Package ‘BLCOP’

October 12, 2022

Type Package

Title Black-Litterman and Copula Opinion Pooling Frameworks

Version 0.3.3

Description An implementation of the Black-Litterman Model and Attilio Meucci's copula opinion pooling framework as described in Meucci, Attilio (2005) <doi:10.2139/ssrn.848407>, Meucci, Attilio (2006) <doi:10.2139/ssrn.872577> and Meucci, Attilio (2008) <doi:10.2139/ssrn.1117574>.

License MIT + file LICENSE

LazyData true

NeedsCompilation no

URL `https://github.com/mangothecat/BLCOP`

BugReports `https://github.com/mangothecat/BLCOP/issues`

Imports methods, MASS, quadprog, RUnit (>= 0.4.22), timeSeries, fBasics, fMultivar, fPortfolio (>= 3011.81), rmarkdown, knitr

Suggests sn, corpcor, mnormt

VignetteBuilder knitr

Author Francisco Gochez [aut],
Richard Chandler-Mant [aut],
Suchen Jin [aut],
Jinjing Xie [aut],
Ava Yang [ctb] (Previous maintainer),
Joe Russell [cre]

Maintainer Joe Russell `<jrussell@mango-solutions.com>`

Repository CRAN

Date/Publication 2021-01-25 23:00:02 UTC

R topics documented:

`BLCOP data sets` *Monthly equity returns*

Description

A matrix holding time series of monthly returns (calculated from closing prices) for six stocks. The returns span the period from Jaunary 1998 through December 2003.

Usage

```
monthlyReturns
```

Format

A matrix with 6 columns and 71 rows. The names of the rows hold the dates of each series entry, and the column names are the names of the six equities from which the return series are taken.

Examples

```
CAPMList(monthlyReturns, marketIndex = sp500Returns, riskFree = US13wTB)
```

`BLCOPOptions` *Global package options*

Description

This function can be used to set or get global options for the BLCOP package.

Usage

```
BLCOPOptions(opt, setting)
```

Arguments

`opt`	A string with the name of an option
`setting`	The new setting for the option

Details

If `setting` is omitted, the current setting for `opt` is returned. If both arguments are omitted, a list with all of the settings is returned. The following settings may be changed: `regFunc`:Function used to perform the regression in `CAPMalphas` `numSimulations`:Number of monte-carlo simulations to perform in copula opinion pooling functions `unitTestPath`: Path where unit tests are located.

Value

If both arguments omitted, a list. If setting is omitted, value of an individual setting.

Author(s)

Francisco Gochez <fgochez@mango-solutions>

Examples

```
    BLCOPOptions("numSimulations")
```

BLPosterior *BLposterior*

Description

BLposterior

Usage

```
BLPosterior(returns, views, tau = 1, marketIndex, riskFree = NULL,
  kappa = 0, covEstimator = "cov")
```

Arguments

`returns`	A matrix of time series of returns. The columns should correspond to individual assets.
`views`	An object of class BLViews
`tau`	The "tau" parameter in the Black-Litterman model.
`marketIndex`	A set of returns of a market index.
`riskFree`	A time series of risk-free rates of return. Defaults to 0
`kappa`	if greater than 0, the confidences in each view are replaced. See the online help for details
`covEstimator`	A string holding the name of the function that should be used to estimate the variance-covariance matrix. This function should simply return a matrix.

Value

An object of class BLResult

Author(s)

Francisco

BLResult-class *Class "BLResult": posterior of a market distribution in the Black-Litterman sense*

Description

This class holds the posterior market mean and variance-covariance matrix calculated from some prior and set of views. The original views are also returned.

Objects from the Class

Objects can be created by calls of the form `new("BLResult", ...)`. However, it is intended that they be created by the function posteriorEst(or wrappers to that function).

Slots

`views`: Object of class `"BLViews"`. These are the original views used to calculate this posterior

`tau`: Object of class `"numeric"`. The value of "tau" used

`priorMean`: Object of class `"numeric"`: prior vector of market means

`priorCovar`: Object of class `"matrix"`: prior of the variance-covariance

`posteriorMean`: Object of class `"numeric"`: posterior mean

`posteriorCovar`: Object of class `"matrix"`: posterior variance-covariance

`kappa`: Object of class `"logical"`: logical flag indicating whether or not confidences-in-views were ignored.

