Package ‘temStaR’

March 14, 2023

Title Tempered Stable Distribution

Version 0.90

Author Aaron Y.S. Kim [aut, cre]

Maintainer Aaron Y.S. Kim <aaron.kim@girininst.com>

Description This package provides useful tools to use the multivariate normal tempered stable distribution and process

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

Imports functional,
nloptr,
pracma,
spatstat,
Matrix,
mvtnorm,
doParallel,
foreach

Suggests functional,
nloptr,
pracma,
spatstat,
Matrix,
mvtnorm,
doParallel,
foreach

R topics documented:

`changeCovMtx2Rho` *changeCovMtx2Rho*

Description

Change covariance matrix to Rho matrix.

Usage

```
changeCovMtx2Rho(CovMtx, alpha, theta, betaVec, PDflag = TRUE)
```

chf_NTS *chf_NTS*

Description

chf_NTS calculates Ch.F of the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If a time parameter value is given, it calculates Ch.F of the NTS profess $\phi(u) = E[\exp(iu(X(t+s) - X(s)))] = \exp(t\log(E[\exp(iuX(1))]))$, where X is the NTS process generated by the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$.

Usage

```
chf_NTS(u, param)
```

Arguments

u	An array of u
ntsparam	A vector of the NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. For NTS process case it is a vector of parameters $(\alpha, \theta, \beta, \gamma, \mu, t)$.

Value

Characteristic function of the NTS distribution

Examples

```
library("temStaR")
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
u <- seq(from = -2*pi, to = 2*pi, length.out = 101)
phi <- chf_NTS(u, ntsparam)


#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
u <- seq(from = -2*pi, to = 2*pi, length.out = 101)
phi <- chf_NTS(u, ntsparam)
```

chf_stdNTS *chf_stdNTS*

Description

`chf_stdNTS` calculates Ch.F of the standard NTS distribution with parameters (α, θ, β). If a time parameter value is given, it calculates Ch.F of the standard NTS profess $\phi(u) = E[\exp(iu(X(t+s) - X(s))] = \exp(t\log(E[\exp(iuX(1))]))$, where X is the standard NTS process generated by the standard NTS distribution with parameters (α, θ, β).

Usage

```
chf_stdNTS(u, param)
```

Arguments

`u`	An array of u
`ntsparam`	A vector of the standard NTS parameters (α, θ, β). For the standard NTS process case it is a vector of parameters $(\alpha, \theta, \beta, t)$.

Value

Characteristic function of the standatd NTS distribution

Examples

```
library("temStaR")
alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
u <- seq(from = -2*pi, to = 2*pi, length.out = 101)
phi <- chf_stdNTS(u, ntsparam)


#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
u <- seq(from = -2*pi, to = 2*pi, length.out = 101)
phi <- chf_stdNTS(u, ntsparam)
```

copulaStdNTS *copulaStdNTS*

Description

`copulaStdNTS` calculates the stdNTS copula values

Usage

```
copulaStdNTS(u, st, subTS = NULL)
```

References

Y. S. Kim, D. Volkmann (2013), Normal Tempered Stable Copula, Applied Mathematics Letters, 26(7), 676-680 https://www.sciencedirect.com/science/article/pii/S0893965913000384

cvarGauss *cvarGauss*

Description

Calculate the CVaR for the normal distributed market model. Developer's version.

Usage

```
cvarGauss(eta, mu = 0, sigma = 1)
```

cvarmarginalmnts *cvarmarginalmnts*

Description

`cvarmarginalmnts` calculates the CVaR of the n-th element of the multivariate NTS distributed random variable.

Usage

```
cvarmarginalmnts(eta, n, st)
```

Arguments

`eta`	the significant level for CVaR. Real value between 0 and 1.
`n`	the n-th element to be calculated.
`st`	Structure of parameters for the n-dimensional NTS distribution.

cvarnts *cvarnts*

Description

cvarnts calculates Conditional Value at Risk (CVaR, or expected shortfall ES) of the NTS market model with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If only three parameters are given, it calculates CVaR of the standard NTS distribution with parameter (α, θ, β)

Usage

```
cvarnts(eps, ntsparam)
```

Arguments

eps	the significant level for CVaR. Real value between 0 and 1.
ntsparam	A vector of the NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. A vector of the standard NTS parameters (α, θ, β).

Value

CVaR of the NTS distribution.

