portfolio element makes to the univariate standard deviation of the whole portfolio.

Formally, this is the partial derivative of each univariate standard deviation with respect to the weights.

As with VaR, this contribution is presented in two forms, both a scalar form that adds up to the univariate standard deviation of the portfolio, and a percentage contribution, which adds up to 100 as with any contribution calculation, contribution can be negative. This indicates that the asset in question is a diversified to the overall standard deviation of the portfolio, and increasing its weight in relation to the rest of the portfolio would decrease the overall portfolio standard deviation.

Author(s)

Brian G. Peterson and Kris Boudt

See Also

Return.clean sd

Examples

```
data(edhec)

# first do normal StdDev calc
StdDev(edhec)
# or the equivalent
StdDev(edhec, portfolio_method = "single")

# now with outliers squished
StdDev(edhec, clean = "boudt")

# add Component StdDev for the equal weighted portfolio
```

```
StdDev(edhec, clean = "boudt", portfolio_method = "component")

# end CRAN check
```

`StdDev.annualized` *calculate a multiperiod or annualized Standard Deviation*

Description

Standard Deviation of a set of observations R_a is given by:

Usage

```
StdDev.annualized(x, scale = NA, ..., sample_method = c("unbiased", "ML"))
```

Arguments

`x`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`scale`	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
`...`	any other passthru parameters
`sample_method`	character string, one of `"unbiased"` (default) or `"ML"` (maximum likelihood). `"unbiased"` uses $n - 1$ in the denominator, while `"ML"` uses n. This matches the behavior of `cov.wt`.

Details

$$\sigma = variance(R_a), std = \sqrt{\sigma}$$

It should follow that the variance is not a linear function of the number of observations. To determine possible variance over multiple periods, it wouldn't make sense to multiply the single-period variance by the total number of periods: this could quickly lead to an absurd result where total variance (or risk) was greater than 100 variance needs to demonstrate a decreasing period-to-period increase as the number of periods increases. Put another way, the increase in incremental variance per additional period needs to decrease with some relationship to the number of periods. The standard accepted practice for doing this is to apply the inverse square law. To normalize standard deviation across multiple periods, we multiply by the square root of the number of periods we wish to calculate over. To annualize standard deviation, we multiply by the square root of the number of periods per year.

$$\sqrt{\sigma} \cdot \sqrt{periods}$$

Note that any multiperiod or annualized number should be viewed with suspicion if the number of observations is small.

Author(s)

Brian G. Peterson

References

Bacon, C. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004. p. 27

See Also

sd
https://en.wikipedia.org/wiki/Inverse-square_law

Examples

```
data(edhec)
sd.annualized(edhec)
sd.annualized(edhec[, 6, drop = FALSE])
# now for three periods:
sd.multiperiod(edhec[, 6, drop = FALSE], scale = 3)
```

StructuredMoments *Functions for calculating structured comoments of financial time series*

Description

calculates covariance, coskewness and cokurtosis matrices as structured estimators

Usage

```
M2.struct(R, struct = c("Indep", "IndepId", "observedfactor", "CC"), f = NULL)

M3.struct(
  R,
  struct = c("Indep", "IndepId", "observedfactor", "CC", "latent1factor", "CS"),
  f = NULL,
  unbiasedMarg = FALSE,
  as.mat = TRUE
)

M4.struct(
  R,
  struct = c("Indep", "IndepId", "observedfactor", "CC"),
  f = NULL,
  as.mat = TRUE
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
struct	string containing the preferred method. See Details.
f	vector or matrix with observations of the factor, to be used with 'observedfactor'. See Details.
unbiasedMarg	TRUE/FALSE whether to use a correction to have an unbiased estimator for the marginal skewness values, in case of 'Indep' or 'IndepId', default FALSE
as.mat	TRUE/FALSE whether to return the full moment matrix or only the vector with the unique elements (the latter is advised for speed), default TRUE

Details

The coskewness and cokurtosis matrices are defined as the matrices of dimension p x p^2 and p x p^3 containing the third and fourth order central moments. They are useful for measuring nonlinear dependence between different assets of the portfolio and computing modified VaR and modified ES of a portfolio.

Structured estimation is based on the assumption that the underlying data-generating process is known, or at least resembles the assumption. The first four structured estimators correspond to the models 'independent marginals', 'independent and identical marginals', 'observed multi-factor model' and 'constant correlation'. Coskewness estimation includes an additional model based on the latent 1-factor model proposed in Simaan (1993).

The constant correlation and 1-factor coskewness and cokurtosis matrices can be found in Martellini and Ziemann (2010). If f is a matrix containing multiple factors, then the multi-factor model of Boudt, Lu and Peeters (2915) is used. For information about the other structured matrices, we refer to Boudt, Cornilly and Verdonck (2017)

Author(s)

Dries Cornilly

References

Boudt, Kris, Wanbo Lu and Benedict Peeters. 2015. Higher order comoments of multifactor models and asset allocation. Finance Research Letters, 13, 225-233.

Boudt, Kris, Brian G. Peterson, and Christophe Croux. 2008. Estimation and Decomposition of Downside Risk for Portfolios with Non-Normal Returns. Journal of Risk. Winter.

Boudt, Kris, Dries Cornilly and Tim Verdonck. 2020 A Coskewness Shrinkage Approach for Estimating the Skewness of Linear Combinations of Random Variables. Journal of Financial Econometrics, 18(1), 1-23.

Ledoit, Olivier and Michael Wolf. 2003. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. Journal of empirical finance, 10(5), 603-621.

Martellini, Lionel, and Volker Ziemann. 2010. Improved estimates of higher-order comoments and implications for portfolio selection. Review of Financial Studies, 23(4), 1467-1502.

Simaan, Yusif. 1993. Portfolio selection and asset pricing: three-parameter framework. Management Science, 39(5), 68-577.

See Also

CoMoments
ShrinkageMoments
EWMAMoments
MCA
NCE

Examples

```
data(edhec)

# structured estimation with constant correlation model
# 'as.mat = F' would speed up calculations in higher dimensions
sigma <- M2.struct(edhec, "CC")
m3 <- M3.struct(edhec, "CC")
m4 <- M4.struct(edhec, "CC")

# compute equal-weighted portfolio modified ES
mu <- colMeans(edhec)
p <- length(mu)
ES(p = 0.95, portfolio_method = "component", weights = rep(1 / p, p), mu = mu,
    sigma = sigma, m3 = m3, m4 = m4)

# compare to sample method
sigma <- cov(edhec)
m3 <- M3.MM(edhec)
m4 <- M4.MM(edhec)
ES(p = 0.95, portfolio_method = "component", weights = rep(1 / p, p), mu = mu,
    sigma = sigma, m3 = m3, m4 = m4)
```

SystematicRisk *Systematic risk of the return distribution*

Description

Systematic risk as defined by Bacon(2008) is the product of beta by market risk. Be careful ! It's not the same definition as the one given by Michael Jensen. Market risk is the standard deviation of the benchmark. The systematic risk is annualized

Usage

```
SystematicRisk(Ra, Rb, Rf = 0, scale = NA, ...)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset

Rf	risk free rate, in same period as your returns
scale	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
...	any other passthru parameters

Details

$$\sigma_s = \beta * \sigma_m$$

where σ_s is the systematic risk, β is the regression beta, and σ_m is the market risk

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.75

Examples

```
data(portfolio_bacon)
print(SystematicRisk(portfolio_bacon[,1], portfolio_bacon[,2])) #expected 0.013

data(managers)
print(SystematicRisk(managers['2002',1], managers['2002',8]))
print(SystematicRisk(managers['2002',1:5], managers['2002',8]))
```

---

table.AnnualizedReturns

Annualized Returns Summary: Statistics and Stylized Facts

Description

Table of Annualized Return, Annualized Std Dev, and Annualized Sharpe

Usage

```
table.AnnualizedReturns(R, scale = NA, Rf = 0, geometric = TRUE, digits = 4)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
scale	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
Rf	risk free rate, in same period as your returns
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
digits	number of digits to round results to

Author(s)

Peter Carl

See Also

Return.annualized
StdDev.annualized
SharpeRatio.annualized

Examples

```
 # don't test on CRAN, since it requires Suggested packages
data(managers)
table.AnnualizedReturns(managers[,1:8])

require("Hmisc")
result = t(table.AnnualizedReturns(managers[,1:8], Rf=.04/12))

textplot(format.df(result, na.blank=TRUE, numeric.dollar=FALSE,
         cdec=c(3,3,1)), rmar = 0.8, cmar = 2,  max.cex=.9,
         halign = "center", valign = "top", row.valign="center",
         wrap.rownames=20, wrap.colnames=10, col.rownames=c("red",
         rep("darkgray",5), rep("orange",2)), mar = c(0,0,3,0)+0.1)

title(main="Annualized Performance")
```

---

table.Arbitrary *wrapper function for combining arbitrary function list into a table*

Description

This function creates a table of statistics from vectors of functions and labels passed in. The resulting table is formatted such that metrics are calculated separately for each column of returns in the data object.

Usage

```
table.Arbitrary(
  R,
  metrics = c("mean", "sd"),
  metricsNames = c("Average Return", "Standard Deviation"),
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
metrics	lisdt of functions to apply
metricsNames	column names for each function
...	any other passthru parameters

Details

Assumes an input of period returns. Scale arguements can be used to specify the number of observations during a year (e.g., 12 = monthly returns).