Methods

denityPlots `signature(result = "BLResult")`: Plots the marginal distributions of the asset returns under the prior and posterior distributions

show `signature(object = "BLResult")`: Displays the contents of a result

optimalPortfolios.fPort `signature(result = "BLResult")`: Generates optimal prior and posterior portfolios using `fPortfolio` package routines

Author(s)

Francisco Gochez

`BLViews-class` *Class "BLViews" (Black-Litterman views)*

Description

An object that holds a set of analyst views, in the Black-Litterman sense, on a set of assets

Objects from the Class

Objects can be created by calls of the form `new("BLViews", ...)` or with the `BLViews` function.

Slots

`P`: Object of class `"matrix"`. The "pick" matrix

`qv`: Object of class `"numeric"`. Means of the views

`confidences`: Object of class `"numeric"`. Holds the confidence in each of the individual views

`assets`: Object of class `"character"`: Name of the asset "universe" to which these views apply

Methods

deleteViews `signature(views = "BLViews", viewsToDel = "numeric")`: Deletes a vector of views from the object, where the vector entries correspond to rows of the pick matrix

show `signature(object = "BLViews")`: Prints views in a user-friendly manner

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

Build Views *Create or add to a BLViews object*

Description

`BLViews` and `COPViews` are "constructors" for BLViews and COPViews objects respectively. `addBLViews` and `addCOPViews` allow one to easily add more views to a pre-existing views objects. `newPMatrix` is a utility function for creating pick matrices.

Usage

```
addBLViews(pickMatrix, q, confidences, views)
addCOPViews(pickMatrix, viewDist, confidences, views)
BLViews(P, q, confidences, assetNames)
COPViews(pickMatrix, viewDist, confidences, assetNames)
newPMatrix(assetNames, numViews, defaultValue = 0)
```

Arguments

`P`	"Pick" matrix with columns named after assets to which views correspond
`pickMatrix`	"Pick" matrix with columns named after assets to which views correspond
`q`	"q" vector of views
`confidences`	Vector of confidences in views. Note that confidences are recipricols of standard deviations
`viewDist`	A list of marginal distributions of the views
`views`	A BLViews object
`assetNames`	Names of the assets in the universe
`numViews`	Number of views in the pick matrix
`defaultValue`	Default value to use to fill the new pick matrix

Value

A BLViews or COPViews class object as appropriate. `newPMatrix` creates a matrix.

Author(s)

Francisco Gochez

See Also

`createBLViews`, `updateBLViews`

Examples

```
    ### example from T. M. Idzorek's paper "A STEP-BY-STEP GUIDE TO THE
### BLACK-LITTERMAN MODEL"
  ## Not run:
    pick <- newPMatrix(letters[1:8], 3)
    pick[1,7] <- 1
    pick[2,1] <- -1
    pick[2,2] <- 1
    pick[3, 3:6] <- c(0.9, -0.9, .1, -.1)
    confidences <- 1 / c(0.00709, 0.000141, 0.000866)
    myViews <- BLViews(pick, q = c(0.0525, 0.0025, 0.02), confidences, letters[1:8])
    myViews

    ### Modified COP example from Meucci's "Beyond Black-Litterman: Views on
### non-normal markets"
    dispersion <- c(.376,.253,.360,.333,.360,.600,.397,.396,.578,.775) / 1000
    sigma <- BLCOP:::.symmetricMatrix(dispersion, dim = 4)
    caps <- rep(1/4, 4)
    mu <- 2.5 * sigma
    dim(mu) <- NULL
    marketDistribution <- mvdistribution("mt", mean = mu, S = sigma, df = 5 )
    pick <- newPMatrix(c("SP", "FTSE", "CAC", "DAX"), 1)
    pick[1,4] <- 1
    vdist <- list(distribution("unif", min = -0.02, max = 0))
    views <- COPViews(pick, vdist, 0.2, c("SP", "FTSE", "CAC", "DAX"))

## End(Not run)
```

CAPMList — *Compute CAPM alphas for a set of assets*

Description

`CAPMList` is a helper function that computes the "alphas" and "betas" in the sense of the CAPM for series of asset returns. It is meant to be used for computing "prior" means for the Black-Litterman model.

