Package ‘RMOPI’

January 20, 2025

Title Risk Management and Optimization for Portfolio Investment

Version 1.1

Description Provides functions for risk management and portfolio investment of securities with practical tools for data processing and plotting. Moreover, it contains functions which perform the COS Method, an option pricing method based on the Fourier-cosine series (Fang, F. (2008) <doi:10.1137/080718061>).

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.2.1

Depends ggplot2, tibble, lubridate, timeSeries, xts

Imports MASS, PerformanceAnalytics, TTR, fPortfolio, rugarch, timeDate

NeedsCompilation no

Author Wei Ling [aut, cre],
Yang Liu [aut]

Maintainer Wei Ling <lingwei3418@163.com>

Repository CRAN

Date/Publication 2022-08-22 12:50:08 UTC

Contents

`CosPdfMulti` *Distribution Recovery with the COS method for Different parameters*

Description

Restore the distribution with the COS method under different parameters settings for error analysis.

Usage

```
CosPdfMulti(x, Chf, N, a, b)
```

Arguments

`x`	vector of observations
`Chf`	the characteristic function
`N`	the number of cos term for summation
`a`	the lower limit of the truncation interval
`b`	the upper limit of the truncation interval

Value

A matrix that contains restored p.d.f. with different parameters

Examples

```
N <- 2**(1:6)
x <- seq(-5, 5, by = 10 / (32 - 1))
a <- -10.0
b <- 10.0
CosPdfMulti(x, StNormChf, N, a, b)
```

CosPdfRecovery *Distribution Recovery with the COS method*

Description

Restore the distribution with the characteristic function through the COS method, an option pricing method based on the Fourier-cosine series.

Usage

```
CosPdfRecovery(x, Chf, N, a, b)
```

Arguments

x	vector of observations
Chf	the characteristic function
N	the number of cos term for summation
a	the lower limit of the truncation interval
b	the upper limit of the truncation interval

Value

The approximated probability density of x

References

Fang F. and Oosterlee C.W. 2008. "A Novel Pricing Method for European Options Based on Fourier-Cosine Series Expansions", Siam Journal on Scientific Computing. 31(2): 826-848. doi: 10.1137/080718061.

Examples

```
N <- 32
x <- seq(-5, 5, by = 10 / (32 - 1))
a <- -6.0
b <- 6.0
CosPdfRecovery(x, StNormChf, N, a, b)
```

CosValueOption — *Approximate the Option Price with the COS Method*

Description

Approximate the standard European call option price with the COS method, an option pricing method based on the Fourier-cosine series.

Usage

```
CosValueOption(ValueOption, GBMChf, r, tau, N, a, b, method = "integrate")
```

Arguments

`ValueOption`	the value function of the option
`GBMChf`	the characteristic function for GBM
`r`	the `r` parameter of GBM
`tau`	the `tau` parameter of GBM
`N`	the number of cos term for summation
`a`	the lower limit of the truncation interval
`b`	the upper limit of the truncation interval
`method`	how to calculate the integral, one of "integrate" and "jiahe"

Value

The approximated euro call option price

References

Fang F. and Oosterlee C.W. 2008. "A Novel Pricing Method for European Options Based on Fourier-Cosine Series Expansions", Siam Journal on Scientific Computing. 31(2): 826-848. doi: 10.1137/080718061.

Examples

```
r <- 0.1
sigmaS0 <- 0.2
tau <- 10
S0 <- 1
K <- 1
mu <- log(S0) + (r - 0.5 * sigmaS0^2) * tau
sigma <- sigmaS0 * sqrt(tau)
a <- -10
b <- 10
N <- 64
GBMChf <- function(u){NormChf(u,mu,sigma)}
ValueOption <- function(x){EuroCallOption(x,K)}
CosValueOption(ValueOption, GBMChf,r,tau, N, a, b)
```

`Describe` *Summary Statistics*

Description

Calculate useful statistics for an multivariate data.