References

Y. S. Kim, S. T. Rachev, M. L. Bianchi, and F. J. Fabozzi (2010), Computing VaR and AVaR in infinitely divisible distributions, Probability and Mathematical Statistics, 30 (2), 223-245.

S. T. Rachev, Y. S. Kim, M. L. Bianchi, and F. J. Fabozzi (2011), Financial Models with Levy Processes and Volatility Clustering, John Wiley & Sons

Examples

```
library("temStaR")
alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
u <- c(0.01,0.05)
q <- cvarnts(u, ntsparam)

alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
u <- c(0.01,0.05)
q <- cvarnts(u, ntsparam)


#Annual based parameters
alpha <- 1.2
theta <- 1
```

```
beta <- -0.2
gamma <- 0.3
mu <- 0.1
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
u <- c(0.01,0.05)
q <- cvarnts(u, ntsparam)
```

---

dBeta *dBeta*

Description

The first derivative of the beta. Developer's version.

Usage

```
dBeta(n, w, betaArray, covMtx)
```

---

dcopulaStdNTS *dcopulaStdNTS*

Description

`dcopulaStdNTS` calculates density of the stdNTS copula.

Usage

```
dcopulaStdNTS(u, st, subTS = NULL)
```

References

Y. S. Kim, D. Volkmann (2013), Normal Tempered Stable Copula, Applied Mathematics Letters, 26(7), 676-680 https://www.sciencedirect.com/science/article/pii/S0893965913000384

dCVaRstdNTS_numint *dCVaRstdNTS_numint*

Description

Calculate the marginal contribution to CVaR for the multivariate stdNTS Model. Developer's version.

Usage

```
dCVaRstdNTS_numint(
  eta,
  alpha,
  theta,
  beta,
  cv = NULL,
  v = NULL,
  N = 20,
  rho = 1e-04
)
```

dCVaR_numint *dCVaR_numint*

Description

The first derivative of CVaR for the beta parameter of the stdNTS. Developer's version.

Usage

```
dCVaR_numint(eta, alpha, theta, beta, N = 200, rho = 0.1)
```

dinvCdf_stdNTS_int *dinvCdf_stdNTS_int*

Description

The first derivative of inverse CDF for the beta parameter of the stdNTS. Developer's version.

Usage

```
dinvCdf_stdNTS_int(eta, x = NULL, alpha, theta, beta)
```

`dmarginalmnts` *dmarginalmnts*

Description

`dmarginalmnts` calculates the marginal density of the n-th element of the multivariate NTS distributed random variable.

Usage

```
dmarginalmnts(x, n, st)
```

Arguments

`x`	the x such that $f(x) = \frac{d}{dx}P(X_n < x)$
`n`	the n-th element to be calculated.
`st`	Structure of parameters for the n-dimensional NTS distribution.

`dmnts` *dmnts*

Description

`dmnts` calculates the density of the multivariate NTS distribution: $f(x_1, \cdots, x_n) = \frac{d^n}{dx_1 \cdots dx_n} P(x_n < R_1, \cdots, x_n < R_n)$. The multivariate NTS random vector $R = (R_1, \cdots, R_n)$ is defined

$R = \mu + diag(\sigma)X$,

where

X follows $stdNTS_n(\alpha, \theta, \beta, \Sigma)$

Usage

```
dmnts(x, st, subTS = NULL)
```

Arguments

`x`	array of the $(x_1, \cdots, x_n)$
`st`	Structure of parameters for the n-dimensional NTS distribution. `st$ndim` : dimension `st$mu` : μ mean vector (column vector) of the input data. `st$sigma` : σ standard deviation vector (column vector) of the input data. `st$alpha` : α of the std NTS distribution (X). `st$theta` : θ of the std NTS distribution (X). `st$beta` : β vector (column vector) of the std NTS distribution (X). `st$Rho` : ρ matrix of the std NTS distribution (X).
`numofsample`	number of samples.