Usage

```
CAPMList(returns, marketIndex, riskFree = NULL, regFunc = BLCOPOptions("regFunc"),
          coeffExtractFunc = NULL, ...)
```

Arguments

`returns`	A matrix or data.frame of asset returns, with different columns corresponding to different assets
`marketIndex`	A time series of returns for some market index (e.g. SP500)
`riskFree`	Risk-free rate of return
`regFunc`	The name of the function to used to regress the asset return series against the market index. This is set in the BLCOP options, and is `lm` by default.
`coeffExtractFunc`	A function that extracts the intercept (alpha) and coefficient of the market index (beta) from the results of a call to the regression function. It should return a vector containing these two elements.
`...`	Additional arguments to the regression function

Details

`coeffExtractFun` is needed because some regression functions such as `gls` from the `nlme` package don't return their results in the same format as `lm` does. If it is not supplied, a default that works with `lm` results is used.

Value

A `data.frame` with one column for the "alphas" and another for the "betas"

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

Examples

```
  library(MASS)
CAPMList(monthlyReturns, marketIndex = sp500Returns, riskFree = US13wTB, regFunc = "rlm")
```

`Construct views` *Create or add to a view object using a graphical interface*

Description

These helper functions allow one to easily create or add to an object of class BLViews or COPViews through the use of R's built-in data editor.

Usage

```
createBLViews(allAssets, numAssetViews = 1, assetSubset = NULL,
              mode = c("editor", "Window"))
updateBLViews(views, includeNullViews = FALSE, numNewViews = 0, assets = NULL)
createCOPViews (allAssets, numAssetViews = 1, assetSubset = NULL,
                mode = c("editor", "Window"))
```

Arguments

`allAssets`	A character vector holding the names of all of the assets in one's "universe"
`numAssetViews`	The number of views to form. Should be less than or equal to the total number of assets
`assetSubset`	A character vector of assets that is a subset of `allAssets`. Views will be formed only on this subset. By default, `assetSubset = allAssets`.
`mode`	Mode of GUI. Currently unused
`views`	Object of class BLViews
`assets`	Set of assets to form or modify views on. If NULL, will use the full set of assets
`includeNullViews`	When updating views, should the 0 columns of the pick matrix be included?
`numNewViews`	In `updateViews`, this is the number of new views to add

Details

`createCOPViews` does not allow one to specify the distributions of the views at the moment. Such a feature may be added later through another GUI. At the moment the object returned by this function has its distribution set to a default. `updateViews` allows one to modify pre-existing views

Value

An object of class `BLViews` or `COPViews` that holds all of the views created.

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

See Also

`addBLViews`, `addCOPViews`, `COPViews`, `BLViews`

Examples

```
    ## Not run:
        views <- createBLViews(colnames(monthlyReturns), 2)

## End(Not run)
```

`COPPosterior` *Calculate the posterior distribution of the market using copula opinion pooling*

Description

`COPPosteior` uses Attilio Meucci's copula opinion pooling method to incorporate an analyst's subjective views with a prior "official" market distribution. Both the views and the market may have an arbitrary distribution as long as it can be sampled in R. Calculations are done with monte-carlo simulation, and the object returned will hold samples drawn from the market posterior distribution.

Usage

```
COPPosterior(marketDist, views, numSimulations = BLCOPOptions("numSimulations"))
```

Arguments

`marketDist`	An object of class mvdistribution which describes the prior "official" distribution of the market.
`views`	An object of class COPViews which describe the subjective views on the market distribution
`numSimulations`	The number of monte carlo samples to draw during calculations. Each asset in one's universe will have numSimulations samples from the posterior.

Value

An object of class COPResult.

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

References

Attilio Meucci, "Beyond Black-Litterman:Views on Non-normal Markets". See also Attilio Meucci, "Beyond Black-Litterman in Practice: a Five-Step Recipe to Input Views on non-Normal Markets."

See Also

`BLPosterior`

Examples

```
        ## Not run:
          # An example based on one found in "Beyond Black-Litterman:Views on Non-normal Markets"
              dispersion <- c(.376,.253,.360,.333,.360,.600,.397,.396,.578,.775) / 1000
              sigma <- BLCOP:::.symmetricMatrix(dispersion, dim = 4)
              caps <- rep(1/4, 4)
              mu <- 2.5 * sigma
              dim(mu) <- NULL
              marketDistribution <- mvdistribution("mt", mean = mu, S = sigma, df = 5 )
          pick <- matrix(0, ncol = 4, nrow = 1, dimnames = list(NULL, c("SP", "FTSE", "CAC", "DAX")))
              pick[1,4] <- 1
              vdist <- list(distribution("unif", min = -0.02, max = 0))

              views <- COPViews(pick, vdist, 0.2, c("SP", "FTSE", "CAC", "DAX"))
              posterior <- COPPosterior(marketDistribution, views)

    ## End(Not run)
```