Usage

```
Describe(data, digits = 2)
```

Arguments

`data`	vector of observations
`digits`	integer deciding the number of decimal places

Value

A tibble of statistics, including min, max, mean, sd, Q25, Q50, Q75, kurt, Skew, n, na

Examples

```
swan <- rGarch(len = 180)
Describe(tibble(a1 = swan, a2 = swan + 1), 2)
```

`DescribeVector` *Summary Statistics of Vector*

Description

Calculate useful statistics for an univariate data.

Usage

```
DescribeVector(data, digits = 2)
```

Arguments

`data`	vector of observations
`digits`	integer deciding the number of decimal places

Value

A tibble of statistics, including min, max, mean, sd, Q25, Q50, Q75, kurt, Skew, n, na

Examples

```
swan <- rGarch(len = 180)
DescribeVector(swan)
```

EuroCallOption *The Value Function of European Call Option*

Description

With global variable K, the strike price, calculate the value of European call option.

Usage

```
EuroCallOption(x, K)
```

Arguments

`x`	the stock price
`K`	the strike price

Value

The value of European call option

Examples

```
EuroCallOption(x = 2,K = 1)
```

FixBacktest *Buy and Hold Backtest*

Description

Backtest for the buy and hold with a fixed weights strategy.

Usage

```
FixBacktest(rets, weights)
```

Arguments

`rets`	historic multivariate returns
`weights`	holding weights of stock

Value

A backtest return series

Examples

```
names <- c("swan", "bear", "tiger")
date <- as.Date("2015-01-01") + days(0:179)
mu <- c(0.2, 0.08, 0.1)
sigma <- matrix(c(1, 0.25, -0.3, 0.25, 0.25, 0, -0.3, 0, 0.36), 3, 3)
allret <- rMvReturnSim(names, date, mu, sigma)
tsret <- as.timeSeries(allret)
FixBacktest(tsret, rep(1 / 3, 3))
```

F_k *F_k Coefficients*

Description

Calculate the F_k coefficients for the COS method, an option pricing method based on the Fourier-cosine series.

Usage

```
F_k(Chf, N, a, b)
```

Arguments

Chf	the characteristic function
N	the number of cos term for summation
a	the lower limit of the truncation interval
b	the upper limit of the truncation interval

Value

A vector of F_k coefficients

References

Fang F. and Oosterlee C.W. 2008. "A Novel Pricing Method for European Options Based on Fourier-Cosine Series Expansions", Siam Journal on Scientific Computing. 31(2): 826-848. doi: 10.1137/080718061.

Examples

```
N <- 32
a <- -6.0
b <- 6.0
F_k(StNormChf, N, a, b)
```

ggacf — *Plot the Acf Figure*

Description

Plot the Acf figure with observations of a single variable beautifully.

Usage

```
ggacf(data, lag = 10)
```

Arguments

`data`	vector of observations
`lag`	the maximum lag to calculate the acf

Value

A ggplot figure of the acf

Examples

```
swan <- rGarch(len = 180)
ggacf(swan^2, 20)
```

ggboxplot — *Plot the Box Figure*

Description

Plot the box figure beautifully with ggplot.

Usage

```
ggboxplot(data, mapping)
```

Arguments

`data`	a tibble
`mapping`	the mapping parameter of ggplot

Value

A box figure by ggplot

Examples

```
names <- c("swan", "bear", "tiger")
date <- as.Date("2015-01-01") + days(0:179)
mu <- c(0.2, 0.08, 0.1)
sigma <- matrix(c(1, 0.25, -0.3, 0.25, 0.25, 0, -0.3, 0, 0.36), 3, 3)
allret <- rMvReturnSim(names, date, mu, sigma)
totret <- StackRet(allret, date)
ggboxplot(totret, aes(x = stock, y = ret))
```

---

gghistplot *Plot the Histogram Figure*

Description

Plot the histgram figure beautifully with ggplot.