The idea here is to be able to pass in sets of metrics and values, like:

metrics = c(DownsideDeviation(x,MAR=mean(x)), sd(subset(x,x>0)), sd(subset(x,x<0)), DownsideDeviation(x,MAR=MAR), DownsideDeviation(x,MAR=Rf=0), DownsideDeviation(x,MAR=0),maxDrawdown(x))

metricsNames = c("Semi Deviation", "Gain Deviation", "Loss Deviation", paste("Downside Deviation (MAR=",MAR*scale*100," paste("Downside Deviation (rf=",rf*scale*100," Deviation (0

Here's how it's working right now: > table.Arbitrary(monthlyReturns.ts,metrics=c("VaR","mean"), metricsNames=c("modVaR","mean"),p=.95)

```
 Actual S&P500TR modVaR
0.04186461 0.06261451 mean 0.00945000 0.01013684
```

Passing in two different sets of attributes to the same function doesn't quite work currently. The issue is apparent in: > table.Arbitrary(edhec,metrics=c("VaR", "VaR"), metricsNames=c("Modified VaR","Traditional VaR"), modified=c(TRUE,FALSE))

```
Convertible.Arbitrage CTA.Global Distressed.Securities Modified VaR
0.04081599 0.0456767 0.1074683 Traditional VaR 0.04081599 0.0456767
0.1074683 Emerging.Markets Equity.Market.Neutral Event.Driven Modified VaR
0.1858624 0.01680917 0.1162714 Traditional VaR 0.1858624 0.01680917
0.1162714 Fixed.Income.Arbitrage Global.Macro Long.Short.Equity Modified VaR
0.2380379 0.03700478 0.04661244 Traditional VaR 0.2380379 0.03700478
0.04661244 Merger.Arbitrage Relative.Value Short.Selling Funds.of.Funds
Modified VaR 0.07510643 0.04123920 0.1071894 0.04525633 Traditional VaR
0.07510643 0.04123920 0.1071894 0.04525633
```

In the case of this example, you would simply call VaR as the second function, like so: > table.Arbitrary(edhec,metrics=c("VaR", "VaR"),metricsNames=c("Modified VaR","Traditional VaR"))

```
Convertible.Arbitrage CTA.Global Distressed.Securities Modified VaR
0.04081599 0.04567670 0.10746831 Traditional VaR 0.02635371 0.04913361
0.03517855 Emerging.Markets Equity.Market.Neutral Event.Driven Modified VaR
0.18586240 0.01680917 0.11627142 Traditional VaR 0.07057278 0.01746554
0.03563019 Fixed.Income.Arbitrage Global.Macro Long.Short.Equity Modified
VaR 0.23803787 0.03700478 0.04661244 Traditional VaR 0.02231236 0.03692096
0.04318713 Merger.Arbitrage Relative.Value Short.Selling Funds.of.Funds
Modified VaR 0.07510643 0.04123920 0.1071894 0.04525633 Traditional VaR
0.02510709 0.02354012 0.0994635 0.03502065
```

but we don't know of a way to compare the same function side by side with different parameters for each. Suggestions Welcome.

Author(s)

Peter Carl

Examples

```
data(edhec)
table.Arbitrary(edhec,metrics=c("VaR", "ES"),
                metricsNames=c("Modified VaR","Modified Expected Shortfall"))
```

`table.Autocorrelation` *table for calculating the first six autocorrelation coefficients and significance*

Description

Produces data table of autocorrelation coefficients ρ and corresponding Q(6)-statistic for each column in R.

Usage

```
table.Autocorrelation(R, digits = 4, max.lag = 6)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`digits`	number of digits to round results to for display
`max.lag`	the maximum autocorrelation lag to include in the table

Note

To test returns for autocorrelation, Lo (2001) suggests the use of the Ljung-Box test, a significance test for the auto-correlation coefficients. Ljung and Box (1978) provide a refinement of the Q-statistic proposed by Box and Pierce (1970) that offers a better fit for the χ^2 test for small sample sizes. `Box.test` provides both.

Author(s)

Peter Carl

References

Lo, Andrew W. 2001. Risk Management for Hedge Funds: Introduction and Overview. SSRN eLibrary.

See Also

Box.test, acf

Examples

```
 # CRAN does not allow examples to load suggested packages in one of its tests

data(managers)
t(table.Autocorrelation(managers))

result = t(table.Autocorrelation(managers[,1:8]))

textplot(result, rmar = 0.8, cmar = 2,  max.cex=.9, halign = "center",
         valign = "top", row.valign="center", wrap.rownames=15,
         wrap.colnames=10, mar = c(0,0,3,0)+0.1)

title(main="Autocorrelation")
```

---

table.CalendarReturns *Monthly and Calendar year Return table*

Description

Returns a table of returns formatted with years in rows, months in columns, and a total column in the last column. For additional columns in R, annual returns will be appended as columns.

Usage

```
table.CalendarReturns(R, digits = 1, as.perc = TRUE, geometric = TRUE)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of monthly or daily asset returns
digits	number of digits to round results to for presentation
as.perc	TRUE/FALSE if TRUE, multiply simple returns by 100 to get %
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE

Note

This function assumes monthly returns or daily returns and does not have handling for other periodicities

This function defaults to the first column as the monthly returns to be formatted.

Author(s)

Peter Carl

Examples

```
data(managers)
t(table.CalendarReturns(managers[,c(1,7,8)]))

 # don't test on CRAN, since it requires Suggested packages

# prettify with format.df in hmisc package
require("Hmisc")
result = t(table.CalendarReturns(managers[,c(1,8)]))

textplot(format.df(result, na.blank=TRUE, numeric.dollar=FALSE,
         cdec=rep(1,dim(result)[2])), rmar = 0.8, cmar = 1,
         max.cex=.9, halign = "center", valign = "top",
         row.valign="center", wrap.rownames=20, wrap.colnames=10,
         col.rownames=c( rep("darkgray",12), "black", "blue"),
         mar = c(0,0,3,0)+0.1)

title(main="Calendar Returns")
```

---

table.CaptureRatios *Calculate and display a table of capture ratio and related statistics*

Description

Creates a table of capture ratios and similar metrics for a set of returns against a benchmark.

Usage

```
table.CaptureRatios(Ra, Rb, digits = 4, geometric = TRUE)

table.UpDownRatios(Ra, Rb, digits = 4, geometric = TRUE)
```

Arguments

Ra	a vector of returns to test, e.g., the asset to be examined
Rb	a matrix, data.frame, or timeSeries of benchmark(s) to test the asset against.
digits	number of digits to round results to for presentation

`geometric` utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE

Details

This table will show statistics pertaining to an asset against a set of benchmarks, or statistics for a set of assets against a benchmark. `table.CaptureRatios` shows only the capture ratio; `table.UpDownRatios` shows three: the capture ratio, the number ratio, and the percentage ratio.

Author(s)

Peter Carl

See Also

`UpDownRatios`, `chart.CaptureRatios`

Examples

```
data(managers)
table.CaptureRatios(managers[, 1:6], managers[, 7, drop = FALSE])
table.UpDownRatios(managers[, 1:6], managers[, 7, drop = FALSE])

result <- t(table.UpDownRatios(managers[, 1:6], managers[, 7, drop = FALSE]))
colnames(result) <- colnames(managers[, 1:6])
textplot(result,
  rmar = 0.8, cmar = 1.5, max.cex = .9,
  halign = "center", valign = "top", row.valign = "center",
  wrap.rownames = 15, wrap.colnames = 10, mar = c(0, 0, 3, 0) + 0.1
)

title(main = "Capture Ratios for EDHEC LS EQ")
```

---

table.Correlation *calculate correlalations of multicolumn data*

Description

This is a wrapper for calculating correlation and significance against each column of the data provided.