`COPResult-class` *Class "COPResult"*

Description

A class that holds the posterior distribution produced with the COP framework

Objects from the Class

Objects can be created by calls of the form `new("COPResult", ...)`. In general however they are created by the function `COPPosterior`

Slots

`views`**:** Object of class `"COPViews"`. These are the views that led to the result

`marketDist`**:** Object of class `"mvdistribution"`. Prior distribution of the market

`posteriorSims`**:** Object of class `"matrix"`. Matrices holding the simulations of the posteriors with a column for each asset.

Methods

densityPlots `signature(result = "COPResult")`: Generates density plots of the marginal prior and posterior distributions of each asset.

show `signature(result = "COPResult")`: Displays basic information about the posterior results

optimalPortfolios.fPort `signature(result = "COPResult")`: Generates optimal prior and posterior portfolios using `fPortfolio` package routines

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

See Also

`COPPosterior`, `BLResult-class`

`COPViews-class` *Class "COPViews" (copula opinion pooling views)*

Description

An object that holds a set of analyst views, in the copula opinion pooling sense, on a set of assets

Objects from the Class

Objects can be created by calls of the form `new("COPViews", ...)` or with the `COPViews` function.

Slots

`pick`**:** Object of class `"matrix"`. The pick matrix

`viewDist`**:** Object of class `"list"`. List of probability distributions of the views

`confidences`**:** Object of class `"numeric"`.

`assets`**:** Object of class `"character"`. Name of the asset "universe" to which these views apply.

Methods

deleteViews `signature(views = "COPViews", viewsToDel = "numeric")`: Deletes a vector of views from the object, where the vector entries correspond to rows of the pick matrix

show `signature(object = "COPViews")`: Prints views in a user-friendly manner

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

See Also

`BLViews`, `COPViews`, `addCOPViews`, `createCOPViews`

Examples

```
showClass("COPViews")
```

`deleteViews` *Delete individual views from view objects*

Description

A generic function that allows one to delete individual views from objects of class `BLViews` or `COPViews`. The inputs are a view object and a numeric vector of views to delete, where the entires of the vector map to rows of the pick matrix.

Usage

```
deleteViews(views, viewsToDel)
```

Arguments

`views`	An object of class `BLViews` or `COPViews`
`viewsToDel`	A numeric vector of views to delete, as described above

Value

The original object with the indicated views deleted

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

See Also

`BLViews-class`, `COPViews-class`

Examples

```
stocks <- colnames(monthlyReturns)
pick <- matrix(0, ncol = 6, nrow = 2, dimnames = list(NULL, stocks))
pick[1,"IBM"] <- 1
pick[1, "DELL"] <- 0.04
pick[2, "C"] <- 1
pick[2, "JPM"] <- 0.6

confidences <- 1 / c(0.7, 0.1)

views <- BLViews( P = pick, q = c(0.1,0.1) , confidences = confidences,stocks)
deleteViews(views, 1)
```

`densityPlots` *Density plots of prior and posterior distributions*

Description

This generic function generates density plots of the marginal posterior and prior distributions of a set of assets in an object of class `BLResult` or `COPResult` for comparative purposes.

Usage

```
densityPlots(result, assetsSel = NULL, numSimulations = BLCOPOptions("numSimulations"),
             ...)
```

Arguments

`result`	Object of class
`assetsSel`	A numeric vector of assets to plot
`numSimulations`	For `COPResult` class objects, the number of simulations to use for the market posterior distribution
`...`	Additional arguments passed to `plot`

Details

For `COPResults` objects, density kernel estimates from the samples are used

Value

None

Author(s)

Francisco Gochez, <fgochez@mango-solutions>

Examples

```
    ## Not run:
        dispersion <- c(.376,.253,.360,.333,.360,.600,.397,.396,.578,.775) / 1000
        sigma <- BLCOP:::.symmetricMatrix(dispersion, dim = 4)
        caps <- rep(1/4, 4)
        mu <- 2.5 * sigma
        dim(mu) <- NULL
        marketDistribution <- mvdistribution("mt", mean = mu, S = sigma, df = 5 )
     pick <- matrix(0, ncol = 4, nrow = 1, dimnames = list(NULL, c("SP", "FTSE", "CAC", "DAX")))
        pick[1,4] <- 1
        vdist <- list(distribution("unif", min = -0.02, max = 0))

        views <- COPViews(pick, vdist, 0.2, c("SP", "FTSE", "CAC", "DAX"))
        posterior <- COPPosterior(marketDistribution, views)
        densityPlots(posterior, 4)

## End(Not run)
```