Usage

```
gghistplot(data, mapping, bins = 10)
```

Arguments

data	a tibble
mapping	the mapping parameter
bins	the number of bins

Value

A histogram figure by ggplot

Examples

```
date <- as.Date("2015-01-01") + days(0:180)
thero <- returns(rGbm("thero", date))[-1]
tthero <- tibble(x = date[-1], y = thero)
gghistplot(tthero, aes(x = thero, y = stat(density)), bins = 20)
```

gglineplot *Plot the Time Series*

Description

Plot the time series data beautifully with ggplot.

Usage

```
gglineplot(data, mapping, date_labels = "%Y/%m/%d", date_breaks = "2 weeks")
```

Arguments

data	a tibble
mapping	the mapping parameter
date_labels	the x label
date_breaks	the period of the x label

Value

A ggplot figure of the time series

Examples

```
date <- as.Date("2015-01-01") + days(0:180)
thero <- returns(rGbm("thero", date))[-1]
tthero <- tibble(x = date[-1], y = thero)
gglineplot(tthero, aes(x, y), "%Y/%m", "1 months")
```

ggpacf *Plot the Pacf Figure*

Description

Plot the Pacf figure with observations of a single variable beautifully.

Usage

```
ggpacf(data, lag = 10)
```

Arguments

data	vector of observations
lag	the maximum lag to calculate the pacf

Value

A ggplot figure of the pacf

Examples

```
swan <- rGarch(len = 180)
ggpacf(swan^2, 20)
```

`InvestmentPortfolio` *Construct Portfolio*

Description

Construct four types portfolio with specificition and constraints.

Usage

```
InvestmentPortfolio(data, method, spec, constraints = "LongOnly")
```

Arguments

`data`	multivariate returns, must be "timeSeries" type
`method`	porofolio type, one of "fea", "minrisk", "globalminrisk" and "sharp"
`spec`	specificition of portfolio
`constraints`	constraints of trade

Value

A portfolio

References

Markowitz H. 1952. "Portfolio Selection", The Journal of Finance, 7(1), 77–91. doi: 10.2307/2975974.

Examples

```
library(fPortfolio)
names <- c("swan", "bear", "tiger")
date <- as.Date("2015-01-01") + days(0:179)
mu <- c(0.2, 0.08, 0.1)
sigma <- matrix(c(1, 0.25, -0.3, 0.25, 0.25, 0, -0.3, 0, 0.36), 3, 3)
allret <- rMvReturnSim(names, date, mu, sigma)
tsret <- as.timeSeries(allret)
feaSpec <- portfolioSpec()
setWeights(feaSpec) <- rep(1 / 3, times = 3)
InvestmentPortfolio(tsret, "fea", feaSpec)
```

LogErrorCosPdf *Calculate the Absolute Error of the COS Method*

Description

Calculate the max absolute error of the cos method for different parameters given a vector of x.

Usage

```
LogErrorCosPdf(x, f, Chf, a, b, N)
```

Arguments

x	vector of observations
f	the true p.d.f.
Chf	the characteristic function
a	the lower limit of the truncation interval
b	the upper limit of the truncation interval
N	the number of cos term for summation

Value

A matrix that contains the log max error for different parameters

Examples

```
N <- c(1:200)
L <- c(10, 20, 60, 100, 1000)
a <- -L / 2
b <- L / 2
x <- seq(-5, 5, by = 10 / (32 - 1))
LogErrorCosPdf(x, dnorm, NormChf, a, b, N)
```

NormChf *The Characteristic Function of Normal Distribution*

Description

The Characteristic Function of Normal Distribution

Usage

```
NormChf(u, mu = 0, sigma = 1)
```

Arguments

`u`	observation
`mu`	the `mu` parameter
`sigma`	the `sigma` parameter

Value

The value of Characteristic Function

Examples

```
NormChf(1)
```

`PdfMultiPlot` *Plot the Probability Density Functions*

Description

Plot the p.d.f functions for the univariate distribution with data processed by StackRet.