Usage

```
table.Correlation(Ra, Rb, ...)
```

Arguments

Ra	a vector of returns to test, e.g., the asset to be examined
Rb	a matrix, data.frame, or timeSeries of benchmark(s) to test the asset against.
...	any other passthru parameters to cor.test

Author(s)

Peter Carl

See Also

cor.test

Examples

```
# First we load the data
data(managers)
table.Correlation(managers[,1:6],managers[,7:8])

result=table.Correlation(managers[,1:6],managers[,8])
rownames(result)=colnames(managers[,1:6])

 # don't test on CRAN, since it requires Suggested packages

require("Hmisc")
textplot(format.df(result, na.blank=TRUE, numeric.dollar=FALSE,
         cdec=rep(3,dim(result)[2])), rmar = 0.8, cmar = 1.5,
         max.cex=.9, halign = "center", valign = "top", row.valign="center"
         , wrap.rownames=20, wrap.colnames=10, mar = c(0,0,3,0)+0.1)
title(main="Correlations to SP500 TR")

ctable = table.Correlation(managers[,1:6],managers[,8,drop=FALSE], conf.level=.99)
dotchart(ctable[,1],labels=rownames(ctable),xlim=c(-1,1))
```

---

table.Distributions *Distributions Summary: Statistics and Stylized Facts*

Description

Table of standard deviation, Skewness, Sample standard deviation, Kurtosis, Excess kurtosis, Sample Skweness and Sample excess kurtosis

Usage

```
table.Distributions(R, scale = NA, digits = 4)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
scale	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
digits	number of digits to round results to

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.87

See Also

StdDev.annualized
skewness
kurtosis

Examples

```
data(managers)
table.Distributions(managers[, 1:8])


# don't test on CRAN, since it requires Suggested packages

require("Hmisc")
result <- t(table.Distributions(managers[, 1:8]))

textplot(format.df(result, na.blank = TRUE, numeric.dollar = FALSE, cdec = c(3, 3, 1)),
  rmar = 0.8, cmar = 2, max.cex = .9, halign = "center", valign = "top",
  row.valign = "center", wrap.rownames = 20, wrap.colnames = 10,
  col.rownames = c("red", rep("darkgray", 5), rep("orange", 2)), mar = c(0, 0, 3, 0) + 0.1
)
title(main = "Portfolio Distributions statistics")
```

---

table.DownsideRisk *Downside Risk Summary: Statistics and Stylized Facts*

Description

Creates a table of estimates of downside risk measures for comparison across multiple instruments or funds.

Usage

```
table.DownsideRisk(
  R,
  ci = 0.95,
  scale = NA,
  Rf = 0,
  MAR = 0.1/12,
  p = 0.95,
  digits = 4
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
ci	confidence interval, defaults to 95%
scale	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
Rf	risk free rate, in same period as your returns
MAR	Minimum Acceptable Return, in the same periodicity as your returns
p	confidence level for calculation, default p=.99
digits	number of digits to round results to

Author(s)

Peter Carl

See Also

DownsideDeviation
maxDrawdown
VaR
ES

Examples

```
data(edhec)
table.DownsideRisk(edhec, Rf=.04/12, MAR =.05/12, p=.95)

result=t(table.DownsideRisk(edhec, Rf=.04/12, MAR =.05/12, p=.95))

 # don't test on CRAN, since it requires Suggested packages

require("Hmisc")
textplot(format.df(result, na.blank=TRUE, numeric.dollar=FALSE,
         cdec=rep(3,dim(result)[2])), rmar = 0.8, cmar = 1.5,
         max.cex=.9, halign = "center", valign = "top", row.valign="center",
         wrap.rownames=15, wrap.colnames=10, mar = c(0,0,3,0)+0.1)
```

```
title(main="Downside Risk Statistics")
```

`table.DownsideRiskRatio`

Downside Summary: Statistics and ratios

Description

Table of downside risk, Annualised downside risk, Downside potential, Omega, Sortino ratio, Upside potential, Upside potential ratio and Omega-Sharpe ratio

Usage

```
table.DownsideRiskRatio(R, MAR = 0, scale = NA, digits = 4)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
MAR	Minimum Acceptable Return, in the same periodicity as your returns
scale	number of periods in a year (daily scale = 252, monthly scale =
digits	number of digits to round results to

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.98

See Also

CalmarRatio
BurkeRatio
PainIndex
UlcerIndex
PainRatio
MartinRatio

Examples

```
data(managers)
table.DownsideRiskRatio(managers[, 1:8])


# don't test on CRAN, since it requires Suggested packages

require("Hmisc")
result <- t(table.DownsideRiskRatio(managers[, 1:8]))

textplot(format.df(result, na.blank = TRUE, numeric.dollar = FALSE, cdec = c(3, 3, 1)),
  rmar = 0.8, cmar = 2, max.cex = .9, halign = "center", valign = "top",
  row.valign = "center", wrap.rownames = 20, wrap.colnames = 10,
  col.rownames = c("red", rep("darkgray", 5), rep("orange", 2)), mar = c(0, 0, 3, 0) + 0.1
)
title(main = "Downside risk statistics")
```

`table.Drawdowns` *Worst Drawdowns Summary: Statistics and Stylized Facts*

Description

Creates table showing statistics for the worst drawdowns.

Usage

```
table.Drawdowns(R, top = 5, digits = 4, geometric = TRUE, ...)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`top`	the number of drawdowns to include
`digits`	number of digits to round results to
`geometric`	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
`...`	any other passthru parameters

Details

Returns an data frame with columns:

- From starting period, high water mark
- Trough period of low point
- To ending period, when initial high water mark is recovered

- Depth drawdown to trough (typically as percentage returns)
- Length length in periods
- toTrough number of periods to trough
- Recovery number of periods to recover

Author(s)

Peter Carl

References

Bacon, C. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004. p. 88

See Also

DownsideDeviation
maxDrawdown
findDrawdowns
sortDrawdowns
chart.Drawdown
table.DownsideRisk

Examples

```
data(edhec)
table.Drawdowns(edhec[,1,drop=FALSE])
table.Drawdowns(edhec[,12,drop=FALSE])
data(managers)
table.Drawdowns(managers[,8,drop=FALSE])

result=table.Drawdowns(managers[,1,drop=FALSE])

# This was really nice before Hmisc messed up 'format' from R-base
#require("Hmisc")
#textplot(Hmisc::format.df(result, na.blank=TRUE, numeric.dollar=FALSE,
#           cdec=c(rep(3,4), rep(0,3))), rmar = 0.8, cmar = 1.5,
#           max.cex=.9, halign = "center", valign = "top", row.valign="center",
#           wrap.rownames=5, wrap.colnames=10, mar = c(0,0,3,0)+0.1)
# title(main="Largest Drawdowns for HAM1")
```

---

table.DrawdownsRatio *Drawdowns Summary: Statistics and ratios*

Description

Table of Calmar ratio, Sterling ratio, Burke ratio, Pain index, Ulcer index, Pain ratio and Martin ratio

Usage

```
table.DrawdownsRatio(R, Rf = 0, scale = NA, digits = 4)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rf	risk free rate, in same period as your returns
scale	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
digits	number of digits to round results to

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.93

See Also

CalmarRatio
BurkeRatio
PainIndex
UlcerIndex
PainRatio
MartinRatio

Examples

```
data(managers)
table.DrawdownsRatio(managers[,1:8])

 # don't test on CRAN, since it requires Suggested packages

require("Hmisc")
result = t(table.DrawdownsRatio(managers[,1:8], Rf=.04/12))

textplot(format.df(result, na.blank=TRUE, numeric.dollar=FALSE, cdec=c(3,3,1)),
rmar = 0.8, cmar = 2,  max.cex=.9, halign = "center", valign = "top",
row.valign="center", wrap.rownames=20, wrap.colnames=10,
col.rownames=c("red", rep("darkgray",5), rep("orange",2)), mar = c(0,0,3,0)+0.1)
title(main="Drawdowns ratio statistics")
```

`table.HigherMoments` *Higher Moments Summary: Statistics and Stylized Facts*

Description

Summary of the higher moements and Co-Moments of the return distribution. Used to determine diversification potential. Also called "systematic" moments by several papers.

Usage

```
table.HigherMoments(Ra, Rb, scale = NA, Rf = 0, digits = 4, method = "moment")
```

Arguments

`Ra`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rb`	return vector of the benchmark asset
`scale`	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
`Rf`	risk free rate, in same period as your returns
`digits`	number of digits to round results to
`method`	method to use when computing `kurtosis` one of: `excess`, `moment`, `fisher`

Author(s)

Peter Carl

References

Martellini L., Vaissie M., Ziemann V. Investing in Hedge Funds: Adding Value through Active Style Allocation Decisions. October 2005. Edhec Risk and Asset Management Research Centre.

See Also

`CoSkewness`
`CoKurtosis`
`BetaCoVariance`
`BetaCoSkewness`
`BetaCoKurtosis`
`skewness`
`kurtosis`

Examples