Distribution class constructors

Constructors for distribution and mvdistribution class objects

---

Description

These functions create objects of class `distribution` and `mvdistribution`

Usage

```
mvdistribution(RName, ...)
distribution(RName, ...)
```

Arguments

`RName`	A string holding the R suffix corresponding to the distribution, e.g. "pois" for the Poisson distribution
`...`	Additional parameters that parametrize the distribution

Details

In general any distribution with a corresponding sampling function can be used. This function should have the name given in `RName` but preceded with an "r", e.g. `rnorm` for the normal distribution. When the constructors are called, they check that the given sampling function exists and that it takes the arguments that were passed in the `...`.

Value

An object of class `distribution` or `mvdistribution`.

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

See Also

`sampleFrom`

Examples

```
    ## Not run:
    # create a uniform distribution object and sample from it
    myUnif <- distribution("unif", min = -0.1, max = 0.1)
    hist(sampleFrom(myUnif, 1000))

    mvNormal <- mvdistribution("mnorm", mean = c(1, 5), varcov = diag(c(2, 0.1)))
    x <- sampleFrom(mvNormal, 1000)
    plot(x[,1] ~ x[,2])

## End(Not run)
```

`distribution-class` *Class "distribution"*

Description

A class that describes univariate distributions

Objects from the Class

Objects can be created by calls of the form `new("distribution", ...)`. There is also a constructor which is also named `distribution`.

Slots

`RName`: Object of class `"character"`. This is the R "suffix" of the distirbution.

`parameters`: Object of class `"numeric"`. A named numeric vector that holds the parameters of the distribution

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

See Also

`distribution`, `mvdistribution`, `mvdistribution`

Examples

```
    showClass("distribution")
```

`Estimators`	*Get prior and posterior estimators stored in package scope*

Description

These functions are not intended to be called directly by the user but exist to allow third party optimizer routines to access prior and posterior estimators calculated as part of the portfolio optimisation.

Usage

```
getPriorEstim(x, spec=NULL, ...)
getPosteriorEstim(x, spec=NULL, ...)
```

Arguments

`x`	multivariate time series
`spec`	optional portfolio specification
`...`	additional arguments

Value

A list with 2 elements:

`mu`	estimate of mean
`Sigma`	estimate of covariance

Author(s)

Richard Chandler-Mant <rchandler-mant@mango-solutions.com>

`Extractors`	*Extract various fields of view or posterior objects*

Description

A collection of functions to extract several fields of BLViews, COPViews, COPPosterior and BLPosterior objects.

Usage

```
assetSet(views)
viewMatrix(views, dropZeroColumns = TRUE)
PMatrix(views)
confidences(views)
posteriorMeanCov(posterior)
posteriorSimulations(posterior)
numSimulations(posterior)
priorViews(posterior)
```

Arguments

`views`	An object of class BLViews or COPViews
`posterior`	An object of class BLPosterior (posteriorMeanCov) or COPPosterior (posteriorSimulations, priorViews) , as appropriate
`dropZeroColumns`	Logical flag. If TRUE, columns of "view matrix" which only have zeros are dropped

Value

`assetSet`	The names of the assets in the view object's universe
`confidences`	The set of confidences in each view.
`PMatrix`	The 'pick' matrix
`viewMatrix`	The pick matrix augmented with the q vector of the BL model
`posteriorMeanCov`	The posterior mean and covariance (in a list) of a BLPosterior object
`posteriorSimulations`	Matrix of posterior distribution simulations held in a COPPosterior object
`numSimulations`	Number of simulations in posterior COP distribution

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

Examples

```
pick <- matrix(0, ncol = 4, nrow = 1, dimnames = list(NULL, c("SP", "FTSE", "CAC", "DAX")))
  pick[1,4] <- 1
  vdist <- list(distribution("unif", min = -0.02, max = 0))
  views <- COPViews(pick, vdist, 0.2, c("SP", "FTSE", "CAC", "DAX"))
  assetSet(views)
  confidences(views)
  PMatrix(views)
```

`mvdistribution-class` *Class "mvdistribution"*

Description

A class that describes multivariate distributions

Objects from the Class

Objects can be created by calls of the form `new("distribution", ...)`. There is also a constructor which is also named `mvdistribution`.