Value

Simulated NTS random vectors

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library("temStaR")
library(mvtnorm)
strPMNTS <- list(ndim = 2,
              mu = c( 0.5, -1.5 ),
              sigma = c( 2, 3 ),
              alpha = 0.1,
              theta = 3,
              beta =  c( 0.1, -0.3 ),
              Rho = matrix( data = c(1.0, 0.75, 0.75, 1.0),
                            nrow = 2, ncol = 2)
)
dmnts(c(0.6, -1.0), st = strPMNTS)


strPMNTS <- list(ndim = 2,
                 mu = c( 0, 0, 0 ),
                 sigma = c( 1, 1, 1 ),
                 alpha = 0.1,
                 theta = 3,
                 beta =  c( 0.1, -0.3, 0 ),
                 Rho = matrix(
                     data = c(1.0, 0.75, 0.1, 0.75, 1.0, 0.2, 0.1, 0.2, 1.0),
                     nrow = 3, ncol = 3)
)
pmnts(c(0,0,0), st = strPMNTS)
dmnts(c(0,0,0), st = strPMNTS)
```

---

dnts *dnts*

Description

dnts calculates pdf of the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If only three parameters are given, it calculates pdf of the standard NTS distribution with parameter (α, θ, β) If a time parameter value is given, it calculates pdf of the NTS profess $f(x)dx = d(P((X(t+s) - X(s)) < x))$, where X is the NTS process generated by the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$.

Usage

```
dnts(xdata, ntsparam)
```

Arguments

xdata An array of x

ntsparam A vector of the NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. For the NTS process case it is a vector of parameters $(\alpha, \theta, \beta, \gamma, \mu, t)$. A vector of the standard NTS parameters (α, θ, β).

Value

Density of NTS distribution

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library("temStaR")

alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
x <- seq(from = -6, to = 6, length.out = 101)
d <- dnts(x, ntsparam)
plot(x,d,type = 'l')

alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
x <- seq(from = -2, to = 2, by = 0.01)
d <- dnts(x, ntsparam)
plot(x,d,type = 'l')


#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
x <- seq(from = -0.02, to = 0.02, length.out = 101)
d <- dnts(x, ntsparam)
plot(x,d,type = 'l')
```

fitmnts *fitmnts*

Description

`fitmnts` fit parameters of the n-dimensional NTS distribution.

$r = \mu + diag(\sigma)X$

where

X follows $stdNTS_n(\alpha, \theta, \beta, \Sigma)$

Usage

```
\code{res <- fitmnts(returndata, n)}
\code{res <- fitmnts(returndata, n, alphaNtheta = c(alpha, theta))}
\code{res <- fitmnts(returndata, n, stdflag = TRUE ) }
\code{res <- fitmnts(returndata, n, alphaNtheta = c(alpha, theta), stdflag = TRUE)}
```

Arguments

returndata	Raw data to fit the parameters. The data must be given as a matrix form. Each column of the matrix contains a sequence of asset returns. The number of row of the matrix is the number of assets.
n	Dimension of the data. That is the number of assets.
alphaNtheta	If α and θ are given, then put those numbers in this parameter. The function fixes those parameters and fits other remaining parameters. If you set `alphaNtheta = NULL`, then the function fits all parameters including α and θ.
stdflag	If you want only standard NTS parameter fit, set this value be TRUE.

Value

Structure of parameters for the n-dimensional NTS distribution.

`res$mu` : μ mean vector of the input data.

`res$sigma` : σ standard deviation vector of the input data.

`res$alpha` : α of the std NTS distribution (X).

`res$theta` : θ of the std NTS distribution (X).

`res$beta` : β vector of the std NTS distribution (X).

`res$Rho` : ρ matrix of the std NTS distribution (X), which is correlation matrix of epsilon.

`res$CovMtx` : Covariance matrix of return data r.

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library(functional)
library(nloptr)
library(pracma)
library(spatstat)
library(Matrix)
library(quantmod)
library(temStaR)

getSymbols("^GSPC", src="yahoo", from = "2016-1-1", to = "2020-08-31")
pr1 <- as.numeric(GSPC$GSPC.Adjusted)
getSymbols("^DJI", src="yahoo", from = "2016-1-1", to = "2020-08-31")
pr2 <- as.numeric(DJI$DJI.Adjusted)

returndata <- matrix(data = c(diff(log(pr1)),diff(log(pr2))),
                     ncol = 2, nrow = (length(pr1)-1))
res <- fitmnts( returndata = returndata, n=2 )