```
data(managers)
table.HigherMoments(managers[,1:3],managers[,8,drop=FALSE])
result=t(table.HigherMoments(managers[,1:6],managers[,8,drop=FALSE]))
rownames(result)=colnames(managers[,1:6])

 # don't test on CRAN, since it requires Suggested packages

require("Hmisc")
textplot(format.df(result, na.blank=TRUE, numeric.dollar=FALSE,
         cdec=rep(3,dim(result)[2])), rmar = 0.8, cmar = 1.5,
         max.cex=.9, halign = "center", valign = "top", row.valign="center",
         wrap.rownames=5, wrap.colnames=10, mar = c(0,0,3,0)+0.1)
title(main="Higher Co-Moments with SP500 TR")
```

---

table.InformationRatio

Information ratio Summary: Statistics and Stylized Facts

Description

Table of Tracking error, Annualised tracking error and Information ratio

Usage

```
table.InformationRatio(R, Rb, scale = NA, digits = 4)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
scale	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
digits	number of digits to round results to

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.81

See Also

InformationRatio
TrackingError

Examples

```
data(managers)
table.InformationRatio(managers[,1:8], managers[,8])

 # don't test on CRAN, since it requires Suggested packages

require("Hmisc")
result = t(table.InformationRatio(managers[,1:8], managers[,8]))

textplot(format.df(result, na.blank=TRUE, numeric.dollar=FALSE, cdec=c(3,3,1)),
rmar = 0.8, cmar = 2,  max.cex=.9, halign = "center", valign = "top",
row.valign="center", wrap.rownames=20, wrap.colnames=10,
col.rownames=c("red", rep("darkgray",5), rep("orange",2)), mar = c(0,0,3,0)+0.1)
title(main="Portfolio information ratio")
```

---

table.ProbOutPerformance

Outperformance Report of Asset vs Benchmark

Description

Table of Outperformance Reporting vs Benchmark

Usage

```
table.ProbOutPerformance(R, Rb, period_lengths = c(1, 3, 6, 9, 12, 18, 36))
```

Arguments

R	an xts, timeSeries or zoo object of asset returns
Rb	an xts, timeSeries or zoo object of the benchmark returns
period_lengths	a vector of periods the user wants to evaluate this over i.e. c(1,3,6,9,12,18,36)

Details

Returns a table that contains the counts and probabilities of outperformance relative to benchmark for the various period_lengths

Tool for robustness analysis of an asset or strategy, can be used to give the probability an investor investing at any point in time will outperform the benchmark over a given horizon. Calculates Count of trailing periods where a fund outperformed its benchmark and calculates the proportion of those

periods, this is commonly used in marketing as the probability of outperformance on a N period basis.

Returns a table that contains the counts and probabilities of outperformance relative to benchmark for the various period_lengths

Author(s)

Kyle Balkissoon

Examples

```
data(edhec)

table.ProbOutPerformance(edhec[,1],edhec[,2])
title(main='Table of Convertible Arbitrage vs Benchmark')
```

---

table.RollingPeriods *Rolling Periods Summary: Statistics and Stylized Facts*

Description

A table of estimates of rolling period return measures

Usage

```
table.RollingPeriods(
  R,
  periods = subset(c(12, 36, 60), c(12, 36, 60) < length(as.matrix(R[, 1]))),
  FUNCS = c("mean", "sd"),
  funcs.names = c("Average", "Std Dev"),
  digits = 4,
  ...
)

table.TrailingPeriodsRel(
  R,
  Rb,
  periods = subset(c(12, 36, 60), c(12, 36, 60) < length(as.matrix(R[, 1]))),
  FUNCS = c("cor", "CAPM.beta"),
  funcs.names = c("Correlation", "Beta"),
  digits = 4,
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
periods	number of periods to use as rolling window(s), subset of c(3, 6, 9, 12, 18, 24, 36, 48)
FUNCS	list of functions to apply the rolling period to
funcs.names	vector of function names used for labeling table rows
digits	number of digits to round results to
...	any other passthru parameters for functions specified in FUNCS
Rb	an xts, vector, matrix, data frame, timeSeries or zoo object of index, benchmark, portfolio, or secondary asset returns to compare against

Details

table.RollingPeriods and table.TrailingPeriods (which are aliases) calculate statistics over rolling windows for individual assets. They apply the FUNCS directly to each column of R independently.

table.TrailingPeriodsRel calculates statistics over rolling windows *relative* to a benchmark. It applies the FUNCS to each column of R and the corresponding column of Rb simultaneously. This is ideal for relative metrics like CAPM.beta, cor, or tracking error. If you need to calculate metrics requiring a benchmark, use table.TrailingPeriodsRel.

Author(s)

Peter Carl

See Also

rollapply

Examples

```
data(edhec)

# Example 1: Univariate Rolling Statistics
table.TrailingPeriods(edhec[, 10:13], periods = c(12, 24, 36))

# Example 2: Relative Rolling Statistics against a Benchmark
table.TrailingPeriodsRel(edhec[, 10:12],
  Rb = edhec[, 13],
  periods = c(12, 24, 36),
  FUNCS = c("cor", "CAPM.beta"),
  funcs.names = c("Correlation", "Beta")
)

result <- table.TrailingPeriods(edhec[, 10:13], periods = c(12, 24, 36))


# don't test on CRAN, since it requires Suggested packages
```

```
require("Hmisc")
textplot(
  format.df(result,
    na.blank = TRUE, numeric.dollar = FALSE,
    cdec = rep(3, dim(result)[2])
  ),
  rmar = 0.8, cmar = 1.5,
  max.cex = .9, halign = "center", valign = "top", row.valign = "center",
  wrap.rownames = 15, wrap.colnames = 10, mar = c(0, 0, 3, 0) + 0.1
)
title(main = "Trailing Period Statistics")
```

`table.SFM` *Single Factor Asset-Pricing Model Summary: Statistics and Stylized Facts*

Description

Takes a set of returns and relates them to a benchmark return. Provides a set of measures related to an excess return single factor model, or CAPM.

Usage

```
table.SFM(Ra, Rb, scale = NA, Rf = 0, digits = 4)
```

Arguments

`Ra`	a vector of returns to test, e.g., the asset to be examined
`Rb`	a matrix, data.frame, or timeSeries of benchmark(s) to test the asset against.
`scale`	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
`Rf`	risk free rate, in same period as your returns
`digits`	number of digits to round results to

Details

This table will show statistics pertaining to an asset against a set of benchmarks, or statistics for a set of assets against a benchmark.

Author(s)

Peter Carl

See Also

CAPM.alpha
CAPM.beta
TrackingError
ActivePremium
InformationRatio
TreynorRatio

Examples

```
 # CRAN does not allow examples to load suggested packages in one of its tests
data(managers)
table.SFM(managers[,1:3], managers[,8], Rf = managers[,10])

result = table.SFM(managers[,1:3], managers[,8], Rf = managers[,10])
require(Hmisc)
textplot(result, rmar = 0.8, cmar = 1.5,  max.cex=.9,
         halign = "center", valign = "top", row.valign="center",
         wrap.rownames=15, wrap.colnames=10, mar = c(0,0,3,0)+0.1)
title(main="Single Factor Model Related Statistics")
```

---

table.SpecificRisk *Specific risk Summary: Statistics and Stylized Facts*

Description

Table of specific risk, systematic risk and total risk

Usage

```
table.SpecificRisk(Ra, Rb, Rf = 0, digits = 4)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
Rf	risk free rate, in same period as your returns
digits	number of digits to round results to

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.76

See Also

SystematicRisk
SpecificRisk
TotalRisk

Examples

```
data(managers)
table.SpecificRisk(managers[,1:8], managers[,8])

 # don't test on CRAN, since it requires Suggested packages

require("Hmisc")
result = t(table.SpecificRisk(managers[,1:8], managers[,8], Rf=.04/12))

textplot(format.df(result, na.blank=TRUE, numeric.dollar=FALSE, cdec=c(3,3,1)),
rmar = 0.8, cmar = 2,  max.cex=.9, halign = "center", valign = "top",
row.valign="center", wrap.rownames=20, wrap.colnames=10,
col.rownames=c("red", rep("darkgray",5), rep("orange",2)), mar = c(0,0,3,0)+0.1)
title(main="Portfolio specific, systematic and total risk")
```

---

table.Stats *Returns Summary: Statistics and Stylized Facts*

Description

Returns a basic set of statistics that match the period of the data passed in (e.g., monthly returns will get monthly statistics, daily will be daily stats, and so on)

Usage

```
table.Stats(R, ci = 0.95, digits = 4)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
ci	confidence interval, defaults to 95%
digits	number of digits to round results to

Details

This was created as a way to display a set of related statistics together for comparison across a set of instruments or funds. Careful consideration to missing data or unequal time series should be given when intepreting the results.

Author(s)

Peter Carl

Examples