Slots

`RName`**:** Object of class `"character"`. This is the R "suffix" of the distirbution.

`parameters`**:** A named list of parameters that characterize the distribution

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

See Also

`distribution`, `mvdistribution`, `distribution-class`

Examples

```
showClass("mvdistribution")
```

`optimalPortfolios` *Calculates optimal portfolios under prior and posterior distributions*

Description

These are wrapper functions that calculate optimal portfolios under the prior and posterior return distributions. `optimalPortfolios` works with a user-supplied optimization function, though simple Markowitz minimum-risk optimization is done with `solve.QP` from `quadprog` if none is supplied. `optimalPortfolios.fPort` is a generic utility function which calculates optimal portfolios using routines from the `fPortfolio` package.

Usage

```
optimalPortfolios(result, optimizer = .optimalWeights.simpleMV, ..., doPlot = TRUE,
                  beside = TRUE)
optimalPortfolios.fPort(result, spec = NULL, constraints = "LongOnly",
                        optimizer = "minriskPortfolio", inputData = NULL,
                        numSimulations = BLCOPOptions("numSimulations"))
```

Arguments

`result`	An object of class `BLResult`
`optimizer`	For `optimalPortfolios`, An optimization function. It should take as arguments a vector of means and a variance-covariance matrix, and should return a vector of optimal weights. For `optimalPortfolios`, the name of a `fPortfolio` function that performs portfolio optimization
`spec`	Object of class `fPORTFOLIOSPEC`. If NULL, will use a basic mean-variance spec for Black-Litterman results, and a basic CVaR spec for COP results
`inputData`	Time series data (any form that can be coerced into a `timeSeries` object)
`constraints`	String of constraints that may be passed into `fPortfolio` optimization routines
`numSimulations`	For COP results only - the number of posterior simulations to use in the optimization (large numbers here will likely cause the routine to fail)
`...`	Additional arguments to the optimization function
`doPlot`	A logical flag. Should barplots of the optimal portfolio weights be produced?
`beside`	A logical flag. If a barplot is generated, should the bars appear side-by side? If `FALSE`, differences of weights will be plotted instead.

Details

By default, `optimizer` is a simple function that performs Markowitz optimization via `solve.QP`. In addition to a mean and variance, it takes an optional `constraints` parameter that if supplied should hold a named list with all of the parameters that `solve.QP` takes.

Value

`optimalPortfolios` will return a list with the following items:

`priorPFolioWeights`	The optimal weights under the prior distribution
`postPFolioWeights`	The optimal weights under the posterior distribution

`optimalPortfolios.fPort` will return a similar list with 2 elements of class `fPORTFOLIO`.

Note

It is expected that `optimalPortfolios` will be deprecated in future releases in favour of `optimalPortfolios.fPort`.

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

References

Wuertz, D., Chalabi, Y., Chen W., Ellis A. (2009); Portfolio Optimization with R/Rmetrics, Rmetrics eBook, Rmetrics Association and Finance Online, Zurich.