#Fix alpha and theta.
#Estimate alpha dna theta from DJIA and use those parameter for IBM, INTC parameter fit.
getSymbols("^DJI", src="yahoo", from = "2016-1-1", to = "2020-08-31")
prDJ <- as.numeric(DJI$DJI.Adjusted)
ret <- diff(log(prDJ))
ntsparam <-  fitnts(ret)
getSymbols("IBM", src="yahoo", from = "2016-1-1", to = "2020-08-31")
pr1 <- as.numeric(IBM$IBM.Adjusted)
getSymbols("INTC", src="yahoo", from = "2016-1-1", to = "2020-08-31")
pr2 <- as.numeric(INTC$INTC.Adjusted)

returndata <- matrix(data = c(diff(log(pr1)),diff(log(pr2))),
                     ncol = 2, nrow = (length(pr1)-1))
res <- fitmnts( returndata = returndata,
                n = 2,
                alphaNtheta = c(ntsparam["alpha"], ntsparam["theta"])  )
```

`fitmnts_par` *fitmnts*

Description

`fitmnts` fit parameters of the n-dimensional NTS distribution. A parallel version of fitmnts()

Usage

```
fitmnts_par(
  returndata,
  n,
  alphaNtheta = NULL,
  stdflag = FALSE,
  parallelSocketCluster = NULL,
  PDflag = TRUE
)
```

Examples

```
library(functional)
library(nloptr)
library(pracma)
library(spatstat)
library(Matrix)
library(foreach)
library(doParallel)
library(quantmod)
library(temStaR)

getSymbols("^GSPC", src="yahoo", from = "2016-1-1", to = "2020-08-31")
pr1 <- as.numeric(GSPC$GSPC.Adjusted)
getSymbols("^DJI", src="yahoo", from = "2016-1-1", to = "2020-08-31")
pr2 <- as.numeric(DJI$DJI.Adjusted)

returndata <- matrix(data = c(diff(log(pr1)),diff(log(pr2))),
                     ncol = 2, nrow = (length(pr1)-1))

numofcluster  <- detectCores()
cl <- makePSOCKcluster(numofcluster)
registerDoParallel(cl)
res <- fitmnts_par( returndata = returndata, n=2, parallelSocketCluster = cl )
stopCluster(cl)
```

`fitnts` *fitnts*

Description

`fitnts` fit parameters $(\alpha, \theta, \beta, \gamma, \mu)$ of the NTS distribution. This function using the curvefit method between the empirical cdf and the NTS cdf.

Usage

```
\code{fitnts(rawdat)}
\code{fitnts(rawdat), ksdensityflag = 1}
\code{fitnts(rawdat, initialparam = c(alpha, theta, beta, gamma, mu))}
\code{fitnts(rawdat, initialparam = c(alpha, theta, beta, gamma, mu)), ksdensityflag = 1}
\code{fitnts(rawdat, initialparam = c(alpha, theta, beta, gamma, mu)), maxeval = 100, ksdensityflag
```

Arguments

`rawdat`	Raw data to fit the parameters.
`initialparam`	A vector of initial NTS parameters. This function uses the `nloptr` package. If it has a good initial parameter then estimation performs better. If users do not know a good initial parameters, then just set it as `initialparam=NaN`, that is default. The function `cffitnts()` may be helpful to find the initial parameters.
`maxeval`	Maximum evaluation number for `nloptr`. The iteration stops on this many function evaluations.

`ksdensityflag` This function fit the parameters using the curvefit method between the empirical cdf and the NTS cdf. If `ksdensityflag = 1` (default), then the empirical cdf is calculated by the kernel density estimation. If `ksdensityflag = 0`, then the empirical cdf is calculated by the empirical cdf.

Value

Estimated parameters

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library("temStaR")
library("quantmod")
getSymbols("^GSPC", src="yahoo", from = "2010-1-1", to = "2020-12-31")
pr <- as.numeric(GSPC$GSPC.Adjusted)
ret <- diff(log(pr))
ntsparam <-  fitnts(ret)

Femp = ecdf(ret)
x = seq(from=min(ret), to = max(ret), length.out = 100)
cemp = Femp(x)
ncdf = pnts(x, c(ntsparam))
plot(x,ncdf,type = 'l', col = "red")
points(x,cemp, type = 'l', col = "blue")
a = density(ret)
p = dnts(x,ntsparam)
plot(x,p,type = 'l', col = "red")
lines(a,type = 'l', col = "blue")
```

`fitstdnts` *fitstdnts*

Description

`fitstdnts` fit parameters (α, θ, β) of the standard NTS distribution. This function using the curvefit method between the empirical cdf and the standard NTS cdf.