```
data(edhec)
table.Stats(edhec[,1:3])
t(table.Stats(edhec))

result=t(table.Stats(edhec))

 # don't test on CRAN, since it requires Suggested packages

require("Hmisc")
textplot(format.df(result, na.blank=TRUE, numeric.dollar=FALSE, cdec=c(rep(1,2),rep(3,14))),
         rmar = 0.8, cmar = 1.5,  max.cex=.9, halign = "center", valign = "top",
         row.valign="center", wrap.rownames=10, wrap.colnames=10, mar = c(0,0,3,0)+0.1)
title(main="Statistics for EDHEC Indexes")
```

---

table.Variability *Variability Summary: Statistics and Stylized Facts*

Description

Table of Mean absolute difference, period standard deviation and annualised standard deviation

Usage

```
table.Variability(R, scale = NA, geometric = TRUE, digits = 4)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
scale	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE
digits	number of digits to round results to

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.65

See Also

StdDev.annualized
MeanAbsoluteDeviation

Examples

```
data(managers)
table.Variability(managers[, 1:8])


# don't test on CRAN, since it requires Suggested packages

require("Hmisc")
result <- t(table.Variability(managers[, 1:8]))

textplot(format.df(result, na.blank = TRUE, numeric.dollar = FALSE, cdec = c(3, 3, 1)),
  rmar = 0.8, cmar = 2, max.cex = .9, halign = "center", valign = "top",
  row.valign = "center", wrap.rownames = 20, wrap.colnames = 10,
  col.rownames = c("red", rep("darkgray", 5), rep("orange", 2)), mar = c(0, 0, 3, 0) + 0.1
)
title(main = "Portfolio variability")
```

---

test_returns *Sample sector returns for use by unit tests*

Description

A dataset containing returns for 10 sectors over 5 month-end periods.

Usage

```
test_returns
```

Format

A data frame with 5 rows and 10 variables:

Sector1 returns for Sector1, numeric

Sector2 returns for Sector2, numeric

Sector3 returns for Sector3, numeric

Sector4 returns for Sector4, numeric

Sector5 returns for Sector5, numeric

Sector6 returns for Sector6, numeric

Sector7 returns for Sector7, numeric

Sector8 returns for Sector8, numeric

Sector9 returns for Sector9, numeric

Sector10 returns for Sector10, numeric

test_weights *Sample sector weights for use by unit tests*

Description

A dataset containing weights for 10 sectors over 5 month-end periods.

Usage

```
test_weights
```

Format

A data frame with 5 rows and 10 variables:

Sector1 weight in Sector1, numeric

Sector2 weight in Sector2, numeric

Sector3 weight in Sector3, numeric

Sector4 weight in Sector4, numeric

Sector5 weight in Sector5, numeric

Sector6 weight in Sector6, numeric

Sector7 weight in Sector7, numeric

Sector8 weight in Sector8, numeric

Sector9 weight in Sector9, numeric

Sector10 weight in Sector10, numeric

to.period.contributions *Aggregate contributions through time*

Description

Higher frequency contributions provided as a time series are converted to a lower frequency for a specified calendar period.

Usage

```
to.period.contributions(
  Contributions,
  period = c("years", "quarters", "months", "weeks", "all")
)
```

Arguments

Contributions	a time series of the per period contribution to portfolio return of each asset
period	period to convert to. See details. "weeks", "months", "quarters", "years", or "all".

Details

From the portfolio contributions of individual assets, such as those of a particular asset class or manager, the multiperiod contribution is neither summable from nor the geometric compounding of single-period contributions. Because the weights of the individual assets change through time as transactions occur, the capital base for the asset changes.

Instead, the asset's multiperiod contribution is the sum of the asset's dollar contributions from each period, as calculated from the wealth index of the total portfolio. Once contributions are expressed in cumulative terms, asset contributions then sum to the returns of the total portfolio for the period.

Valid period character strings for period include: "weeks", "months", "quarters", "years", or "all". These are calculated internally via endpoints. See that function's help page for further details.

For the special period "all", the contribution is calculated over all rows, giving a single contribution across all observations.

Author(s)

Peter Carl, with thanks to Paolo Cavatore

References

Morningstar, *Total Portfolio Performance Attribution Methodology*, p.36. Available at https://morningstardirect.morningstar.com/clientcomm/Morningstar-Total-Portfolio-Performance-Attribution-Me
pdf

See Also

Return.portfolio
endpoints

Examples

```
data(managers, package="PerformanceAnalytics")

res_qtr_rebal = Return.portfolio(  managers["2002::",1:5]
                                 , weights=c(.05,.1,.3,.4,.15)
                                 , rebalance_on = "quarters"
                                 , verbose=TRUE)

to.period.contributions(res_qtr_rebal$contribution, period="years")
to.yearly.contributions(res_qtr_rebal$contribution)
```

`TotalRisk` *Total risk of the return distribution*

Description

The square of total risk is the sum of the square of systematic risk and the square of specific risk. Specific risk is the standard deviation of the error term in the regression equation. Both terms are annualized to calculate total risk.

Usage

```
TotalRisk(Ra, Rb, Rf = 0, ...)
```

Arguments

`Ra`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rb`	return vector of the benchmark asset
`Rf`	risk free rate, in same period as your returns
`...`	any other passthru parameters

Details

$$TotalRisk = \sqrt{SystematicRisk^2 + SpecificRisk^2}$$

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.75

Examples

```
data(portfolio_bacon)
print(TotalRisk(portfolio_bacon[,1], portfolio_bacon[,2])) #expected 0.0134

data(managers)
print(TotalRisk(managers['1996',1], managers['1996',8]))
print(TotalRisk(managers['1996',1:5], managers['1996',8]))
```

`TrackingError` *Calculate Tracking Error of returns against a benchmark*

Description

A measure of the unexplained portion of performance relative to a benchmark.

Usage

```
TrackingError(Ra, Rb, scale = NA)
```

Arguments

`Ra`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rb`	return vector of the benchmark asset
`scale`	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)

Details

Tracking error is calculated by taking the square root of the average of the squared deviations between the investment's returns and the benchmark's returns, then multiplying the result by the square root of the scale of the returns.

$$TrackingError = \sqrt{\sum \frac{(R_a - R_b)^2}{len(R_a)\sqrt{scale}}}$$

Author(s)

Peter Carl

References

Sharpe, W.F. The Sharpe Ratio,*Journal of Portfolio Management*,Fall 1994, 49-58.

See Also

`InformationRatio` `TrackingError`

Examples

```
data(managers)
TrackingError(managers[,1,drop=FALSE], managers[,8,drop=FALSE])
TrackingError(managers[,1:6], managers[,8,drop=FALSE])
TrackingError(managers[,1:6], managers[,8:7,drop=FALSE])
```

`TreynorRatio` *calculate Treynor Ratio or modified Treynor Ratio of excess return over CAPM beta*

Description

The Treynor ratio is similar to the Sharpe Ratio, except it uses beta as the volatility measure (to divide the investment's excess return over the beta).

Usage

```
TreynorRatio(Ra, Rb, Rf = 0, scale = NA, modified = FALSE)
```

Arguments

`Ra`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`Rb`	return vector of the benchmark asset
`Rf`	risk free rate, in same period as your returns
`scale`	number of periods in a year (daily scale = 252, monthly scale = 12, quarterly scale = 4)
`modified`	a boolean to decide whether to return the Treynor ratio or Modified Treynor ratio

Details

To calculate modified Treynor ratio, we divide the numerator by the systematic risk instead of the beta.

Equation:

$$TreynorRatio = \frac{\overline{(R_a - R_f)}}{\beta_{a,b}}$$

$$ModifiedTreynorRatio = \frac{r_p - r_f}{\sigma_s}$$

Author(s)

Peter Carl, Matthieu Lestel

References

https://en.wikipedia.org/wiki/Treynor_ratio, Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.77

See Also

`SharpeRatio` `SortinoRatio` `CAPM.beta`

Examples

```
data(portfolio_bacon)
data(managers)
round(TreynorRatio(managers[,1], managers[,8], Rf=.035/12),4)
round(TreynorRatio(managers[,1], managers[,8], Rf = managers[,10]),4)
round(TreynorRatio(managers[,1:6], managers[,8], Rf=.035/12),4)
round(TreynorRatio(managers[,1:6], managers[,8], Rf = managers[,10]),4)
round(TreynorRatio(managers[,1:6], managers[,8:7], Rf=.035/12),4)
round(TreynorRatio(managers[,1:6], managers[,8:7], Rf = managers[,10]),4)

print(TreynorRatio(portfolio_bacon[,1], portfolio_bacon[,2], modified = TRUE)) #expected 0.7975

print(TreynorRatio(managers['2002',1], managers['2002',8], modified = TRUE))
print(TreynorRatio(managers['2002',1:5], managers['2002',8], modified = TRUE))
```

UlcerIndex *calculate the Ulcer Index*

Description

Developed by Peter G. Martin in 1987 (Martin and McCann, 1987) and named for the worry caused to the portfolio manager or investor. This is similar to drawdown deviation except that the impact of the duration of drawdowns is incorporated by selecting the negative return for each period below the previous peak or high water mark. The impact of long, deep drawdowns will have significant impact because the underperformance since the last peak is squared.