Examples

```
## Not run:
    entries <- c(0.001005,0.001328,-0.000579,-0.000675,0.000121,0.000128,
                    -0.000445, -0.000437, 0.001328,0.007277,-0.001307,-0.000610,
                    -0.002237,-0.000989,0.001442,-0.001535, -0.000579,-0.001307,
                    0.059852,0.027588,0.063497,0.023036,0.032967,0.048039,-0.000675,
                    -0.000610,0.027588,0.029609,0.026572,0.021465,0.020697,0.029854,
                    0.000121,-0.002237,0.063497,0.026572,0.102488,0.042744,0.039943,
                    0.065994 ,0.000128,-0.000989,0.023036,0.021465,0.042744,0.032056,
                    0.019881,0.032235 ,-0.000445,0.001442,0.032967,0.020697,0.039943,
                    0.019881,0.028355,0.035064 ,-0.000437,-0.001535,0.048039,0.029854,
                    0.065994,0.032235,0.035064,0.079958 )

    varcov <- matrix(entries, ncol = 8, nrow = 8)
    mu <- c(0.08, 0.67,6.41, 4.08, 7.43, 3.70, 4.80, 6.60) / 100
    pick <- matrix(0, ncol = 8, nrow = 3, dimnames = list(NULL, letters[1:8]))
    pick[1,7] <- 1
    pick[2,1] <- -1; pick[2,2] <- 1
    pick[3, 3:6] <- c(0.9, -0.9, .1, -.1)
    confidences <- 1 / c(0.00709, 0.000141, 0.000866)
    views <- BLViews(pick, c(0.0525, 0.0025, 0.02), confidences, letters[1:8])
    posterior <- posteriorEst(views, tau = 0.025, mu, varcov )
    optimalPortfolios(posterior, doPlot = TRUE)

     optimalPortfolios.fPort(posterior, optimizer = "tangencyPortfolio")

    # An example based on one found in "Beyond Black-Litterman:Views on Non-normal Markets"
        dispersion <- c(.376,.253,.360,.333,.360,.600,.397,.396,.578,.775) / 1000
        sigma <- BLCOP:::.symmetricMatrix(dispersion, dim = 4)
        caps <- rep(1/4, 4)
        mu <- 2.5 * sigma
        dim(mu) <- NULL
        marketDistribution <- mvdistribution("mt", mean = mu, S = sigma, df = 5 )
      pick <- matrix(0, ncol = 4, nrow = 1, dimnames = list(NULL, c("SP", "FTSE", "CAC", "DAX")))
        pick[1,4] <- 1
        vdist <- list(distribution("unif", min = -0.02, max = 0))

        views <- COPViews(pick, vdist, 0.2, c("SP", "FTSE", "CAC", "DAX"))
        posterior <- COPPosterior(marketDistribution, views)

       optimalPortfolios.fPort(myPosterior, spec = NULL, optimizer = "minriskPortfolio",
                        inputData = NULL, numSimulations  = 100 )


## End(Not run)
```

`posteriorEst` *This function performs the "core" calculation of the Black-Litterman model.*

Description

This function performs the "core" calculation of the Black-Litterman model.

Usage

```
posteriorEst(views, mu, tau = 0.5, sigma, kappa = 0)
```

Arguments

`views`	An object of class BLViews
`mu`	A vector of mean equilibrium returns
`tau`	The "tau" parameter in the Black-Litterman model.
`sigma`	The variance-covariance matrix of the returns of the assets
`kappa`	if greater than 0, the confidences in each view are replaced. See the online help for details

Value

An object of class BLResult holding the updated Black-Litterman posterior

Author(s)

Francisco

`posteriorFeasibility` *Calculate the "feasibility" of the (Black-Litterman) posterior mean*

Description

Attilio Meucci and Gianluca Fusai have suggested using the Mahalanobis distance to assess the feasibility of a set of Black-Litterman views. This function calculates this distance, along with a "feasibility" measure based on this distance and the sensitivity of the measure to changes in the "q" vector.

Usage

```
posteriorFeasibility(result)
```

Arguments

`result`	An object of class BLResult

Details

The feasibility measure proposed by Meucci and Fusai (see the references below) is 1 - F(m), where m is the Mahalanobis distance from from the prior mean calculated with respect to the prior distribution. F is the chi-squared CDF of n-degrees of freedom, where n is the number assets in one's universe. It should be noted that in Meucci and Fusai's paper, a version of Black-Litterman is used in which the tau parameter is always set to 1.

Value

`mahalDist`	Mahalonobis distance of posterior mean vector from prior mean
`mahalDistProb`	1 - F(mahalDist), where F is the CDF of the Chi-squared distribution with n = \#assets degrees of freedom
`sensitivities`	Derivatives of mahalDistProb with respect to the elements of the "q" vector in the set of views. Not yet implemented

Warning

It is not clear that the results produced by this routine are entirely sensible, though the calculation is very straightforward and seems to match the one discussed in the source paper. Use with caution.

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

References

Fusai, Gianluca and Meucci, Attilio. "Assessing Views", 2002. http://www.symmys.com/AttilioMeucci/Research/PublFinanc

Examples

```
pickMatrix <- matrix(c(rep(1/2, 2), -1,  rep(0, 3)), nrow = 1, ncol = 6 )
views <- BLViews(P = pickMatrix, q = 0.08,confidences =  100,
             assetNames = colnames(monthlyReturns))
marketPosterior <- BLPosterior(monthlyReturns, views, marketIndex = sp500Returns,
    riskFree = US13wTB)
posteriorFeasibility(marketPosterior)
```

`Replacer functions` *Various functions for modifying fields of view objects*

Description

These functions allow for direct replacement of fields of view objects such as the pick matrix and vector of confidences.