Usage

```
\code{fitstdnts(rawdat)}
\code{fitstdnts(rawdat), ksdensityflag = 1}
\code{fitstdnts(rawdat, initialparam = c(alpha, theta, beta))}
\code{fitstdnts(rawdat, initialparam = c(alpha, theta, beta)), ksdensityflag = 1}
\code{fitstdnts(rawdat, initialparam = c(alpha, theta, beta)), maxeval = 100, ksdensityflag = 1}
```

Arguments

`rawdat`	Raw data to fit the parameters.
`initialparam`	A vector of initial standard NTS parameters. This function uses the `nloptr` package. If it has a good initial parameter then estimation performs better. If users do not know a good initial parameters, then just set it as `initialparam=NaN`, that is default.
`maxeval`	Maximum evaluation number for `nloptr`. The iteration stops on this many function evaluations.
`ksdensityflag`	This function fit the parameters using the curvefit method between the empirical cdf and the standard NTS cdf. If `ksdensityflag = 1` (default), then the empirical cdf is calculated by the kernel density estimation. If `ksdensityflag = 0`, then the empirical cdf is calculated by the empirical cdf.

Value

Estimated parameters

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library("temStaR")
library("quantmod")
getSymbols("^GSPC", src="yahoo", from = "2010-1-1", to = "2020-12-31")
pr <- as.numeric(GSPC$GSPC.Adjusted)
ret <- diff(log(pr))
stdret <- (ret-mean(ret))/sd(ret)
stdntsparam <-  fitstdnts(stdret)

Femp = ecdf(stdret)
x = seq(from=min(stdret), to = max(stdret), length.out = 100)
cemp = Femp(x)
ncdf = pnts(x, c(stdntsparam))
plot(x,ncdf,type = 'l', col = "red")
lines(x,cemp, type = 'l', col = "blue")
a = density(stdret)
p = dnts(x,stdntsparam)
plot(x,p,type = 'l', col = "red", ylim = c(0, max(a$y, p)))
lines(a,type = 'l', col = "blue")
```

`fitstdntsFixAlphaThata` *fitstdntsFixAlphaThata*

Description

Fit beta of stdNTS distribution with fixed alpha and theta.

Usage

```
fitstdntsFixAlphaThata(
  rawdat,
  alpha,
  theta,
  initialparam = NaN,
  maxeval = 100,
  ksdensityflag = 1
)
```

---

gensamplepathnts *gensamplepathnts*

Description

gensamplepathnts generate sample paths of the NTS process with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If only three parameters are given, it generate sample paths of the standard NTS process with parameters (α, θ, β).

Usage

```
gensamplepathnts(npath, ntimestep, ntsparam, dt)
```

Arguments

npath	Number of sample paths
ntimestep	number of time step
ntsparam	A vector of the NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. A vector of the standard NTS parameters (α, θ, β).
dt	the time length of one time step by the year fraction. "dt=1" means 1-year.

Value

Structure of the sample path. Matrix of sample path. Column index is time.

Examples

```
library("temStaR")
#standard NTS process sample path
alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
npath <- 5
ntimestep <- 250
dt <- 1/250
simulation <- gensamplepathnts(npath, ntimestep, ntsparam, dt)
matplot(colnames(simulation), t(simulation), type = 'l')

#NTS process sample path
alpha <- 1.2
```

```
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
npath <- 5
ntimestep <- 250
dt <- 1/250
simulation <- gensamplepathnts(npath, ntimestep, ntsparam, dt)
matplot(colnames(simulation), t(simulation), type = 'l')
```

`getGammaVec` *getGammaVec*

Description

beta to gamma in StdNTS

Usage

```
getGammaVec(alpha, theta, betaVec)
```

`getPortNTSParam` *getPortNTSParam*

Description

Portfolio return with capital allocation weight is $R_p =< w, r >$, which is a weighted sum of of elements in the N-dimensional NTS random vector. R_p becomes an 1-dimensional NTS random variable. `getPortNTSParam` find the parameters of R_p.

Usage