Usage

```
UlcerIndex(R, ...)
```

Arguments

R	a vector, matrix, data frame, timeSeries or zoo object of asset returns
...	any other passthru parameters

Details

UI = sqrt(sum[i=1,2,...,n](D'_i^2/n)) where D'_i = drawdown since previous peak in period i

DETAILS: This approach is sensitive to the frequency of the time periods involved and penalizes managers that take time to recover to previous highs.

REFERENCES: Martin, P. and McCann, B. (1989) The investor's Guide to Fidelity Funds: Winning Strategies for Mutual Fund Investors. John Wiley & Sons, Inc. Peter Martin's web page on UI: https://www.tangotools.com/ui/ui.htm

Test against spreadsheet at: https://www.tangotools.com/ui/UlcerIndex.xls

Author(s)

Matthieu Lestel

`unique-comoments` *Helper function for comoment matrices*

Description

transforms vector with unique coskewness or cokurtosis elements to the full coskewness and cokurtosis matrix. Also works in the reverse direction by extracting the unique coskewness and cokurtosis elements from full the coskewness or cokurtosis matrices.

Usage

```
M3.vec2mat(M3, p)

M3.mat2vec(M3)

M4.vec2mat(M4, p)

M4.mat2vec(M4)
```

Arguments

`M3`	matrix of dimension p x p^2, or a vector with (p * (p + 1) * (p + 2) / 6) unique coskewness elements
`p`	number of variables; the number of instruments for which the coskewness or cokurtosis matrix/vector was computed
`M4`	matrix of dimension p x p^3, or a vector with (p * (p + 1) * (p + 2) * (p + 3) / 12) unique coskewness elements

Details

For documentation on the coskewness and cokurtosis matrices, we refer to ?CoMoments. Both the full matrices and reduced form can be the output of M3.MM and M4.MM, depending on the optional argument as.mat.

Author(s)

Kris Boudt, Peter Carl, Dries Cornilly, Brian Peterson

See Also

CoMoments
ShrinkageMoments
EWMAMoments
StructuredMoments
MCA
NCE

Examples

```
data(managers)
p <- ncol(edhec)

# transform coskewness between matrix and vector format
m3 <- M3.MM(edhec, as.mat=TRUE)
m3bis <- M3.vec2mat(M3.MM(edhec, as.mat=FALSE), p)
sum((m3 - m3bis)^2)

# transform cokurtosis between matrix and vector format
m4 <- M4.MM(edhec, as.mat=FALSE)
m4bis <- M4.mat2vec(M4.MM(edhec, as.mat=TRUE))
sum((m4 - m4bis)^2)
```

UpDownRatios *calculate metrics on up and down markets for the benchmark asset*

Description

Calculate metrics on how the asset in R performed in up and down markets, measured by periods when the benchmark asset was up or down.

Usage

```
UpDownRatios(
  Ra,
  Rb,
  method = c("Capture", "Number", "Percent"),
  side = c("Up", "Down"),
  geometric = TRUE
)
```

Arguments

Ra	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
Rb	return vector of the benchmark asset
method	"Capture", "Number", or "Percent" to indicate which measure to return

side	"Up" or "Down" market statistics
geometric	utilize geometric chaining (TRUE) or simple/arithmetic chaining (FALSE) to aggregate returns, default TRUE

Details

This is a function designed to calculate several related metrics:

Up (Down) Capture Ratio: this is a measure of an investment's compound return when the benchmark was up (down) divided by the benchmark's compound return when the benchmark was up (down). The greater (lower) the value, the better.

Up (Down) Number Ratio: similarly, this is a measure of the number of periods that the investment was up (down) when the benchmark was up (down), divided by the number of periods that the Benchmark was up (down).

Up (Down) Percentage Ratio: this is a measure of the number of periods that the investment outperformed the benchmark when the benchmark was up (down), divided by the number of periods that the benchmark was up (down). Unlike the prior two metrics, in both cases a higher value is better.

Author(s)

Peter Carl

References

Bacon, C. *Practical Portfolio Performance Measurement and Attribution*. Wiley. 2004. p. 47

Examples

```
data(managers)
UpDownRatios(managers[, 1, drop = FALSE], managers[, 8, drop = FALSE])
UpDownRatios(managers[, 1:6, drop = FALSE], managers[, 8, drop = FALSE])
UpDownRatios(managers[, 1, drop = FALSE], managers[, 8, drop = FALSE], method = "Capture")
# Up Capture:
UpDownRatios(managers[, 1, drop = FALSE], managers[, 8, drop = FALSE],
  side = "Up", method = "Capture"
)
# Down Capture:
UpDownRatios(managers[, 1, drop = FALSE], managers[, 8, drop = FALSE],
  side = "Down", method = "Capture"
)
```

`UpsideFrequency` *upside frequency of the return distribution*

Description

To calculate Upside Frequency, we take the subset of returns that are more than the target (or Minimum Acceptable Returns (MAR)) returns and divide the length of this subset by the total number of returns.

Usage

```
UpsideFrequency(R, MAR = 0, ...)
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`MAR`	Minimum Acceptable Return, in the same periodicity as your returns
`...`	any other passthru parameters

Details

$$UpsideFrequency(R, MAR) = \sum_{t=1}^{n} \frac{max[(R_t - MAR), 0]}{R_t * n}$$

where n is the number of observations of the entire series

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.94

Examples

```
data(portfolio_bacon)
MAR = 0.005
print(UpsideFrequency(portfolio_bacon[,1], MAR)) #expected 0.542

data(managers)
print(UpsideFrequency(managers['1996']))
print(UpsideFrequency(managers['1996',1])) #expected 0.75
```

`UpsidePotentialRatio` *calculate Upside Potential Ratio of upside performance over downside risk*

Description

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. That measure is the `SortinoRatio`. This function, Upside Potential Ratio, was a further improvement, extending the measurement of only upside on the numerator, and only downside of the denominator of the ratio equation.

Usage

```
UpsidePotentialRatio(R, MAR = 0, method = c("subset", "full"))
```

Arguments

`R`	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
`MAR`	Minimum Acceptable Return, in the same periodicity as your returns
`method`	one of "full" or "subset", indicating whether to use the length of the full series or the length of the subset of the series above(below) the MAR as the denominator, defaults to "subset"

Details

calculate Upside Potential Ratio of upside performance over downside risk

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. That measure is the `SortinoRatio`. This function, Upside Potential Ratio, was a further improvement, extending the measurement of only upside on the numerator, and only downside of the denominator of the ratio equation.

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.

$$UPR = \frac{\sum_{t=1}^{n}(R_t - MAR)}{\delta_{MAR}}$$

where δ_{MAR} is the `DownsideDeviation`.

The numerator in UpsidePotentialRatio only uses returns that exceed the MAR, and the denominator (in DownsideDeviation) only uses returns that fall short of the MAR by default. Sortino contends that this is a more accurate and balanced protrayal of return potential, wherase SortinoRatio can reward managers most at the peak of a cycle, without adequately penalizing them for past mediocre performance. Others have used the full series, and this is provided as an option by the method argument.

Author(s)

Brian G. Peterson

References

Sortino, F. and Price, L. Performance Measurement in a Downside Risk Framework. *Journal of Investing*. Fall 1994, 59-65.

Plantinga, A., van der Meer, R. and Sortino, F. The Impact of Downside Risk on Risk-Adjusted Performance of Mutual Funds in the Euronext Markets. July 19, 2001. Available at SSRN: https://www.ssrn.com/abstract=277352

See Also

SharpeRatio
SortinoRatio
DownsideDeviation
SemiVariance
SemiDeviation
InformationRatio

Examples

```
data(edhec)
UpsidePotentialRatio(edhec[, 6], MAR = .05 / 12) # 5 percent/yr MAR
UpsidePotentialRatio(edhec[, 1:6], MAR = 0)
```

UpsideRisk *upside risk, variance and potential of the return distribution*

Description

Upside Risk is the similar of semideviation taking the return above the Minimum Acceptable Return instead of using the mean return or zero. To calculate it, we take the subset of returns that are more than the target (or Minimum Acceptable Returns (MAR)) returns and take the differences of those to the target. We sum the squares and divide by the total number of returns and return the square root.

Usage

```
UpsideRisk(
  R,
  MAR = 0,
  method = c("full", "subset"),
  stat = c("risk", "variance", "potential"),
  ...
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
MAR	Minimum Acceptable Return, in the same periodicity as your returns
method	one of "full" or "subset", indicating whether to use the length of the full series or the length of the subset of the series below the MAR as the denominator, defaults to "full"
stat	one of "risk", "variance" or "potential" indicating whether to return the Upside risk, variance or potential
...	any other passthru parameters

Details

$$UpsideRisk(R, MAR) = \sqrt{\sum_{t=1}^{n} \frac{max[(R_t - MAR), 0]^2}{n}}$$

$$UpsideVariance(R, MAR) = \sum_{t=1}^{n} \frac{max[(R_t - MAR), 0]^2}{n}$$

$$UpsidePotential(R, MAR) = \sum_{t=1}^{n} \frac{max[(R_t - MAR), 0]}{n}$$

where n is either the number of observations of the entire series or the number of observations in the subset of the series falling below the MAR.