Usage

```
PMatrix(views) <- value
confidences(views) <- value
qv(views) <- value
```

Arguments

`views`	An object of class BLViews or COPViews, except in the case of qv<- which applies only to BLViews
`value`	A vector in `confidences<-` and `qv<-` or a matrix in `PMatrix<-`.

Value

The object is modified directly

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

Examples

```
## example from Thomas M. Idzorek's paper "A STEP-BY-STEP GUIDE TO THE BLACK-LITTERMAN MODEL"
x <- c(0.001005,0.001328,-0.000579,-0.000675,0.000121,0.000128,-0.000445,-0.000437 ,
     0.001328,0.007277,-0.001307,-0.000610,-0.002237,-0.000989,0.001442,-0.001535 ,
     -0.000579,-0.001307,0.059852,0.027588,0.063497,0.023036,0.032967,0.048039 ,
    -0.000675,-0.000610,0.027588,0.029609,0.026572,0.021465,0.020697,0.029854 ,
     0.000121,-0.002237,0.063497,0.026572,0.102488,0.042744,0.039943,0.065994 ,
     0.000128,-0.000989,0.023036,0.021465,0.042744,0.032056,0.019881,0.032235 ,
    -0.000445,0.001442,0.032967,0.020697,0.039943,0.019881,0.028355,0.035064 ,
    -0.000437,-0.001535,0.048039,0.029854,0.065994,0.032235,0.035064,0.079958 )

    varCov <- matrix(x, ncol = 8, nrow = 8)
    mu <- c(0.08, 0.67,6.41, 4.08, 7.43, 3.70, 4.80, 6.60) / 100
    pick <- matrix(0, ncol = 8, nrow = 3, dimnames = list(NULL, letters[1:8]))
    pick[1,7] <- 1
    pick[2,1] <- -1; pick[2,2] <- 1
    pick[3, 3:6] <- c(0.9, -0.9, .1, -.1)
    confidences <- 1 / c(0.000709, 0.000141, 0.000866)
    myViews <- BLViews(pick, c(0.0525, 0.0025, 0.02), confidences, letters[1:8])
    myPosterior <- posteriorEst(myViews, tau = 0.025, mu, varCov )
    myPosterior
    # increase confidences
    confidences(myViews) <-  1 / c(0.0001, 0.0001, 0.0005)
    myPosterior2 <- posteriorEst(myViews, tau = 0.025, mu, varCov )
    myPosterior2
```

runBLCOPTests — *Execute the BLCOP unit tests*

Description

Uses the RUnit package to execute a series of unit tests.

Usage

```
runBLCOPTests(testPath = BLCOPOptions("unitTestPath"), protocolFile = "BLCOPTests.html",
    writeProtocol = FALSE)
```

Arguments

`testPath`	Location of the unit tests.
`protocolFile`	Name of the html report file generated by the RUnit function printHTMLProtocol
`writeProtocol`	Logical flag. Should the above html report be produced?

Value

The summary of an object returned by RUnit's `runTestSuite`

Warning

These unit tests are in need of additional test cases, and should not be regarded as exhaustive in their current state.

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

Examples

```
## Not run:
    runBLCOPTests()

## End(Not run)
```

`sampleFrom` *Sample from a distribution object*

Description

Generates samples from a distribution held by an object of class `distribution` or `mvdistribution`. Intended mainly for internal use.

Usage

```
sampleFrom(dstn, n = 1)
```

Arguments

`dstn`	an object of class `distribution` or `mvdistribution`.
`n`	Number of samples to generate

Value

A vector or matrix of samples.

Author(s)

Francisco Gochez <fgochez@mango-solutions.com>

Examples

```
    x <- distribution("pois", lambda = 5)
    hist(sampleFrom(x, 1000), col = "blue", prob = TRUE)
```

`sp500Returns` *S\&P500 Returns*

Description

Monthly returns of the S&P 500 index for the period 2/2/1998 through 1/12/2003

Usage

```
sp500Returns
```

Format

A matrix with 1 column and 71 rows.

Examples

```
ts.plot(sp500Returns)
```

`US13wTB` *Risk free rate of return*

Description

The monthly rate of return of the US 13 week Treasury Bill for the period 30/1/1998 through 30/11/2003.

Usage

```
US13wTB
```

Format

A one-column matrix with 71 rows.

Examples

```
ts.plot(US13wTB)
```

Index