Usage

```
PdfMultiPlot(data, x, y, Variable)
```

Arguments

`data`	a tibble contains x, y and Variable and the last one is the group variable
`x`	x
`y`	y
`Variable`	the group label

Value

A ggplot figure of the probability density functions

Examples

```
N <- 2**(1:6)
x <- seq(-5, 5, by = 10 / (32 - 1))
a <- -10.0
b <- 10.0
f_x1 <- CosPdfMulti(x, StNormChf, N, a, b)
colnames(f_x1) <- paste("N = 2 ^ ", c(1:6), sep = "")
mt1 <- StackRet(f_x1, x)
colnames(mt1) <- c("x", "y", "Variable")
PdfMultiPlot(mt1, x, y, Variable)
```

PdfSinglePlot *Plot the Probability Density Function*

Description

Plot the p.d.f function for the univariate distribution with x and y.

Usage

```
PdfSinglePlot(data, x, y)
```

Arguments

data	a tiible contains x and y
x	x
y	y

Value

A ggplot figure of the probability density function

Examples

```
N <- 32
x <- seq(-5, 5, by = 10 / (32 - 1))
a <- -6.0
b <- 6.0
f_x <- CosPdfRecovery(x, StNormChf, N, a, b)
tnorm <- tibble(x = x, y = f_x)
PdfSinglePlot(tnorm, x, y)
```

rGarch *Simulate a Garch Series*

Description

Simulate a Garch series given its data generate process with mean part.

Usage

```
rGarch(
  u = 0,
  a0 = rnorm(1, 0, 1),
  sigma20 = rnorm(1, 0, 1)^2,
  alpha = c(0.5, 0.5),
  beta = 0.25,
  len = 10
)
```

Arguments

`u`	the mean series
`a0`	vector of the start part
`sigma20`	vector of the initial variance `sigma2`
`alpha`	the `alpha` parameter
`beta`	the `beta` parameter
`len`	the length, include defined `a0`

Value

A simulated garch series

References

Bollerslev T. 1986. "Generalized autoregressive conditional heteroskedasticity", Journal of Econometrics, 31(3): 307-327. doi: 10.1016/0304-4076(86)90063-1.

Examples

```
rGarch()
```

---

rGarcha *Simulate a Garch Series*

Description

Simulate a Garch series given its data generate process without mean part.

Usage

```
rGarcha(
  a0 = rnorm(1, 0, 1),
  sigma20 = rnorm(1, 0, 1)^2,
  alpha = c(0.5, 0.5),
  beta = 0.25,
  len = 10
)
```

Arguments

`a0`	vector of the start part
`sigma20`	vector of the initial variance `sigma2`
`alpha`	the `alpha` parameter
`beta`	the `beta` parameter
`len`	the length, include defined `a0`

Value

A simulated garch series

References

Bollerslev T. 1986. "Generalized autoregressive conditional heteroskedasticity", Journal of Econometrics, 31(3): 307-327. doi: 10.1016/0304-4076(86)90063-1.

Examples

```
rGarcha()
```

`rGbm` *Simulate prices series of stocks*

Description

Simulate an multivariate series following Geometric Brownian Motion (GBM)

Usage

```
rGbm(name, time, start = 100, mu = 0.01, sigma = 0.02)
```

Arguments

`name`	vector of series names
`time`	vector of time, must be a "Date" type variable
`start`	vector of start positions
`mu`	vector of `mu`
`sigma`	vector of `sigma`

Value

a simulated multivariate GBM series

Examples

```
date <- as.Date("2015-01-01") + days(0:29)
rGbm(c("bear", "tiger", "swan"), date)
```

rGbms *Simulate Multivariate Stocks Prices Data*

Description

Simulate multivariate prices for interconnected stocks with each price series following Geometric Brownian Motion (GBM).

Usage

```
rGbms(
  name,
  len,
  start = c(1000, 1000),
  mu = rep(1e-04, 2),
  sigma = matrix(c(2e-04, 1e-04, 1e-04, 2e-04), 2, 2),
  digits = 2
)
```

Arguments

`name`	vector of series names
`len`	the length
`start`	vector of start positions
`mu`	vector of `mu`
`sigma`	vector of `sigma`
`digits`	integer deciding the number of decimal places

Value

A simulated multivariate GBM series with each series interconnected

Examples

```
rGbms(c("bear", "tiger"), len = 36)
```

rGbmSingle — *Simulate a single stock price series*

Description

Simulate an univariate series following Geometric Brownian Motion (GBM).