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008

Examples

```
data(portfolio_bacon)
MAR = 0.005
print(UpsideRisk(portfolio_bacon[,1], MAR, stat="risk")) #expected 0.02937
print(UpsideRisk(portfolio_bacon[,1], MAR, stat="variance")) #expected 0.08628
print(UpsideRisk(portfolio_bacon[,1], MAR, stat="potential")) #expected 0.01771

MAR = 0
data(managers)
print(UpsideRisk(managers['1996'], MAR, stat="risk"))
print(UpsideRisk(managers['1996',1], MAR, stat="risk")) #expected 1.820
```

VaR *calculate various Value at Risk (VaR) measures*

Description

Calculates Value-at-Risk(VaR) for univariate, component, and marginal cases using a variety of analytical methods.

Usage

```
VaR(
  R = NULL,
  p = 0.95,
  ...,
  method = c("modified", "gaussian", "historical", "kernel", "gpd", "lognormal",
    "montecarlo"),
  clean = c("none", "boudt", "geltner", "locScaleRob"),
  portfolio_method = c("single", "component", "marginal"),
  weights = NULL,
  mu = NULL,
  sigma = NULL,
  m3 = NULL,
  m4 = NULL,
  invert = TRUE,
  SE = FALSE,
  SE.control = NULL,
  p.tr = 0.97,
  init = c(1, 0.3),
  nsim = 10000
)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns

p	confidence level for calculation, default p=.95
...	any other passthru parameters
method	one of "modified","gaussian","historical", "kernel", "gpd", "lognormal", "montecarlo", see Details.
clean	method for data cleaning through Return.clean. Current options are "none", "boudt", "geltner", or "locScaleRob".
portfolio_method	one of "single","component","marginal" defining whether to do univariate, component, or marginal calc, see Details.
weights	portfolio weighting vector, default NULL, see Details
mu	If univariate, mu is the mean of the series. Otherwise mu is the vector of means of the return series, default NULL, see Details
sigma	If univariate, sigma is the variance of the series. Otherwise sigma is the covariance matrix of the return series, default NULL, see Details
m3	If univariate, m3 is the skewness of the series. Otherwise m3 is the coskewness matrix (or vector with unique coskewness values) of the returns series, default NULL, see Details
m4	If univariate, m4 is the excess kurtosis of the series. Otherwise m4 is the cokurtosis matrix (or vector with unique cokurtosis values) of the return series, default NULL, see Details
invert	TRUE/FALSE whether to invert the VaR measure. see Details.
SE	TRUE/FALSE whether to ouput the standard errors of the estimates of the risk measures, default FALSE.
SE.control	Control parameters for the computation of standard errors. Should be done using the RPESE.control function.
p.tr	for method="gpd", the threshold probability for GPD fit, default 0.97
init	for method="gpd", initial values for the Nelder-Mead optimization, default c(1.00, 0.3)
nsim	for method="montecarlo", number of simulations, default 10000

Value

VaR measures are evaluated and returned based on the portfolio_method and SE arguments:

- portfolio_method = "single": Returns a scalar or matrix (depending on the number of asset columns) of VaR estimates.
- portfolio_method = "component": Returns a list with three components: the univariate portfolio VaR, the scalar contribution of each component to the portfolio VaR, and a percentage risk contribution.
- portfolio_method = "marginal": Returns a matrix of marginal VaR calculations.

If SE = TRUE, the output will also include standard errors or confidence intervals:

- **Standard Methods** ("modified", "gaussian", "historical"): Leverages the RPESE package to return standard error estimates, appending rows such as se or IFiid to the VaR matrix.

- **Extreme Value Theory** (`method = "gpd"`)**:** Uses Profile Log-Likelihood ratio tests to calculate asymmetric confidence bounds, appending rows for the Lower Confidence Limit (`LCL`) and Upper Confidence Limit (`UCL`).

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.

Univariate VaR estimation methods

The VaR at a probability level p (e.g. 95%) is the p-quantile of the negative returns, or equivalently, is the negative value of the $c = 1 - p$ quantile of the returns. In a set of returns for which sufficently long history exists, the per-period Value at Risk is simply the quantile of the period negative returns :

$$VaR = q_{.99}$$

where $q_{.99}$ is the 99% empirical quantile of the negative return series.

This method is also sometimes called "historical VaR", as it is by definition *ex post* analysis of the return distribution, and may be accessed with `method="historical"`.

When you don't have a sufficiently long set of returns to use non-parametric or historical VaR, or wish to more closely model an ideal distribution, it is common to us a parmetric estimate based on the distribution. J.P. Morgan's RiskMetrics parametric mean-VaR was published in 1994 and this methodology for estimating parametric mean-VaR has become what most literature generally refers to as "VaR" and what we have implemented as `VaR`. See *Return to RiskMetrics: Evolution of a Standard*`https://www.msci.com/documents/10199/dbb975aa-5dc2-4441-aa2d-ae34ab5f0945`.

Parametric mean-VaR does a better job of accounting for the tails of the distribution by more precisely estimating shape of the distribution tails of the risk quantile. The most common estimate is a normal (or Gaussian) distribution $R \sim N(\mu, \sigma)$ for the return series. In this case, estimation of VaR requires the mean return $\bar{R}$, the return distribution and the variance of the returns σ. In the most common case, parametric VaR is thus calculated by

$$\sigma = variance(R)$$

$$VaR = -\bar{R} - \sqrt{\sigma} \cdot z_c$$

where z_c is the c-quantile of the standard normal distribution. Represented in R by `qnorm(c)`, and may be accessed with `method="gaussian"`.

Other forms of parametric mean-VaR estimation utilize a different distribution for the distribution of losses to better account for the possible fat-tailed nature of downside risk. The now-archived package `VaR` by Talgat Daniyarov contained methods for simulating and estimating lognormal and generalized Pareto distributions to overcome some of the problems with nonparametric or parametric mean-VaR calculations on a limited sample size or on potentially fat-tailed distributions. You may access the lognormal estimate using `method="lognormal"`. A Generalized Pareto Distribution estimate is available using `method="gpd"`. A Monte Carlo simulation estimate is available using `method="montecarlo"`. The `VaR.backtest` function may be used to apply a binomial proportion of failures test to any of these VaR estimates to evaluate their ex-post accuracy.

The limitations of mean Value-at-Risk are well covered in the literature. The limitations of traditional mean-VaR are all related to the use of a symetrical distribution function. Use of simulations, resampling, or Pareto distributions all help in making a more accurate prediction, but they are still flawed for assets with significantly non-normal (skewed or kurtotic) distributions. Zangari (1996) and Favre and Galeano(2002) provide a modified VaR calculation that takes the higher moments of non-normal distributions (skewness, kurtosis) into account through the use of a Cornish Fisher expansion, and collapses to standard (traditional) mean-VaR if the return stream follows a standard distribution. This measure is now widely cited and used in the literature, and is usually referred to as "Modified VaR" or "Modified Cornish-Fisher VaR". They arrive at their modified VaR calculation in the following manner:

$$z_{cf} = z_c + \frac{(z_c^2 - 1)S}{6} + \frac{(z_c^3 - 3z_c)K}{24} - \frac{(2z_c^3 - 5z_c)S^2}{36}$$

$$Cornish - FisherVaR = -\bar{R} - \sqrt{(\sigma)} \cdot z_{cf}$$

where S is the skewness of R and K is the excess kurtosis of R.

Cornish-Fisher VaR collapses to traditional mean-VaR when returns are normally distributed. As such, the `VaR` and `VaR` functions are wrappers for the `VaR` function. The Cornish-Fisher expansion also naturally encompasses much of the variability in returns that could be uncovered by more computationally intensive techniques such as resampling or Monte-Carlo simulation. This is the default method for the `VaR` function, and may be accessed by setting `method="modified"`.

Favre and Galeano also utilize modified VaR in a modified Sharpe Ratio as the return/risk measure for their portfolio optimization analysis, see `SharpeRatio` for more information.

Component VaR

By setting `portfolio_method="component"` you may calculate the risk contribution of each element of the portfolio. The return from the function in this case will be a list with three components: the univariate portfolio VaR, the scalar contribution of each component to the portfolio VaR (these will sum to the portfolio VaR), and a percentage risk contribution (which will sum to 100%).

Both the numerical and percentage component contributions to VaR may contain both positive and negative contributions. A negative contribution to Component VaR indicates a portfolio risk diversifier. Increasing the position weight will reduce overall portoflio VaR.

If a weighting vector is not passed in via `weights`, the function will assume an equal weighted (neutral) portfolio.

Multiple risk decomposition approaches have been suggested in the literature. A naive approach is to set the risk contribution equal to the stand-alone risk. This approach is overly simplistic and neglects important diversification effects of the units being exposed differently to the underlying risk factors. An alternative approach is to measure the VaR contribution as the weight of the position in the portfolio times the partial derivative of the portfolio VaR with respect to the component weight.

$$C_i\text{VaR} = w_i \frac{\partial \text{VaR}}{\partial w_i}.$$

Because the portfolio VaR is linear in position size, we have that by Euler's theorem the portfolio VaR is the sum of these risk contributions. Gourieroux (2000) shows that for VaR, this mathematical decomposition of portfolio risk has a financial meaning. It equals the negative value of the asset's expected contribution to the portfolio return when the portfolio return equals the negative portfolio VaR:

$$C_i\text{VaR} == -E\left[w_i r_i | r_p = -\text{VaR}\right]$$

For the decomposition of Gaussian VaR, the estimated mean and covariance matrix are needed. For the decomposition of modified VaR, also estimates of the coskewness and cokurtosis matrices are needed. If r denotes the $Nx1$ return vector and mu is the mean vector, then the $N \times N^2$ co-skewness matrix is

$$m3 = E\left[(r-\mu)(r-\mu)' \otimes (r-\mu)'\right]$$

The $N \times N^3$ co-kurtosis matrix is

$$m_4 = E\left[(r-\mu)(r-\mu)' \otimes (r-\mu)' \otimes (r-\mu)'\right]$$

where $\otimes$ stands for the Kronecker product. The matrices can be estimated through the functions `skewness.MM` and `kurtosis.MM`. More efficient estimators have been proposed by Martellini and Ziemann (2007) and will be implemented in the future.

As discussed among others in Cont, Deguest and Scandolo (2007), it is important that the estimation of the VaR measure is robust to single outliers. This is especially the case for modified VaR and its decomposition, since they use higher order moments. By default, the portfolio moments are estimated by their sample counterparts. If `clean="boudt"` then the $1-p$ most extreme observations are winsorized if they are detected as being outliers. For more information, see Boudt, Peterson and Croux (2008) and `Return.clean`. If your data consist of returns for highly illiquid assets, then `clean="geltner"` may be more appropriate to reduce distortion caused by autocorrelation, see `Return.Geltner` for details.

Epperlein and Smillie (2006) introduced a non-parametric kernel estimator for component risk contributions, which is available via `method="kernel"` and `portfolio_method="component"`.

Marginal VaR

Different papers call this different things. In the Denton and Jayaraman paper referenced here, this calculation is called Incremental VaR. We have chosen the more common usage of calling this difference in VaR's in portfolios without the instrument and with the instrument as the "difference at the Margin", thus the name Marginal VaR. This is incredibly confusing, and hasn't been resolved in the literature at this time.

Simon Keel and David Ardia (2009) attempt to reconcile some of the definitional issues and address some of the shortcomings of this measure in their working paper titled "Generalized Marginal Risk". Hopefully their improved Marginal Risk measures may be included here in the future.

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)

Brian G. Peterson and Kris Boudt, Talgat Daniyarov

References

Daniyarov, T. *VaR: Value at Risk estimation*. CRAN Archive. 2004. https://cran.r-project.org/src/contrib/Archive/VaR/

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, L. and Ziemann, V., 2010. Improved estimates of higher-order comoments and implications for portfolio selection. Review of Financial Studies, 23(4):1467-1502.

Return to RiskMetrics: Evolution of a Standard https://www.msci.com/documents/10199/dbb975aa-5dc2-4441-aa2d-

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.

See Also

SharpeRatio
chart.VaRSensitivity
Return.clean

Examples