```
\code{res <- setPortfolioParam(strPMNTS,w)}
\code{res <- setPortfolioParam(strPMNTS,w, FALSE)}
```

Arguments

`strPMNTS` Structure of parameters for the n-dimensional NTS distribution.

`strPMNTS$ndim` : dimension

`strPMNTS$mu` : μ mean vector (column vector) of the input data.

`strPMNTS$sigma` : σ standard deviation vector (column vector) of the input data.

`strPMNTS$alpha` : α of the std NTS distribution (X).

`strPMNTS$theta` : θ of the std NTS distribution (X).

`strPMNTS$beta` : β vector (column vector) of the std NTS distribution (X).

`res$Rho` : ρ matrix (Correlation) of the std NTS distribution (X).

`res$Sigma` : Covariance Σ matrix of return data r.

`w` Capital allocation weight vector.

`stdform` If `stdform` is `FALSE`, then the return parameter has the following representation
$R_p =< w, r >= \mu + diag(\sigma)X$,
where
X follows $stdNTS_1(\alpha, \theta, \beta, 1)$.
If `stdform` is `TRUE`, then the return parameter has the following representation
$R_p =< w, r >$ follows $NTS_1(\alpha, \theta, \beta, \gamma, \mu, 1)$

Value

The weighted sum follows 1-dimensional NTS.

$R_p =< w, r >= \mu + diag(\sigma)X$,

where

X follows $stdNTS_1(\alpha, \theta, \beta, 1)$.

Hence we obtain

`res$mu` : μ mean of R_p.

`res$sigma` : σ standard deviation of R_p.

`res$alpha` : α of X.

`res$theta` : θ of X.

`res$beta` : β of X.

References

Proposition 2.1 of Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library("temStaR")
strPMNTS <- list(ndim = 2,
                 mu = c( 9.876552e-05, 4.747343e-04 ),
                 sigma = c( 0.01620588, 0.02309643 ),
                 alpha = 0.1888129 ,
                 theta = 0.523042,
                 beta =  c( -0.04632938,  0.04063555 ),
                 Rho = matrix( data = c(1.0, 0.469883,
                                        0.469883, 1.0),
                               nrow = 2, ncol = 2)
                 CovMtx = matrix( data = c(0.0002626304, 0.0001740779,
                                          0.0001740779, 0.0005334452),
                                 nrow = 2, ncol = 2)
                 )
w <- c(0.3, 0.7)
res <- getPortNTSParam(strPMNTS,w)
```

`importantSamplining` *importantSamplining*

Description

`importantSamplining` do the important sampling for the TS Subordinator.

Usage

```
importantSamplining(alpha, theta)
```

`ipnts` *ipnts*

Description

`ipnts` calculates inverse cdf of the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If only three parameters are given, it calculates inverse cdf of the standard NTS distribution with parameter (α, θ, β)

Usage

```
ipnts(u, ntsparam, maxmin = c(-10, 10), du = 0.01)
```

Arguments

`u`	Real value between 0 and 1
`ntsparam`	A vector of the NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. A vector of the standard NTS parameters (α, θ, β).

Value

Inverse cdf of the NTS distribution. It is the same as qnts function.

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library("temStaR")
alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
u <- seq(from = 0.01, to = 0.99, length.out = 99)
q <- ipnts(u, ntsparam)
plot(u,q,type = 'l')

alpha <- 1.2
```

```
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
u <- seq(from = 0.01, to = 0.99, length.out = 99)
q <- ipnts(u, ntsparam)
plot(x,q,type = 'l')


#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
u <- seq(from = 0.01, to = 0.99, length.out = 99)
q <- ipnts(u, ntsparam)
plot(x,q,type = 'l')
```

`mctCVaR_MNTS` *mctCVaR_MNTS*

Description

Calculate the marginal contribution to CVaR for the multivariate NTS market model: the random vector r is

$r = \mu + diag(\sigma)X$

where

X follows $stdNTS_N(\alpha, \theta, \beta, \Sigma)$

Usage