Usage

```
rGbmSingle(len, start = 100, mu = 0.01, sigma = 0.02)
```

Arguments

`len`	the length
`start`	the start position
`mu`	the `mu` parameter of GBM
`sigma`	the `sigma` parameter of GBM

Value

a simulated univariate GBM series

Examples

```
rGbmSingle(100)
```

RiskIndicators — *Calculate Useful Indicators for returns*

Description

Calculate cumulative return, annualized return, max drawdown, annualized sharp ratio, calmar ratio, sortino ratio, alpha, beta and information ratio with returns.

Usage

```
RiskIndicators(ret, rb, rf = 0)
```

Arguments

`ret`	vector of return
`rb`	return of market portfolio
`rf`	risk free rate

Value

A matrix of return and risk indicators

Examples

```
date <- as.Date("2015-01-01") + days(0:249)
ret <- as.xts(rnorm(250), date)
rb <- as.xts(rep(0, 250), date)
RiskIndicators(ret, rb = rb, rf = 0)
```

---

rMvReturnSim *Simulate Stocks Prices*

Description

Simulate stocks prices following multivariate normal distribution.

Usage

```
rMvReturnSim(
  names,
  date,
  mu = rep(0, 2),
  sigma = matrix(c(1, 0.5, 0.5, 1), 2, 2)
)
```

Arguments

names	vector of names
date	vector of time, must be "Date" type
mu	vector of mu
sigma	vector of sigma

Value

Multivariate stock prices

Examples

```
names <- c("swan", "bear")
date <- as.Date("2015-01-01") + days(0:29)
rMvReturnSim(names, date)
```

`rTrade` *Simulate stock trade data*

Description

Simulate stock trade data with assumption that the stock price following Geometric Brownian Motion (GBM).

Usage

```
rTrade(time, start = 100, mu = 1e-04, sigma = 2e-04)
```

Arguments

`time`	time vector of time, must be a "Date" type variable
`start`	the start position
`mu`	the `mu` parameter of GBM
`sigma`	the `sigma` parameter of GBM

Value

Stock trade data with Open, High, Low and Close

Examples

```
date <- as.Date("2015-01-01") + days(0:29)
rTrade(date)
```

`rTrades` *Simulate Multivariate Stock Trade Data*

Description

Simulate multivariate stock trade data with assumption that each stock price following Geometric Brownian Motion (GBM). And these prices are interconnected.

Usage

```
rTrades(
  name,
  time,
  start = c(1000, 1000),
  mu = rep(1e-04, 2),
  sigma = matrix(c(2e-04, 1e-04, 1e-04, 2e-04), 2, 2),
  digits = 2
)
```

Arguments

`name`	vector of names
`time`	time vector of time, must be "Date" type
`start`	vector of start positions
`mu`	vector of `mu`
`sigma`	vector of `sigma`
`digits`	integer deciding the number of deciamal places

Value

A list of stock trade data with Open, High, Low and Close

Examples

```
date <- as.Date("2015-01-01") + days(0:29)
rTrades(c("swan", "bear"), date)
```

Sharp — *Calculate Sharp Ratio with stock prices*

Description

Calculate sharp ratio of stock with running window.