```
data(edhec)

# first do normal VaR calc
VaR(edhec, p = .95, method = "historical")

# now use Gaussian
VaR(edhec, p = .95, method = "gaussian")

# now use modified Cornish Fisher calc to take non-normal distribution into account
VaR(edhec, p = .95, method = "modified")

# now use p=.99
VaR(edhec, p = .99)
# or the equivalent alpha=.01
VaR(edhec, p = .01)

# now with outliers squished
VaR(edhec, clean = "boudt")

# add Component VaR for the equal weighted portfolio
VaR(edhec, clean = "boudt", portfolio_method = "component")

# end CRAN check
```

VaR.backtest *VaR Backtest*

Description

Provides a simple binomial backtest for a Value at Risk (VaR) model.

This function performs a Kupiec Proportion of Failures (POF) test to evaluate whether the number of times returns breached the VaR threshold matches the expected frequency `1-p`.

The original implementation was provided in the archived `VaR` package by Talgat Daniyarov.

Usage

```
VaR.backtest(R, VaR, p = 0.95)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of actual asset returns
VaR	estimated Value at Risk scalar or vector corresponding to the same periods
p	confidence level for calculation, default p=.95

Value

A list containing:

- expected_exceedances the expected number of VaR breaches.
- actual_exceedances the number of actual VaR breaches.
- p.value the p-value from the binomial proportion test.

Author(s)

Brian G. Peterson, Talgat Daniyarov

References

Daniyarov, T. *VaR: Value at Risk estimation*. CRAN Archive. 2004. https://cran.r-project.org/src/contrib/Archive/VaR/

See Also

VaR

Examples

```
data(edhec)
# calculate VaR at 95 percent confidence
v <- as.numeric(VaR(edhec[, 1], p = 0.95, method = "historical"))

# evaluate the backtest
VaR.backtest(edhec[, 1], v, p = 0.95)
```

VolatilitySkewness *Volatility and variability of the return distribution*

Description

Volatility skewness is a similar measure to omega but using the second partial moment. It's the ratio of the upside variance compared to the downside variance. Variability skewness is the ratio of the upside risk compared to the downside risk.

Usage

```
VolatilitySkewness(R, MAR = 0, stat = c("volatility", "variability"), ...)
```

Arguments

R	an xts, vector, matrix, data frame, timeSeries or zoo object of asset returns
MAR	Minimum Acceptable Return, in the same periodicity as your returns
stat	one of "volatility", "variability" indicating whether to return the volatility skewness or the variability skweness
...	any other passthru parameters

Details

$$VolatilitySkewness(R, MAR) = \frac{\sigma_U^2}{\sigma_D^2}$$

$$VariabilitySkewness(R, MAR) = \frac{\sigma_U}{\sigma_D}$$

where σ_U is the Upside risk and σ_D is the Downside Risk

Author(s)

Matthieu Lestel

References

Carl Bacon, *Practical portfolio performance measurement and attribution*, second edition 2008 p.97-98

Examples

```
data(portfolio_bacon)
MAR = 0.005
print(VolatilitySkewness(portfolio_bacon[,1], MAR, stat="volatility")) #expected 1.32
print(VolatilitySkewness(portfolio_bacon[,1], MAR, stat="variability")) #expected 1.15

MAR = 0
data(managers)
# print(VolatilitySkewness(managers['1996'], MAR, stat="volatility"))
print(VolatilitySkewness(managers['1996',1], MAR, stat="volatility"))
```

weights *Selected Portfolio Weights Data*

Description

An xts object that contains columns of monthly weights for a subset of the EDHEC hedge fund indexes that demonstrate rebalancing portfolios through time.

Note that all the EDHEC indices are available in edhec.

Usage

```
managers
```

Format

CSV conformed into an xts object with monthly observations

Details

A relatively random weights file used for charting examples.

Examples

```
data(weights)

#preview the data
head(weights)
```

zerofill *zerofill*

Description

Fill NA's with zeros in a time series to allow analysis when the data must be complete.

Usage

```
zerofill(x)
```

Arguments

x	time series to zero fill

Details

Note that this function has risks, use carefully. Complete data is preferred. Barring that, filling a small percentage of results in the middle of a large set is unlikely to cause problems. Barring that, realize that this will skew your results.

Index