Usage

```
Sharp(x, rf = 0, n = 10)
```

Arguments

`x`	vector of price
`rf`	risk free rate
`n`	the length of running window

Value

The sharp ratio series with length the same as x

Examples

```
date <- as.Date("2015-01-01") + days(0:29)
trade <- rTrade(date)
x <- trade$Close
Sharp(x)
```

StackForPlot — *Rearrange the data from* `LogErrorCosPdf` *for plot*

Description

Rearrange the data from `LogErrorCosPdf` for plot

Usage

```
StackForPlot(error, a, b, N)
```

Arguments

error	return of `LogErrorCosPdf`
a	the lower limit of the truncation interval
b	the upper limit of the truncation interval
N	the number of cos term for summation

Value

Suitable tibble data for plot by group in ggplot

Examples

```
N <- c(1:200)
L <- c(10, 20, 60, 100, 1000)
a <- -L / 2
b <- L / 2
x <- seq(-5, 5, by = 10 / (32 - 1))
el <- LogErrorCosPdf(x, dnorm, NormChf, a, b, N)
StackForPlot(el, a, b, N)
```

StackRet — *Stack Rets for ggplot*

Description

Change the arrangement of multivariate data to generate suitable data for ggplot.

Usage

```
StackRet(rets, date)
```

Arguments

`rets`	multivariate data, arranged by column
`date`	vector of common information for variables

Value

Suitable tibble data for plot by group in ggplot

Examples

```
names <- c("swan", "bear", "tiger")
date <- as.Date("2015-01-01") + days(0:179)
mu <- c(0.2, 0.08, 0.1)
sigma <- matrix(c(1, 0.25, -0.3, 0.25, 0.25, 0, -0.3, 0, 0.36), 3, 3)
allret <- rMvReturnSim(names, date, mu, sigma)
StackRet(allret, date)
```

StNormChf — *The Characteristic Function of Standard Normal Distribution*

Description

The Characteristic Function of Standard Normal Distribution

Usage

```
StNormChf(u)
```

Arguments

`u`	observation

Value

The value of Characteristic Function

Examples

```
StNormChf(1)
```

VaRSimTest *VaR Calculation and Coverage Test*

Description

Calculate VaR with three method and implement unconditional and conditional coverage test.

Usage

```
VaRSimTest(data, method, alpha, fun, ...)
```

Arguments

`data`	vector of returns
`method`	the VaR method, one of "param", "hist" and "mc"
`alpha`	the VaR confidence level
`fun`	function calculating VaR, limited by `method`
`...`	the extra parameters of `fun`

Value

A list of VaR and coverage test outcome

References

Christoffersen P. F. 1998. "Evaluating Interval Forecasts", International Economic Review, 841-862. doi: 10.2307/2527341.

Kupiec PH. 1995. "Techniques for Verifying the Accuracy of Risk Measurement Models", The Journal of Derivatives, 3(2), 73-84. doi: 10.3905/jod.1995.407942.

Examples

```
swan <- rGarch(len = 30)
date <- as.Date("2015-01-01") + days(0:(length(swan) - 1))
tswan <- tibble(garch = swan, date = date)
tsswan <- as.xts(swan, date)
alpha = 0.05
num = 100000
mu = mean(tsswan)
sd = sd(tsswan)
VaRSimTest(tsswan, "mc", alpha , rnorm, 100000, mu, sd)
```

V_k *V_k Series*

Description

Calculate the V_k Series for Option Pricing with the COS Method, an option pricing method based on the Fourier-cosine series.

Usage

```
V_k(ValueOption, N, a, b, method = "integrate")
```

Arguments

ValueOption	the value function of the option
N	the number of cos term for summation
a	the lower limit of the truncation interval
b	the upper limit of the truncation interval
method	how to calculate the integral, one of "integrate" and "jiahe"

Value

The V_k series

References

Fang F. and Oosterlee C.W. 2008. "A Novel Pricing Method for European Options Based on Fourier-Cosine Series Expansions", Siam Journal on Scientific Computing. 31(2): 826-848. doi: 10.1137/080718061.

Examples

```
r <- 0.1
sigmaS0 <- 0.2
tau <- 10
S0 <- 1
K <- 1
mu <- log(S0) + (r - 0.5 * sigmaS0^2) * tau
sigma <- sigmaS0 * sqrt(tau)
a <- -10
b <- 10
N <- 64
ValueOption <- function(x){EuroCallOption(x,K)}
V_k(ValueOption, N, a, b)
```

Index