```
\code{mctCVaR_MNTS(eta, n, w, stmnts)}
```

Arguments

`n`	The targer stock to calculate the mctCVaR
`eta`	Significant level of CVaR.
`w`	The capital allocation rate vector for the current portfolio
`stmnts`	Structure of parameters for the N-dimensional NTS distribution.
`CVaRstd`	CVaR Value of StdNTS residual. If NULL, the function automatically find it. The default value is NULL
`dCVaRstd`	The first derivative of the stdNTS CVaR for beta. If NULL, the function automatically find it. The default value is NULL

iCDFstd The inverst cdf of stdNTS at the significant level eta. If NULL, the function automatically find it. The default value is NULL

`st$ndim` : Dimension of the model. Here `st$ndim=N`.

`st$mu` : μ mean vector (column vector) of the input data.

`st$sigma` : σ standard deviation vector (column vector) of the input data.

`st$alpha` : α of the std NTS distribution (X).

`st$theta` : θ of the std NTS distribution (X).

`st$beta` : β vector (column vector) of the std NTS distribution (X).

`st$Rho` : ρ matrix of the std NTS distribution (X), which is correlation matrix of epsilon.

`st$CovMtx` : Covariance matrix of return data r.

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library(functional)
library(nloptr)
library(pracma)
library(spatstat)
library(Matrix)
library(quantmod)
library(mvtnorm)
library("temStaR")

#Fix alpha and theta.
#Estimate alpha dna theta from DJIA and use those parameter for IBM, INTL parameter fit.
getSymbols("^DJI", src="yahoo", from = "2020-8-25", to = "2020-08-31")
prDJ <- as.numeric(DJI$DJI.Adjusted)
ret <- diff(log(prDJ))
ntsparam <-  fitnts(ret)
getSymbols("IBM", src="yahoo", from = "2016-1-1", to = "2020-08-31")
pr1 <- as.numeric(IBM$IBM.Adjusted)
getSymbols("INTL", src="yahoo", from = "2016-1-1", to = "2020-08-31")
pr2 <- as.numeric(INTL$INTL.Adjusted)

returndata <- matrix(data = c(diff(log(pr1)),diff(log(pr2))),
                     ncol = 2, nrow = (length(pr1)-1))
st <- fitmnts( returndata = returndata,
                n = 2,
                alphaNtheta = c(ntsparam["alpha"], ntsparam["theta"])  )
w <- c(0.3, 0.7)
eta <- 0.01

mctCVaR_MNTS(1, eta, w, st) #MCT-CVaR for IBM
mctCVaR_MNTS(2, eta, w, st) #MCT-CVaR for INTL
```

mctStdDev *mctStdDev*

Description

Morginal contribution to Risk for Standard Deviation.

Usage

```
mctStdDev(n, w, covMtx)
```

Arguments

n	The targer stock to calculate the mctCVaR
w	The capital allocation rate vector for the current portfolio
CovMtx	Covariance matrix of return data.

mctVaR_MNTS *mctVaR_MNTS*

Description

Calculate the marginal contribution to VaR for the multivariate NTS market model: the random vector r is

$r = \mu + diag(\sigma)X$

where

X follows $stdNTS_N(\alpha, \theta, \beta, \Sigma)$

Usage

```
\code{mctVaR_MNTS(n, eta, w, st)}
```

Arguments

n	The target stock to calculate the mctVaR
eta	Significant level of VaR.
w	The capital allocation rate vector for the current portfolio
iCDFstd	The inverst cdf of stdNTS at the significant level eta. If NULL, the function automatically find it. The default value is NULL `st$ndim` : Dimension of the model. Here `st$ndim=N`. `st$mu` : μ mean vector (column vector) of the input data. `st$sigma` : σ standard deviation vector (column vector) of the input data. `st$alpha` : α of the std NTS distribution (X). `st$theta` : θ of the std NTS distribution (X). `st$beta` : β vector (column vector) of the std NTS distribution (X). `st$Rho` : ρ matrix of the std NTS distribution (X), which is correlation matrix of epsilon. `st$CovMtx` : Covariance matrix of return data r.
st	Structure of parameters for the N-dimensional NTS distribution.

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library(functional)
library(nloptr)
library(pracma)
library(spatstat)
library(Matrix)
library(quantmod)
library(mvtnorm)
library(temStaR)

#Fix alpha and theta.
#Estimate alpha dna theta from DJIA and use those parameter for IBM, INTL parameter fit.
getSymbols("^DJI", src="yahoo", from = "2020-8-25", to = "2020-08-31")
prDJ <- as.numeric(DJI$DJI.Adjusted)
ret <- diff(log(prDJ))
ntsparam <-  fitnts(ret)
getSymbols("IBM", src="yahoo", from = "2016-1-1", to = "2020-08-31")
pr1 <- as.numeric(IBM$IBM.Adjusted)
getSymbols("INTL", src="yahoo", from = "2016-1-1", to = "2020-08-31")
pr2 <- as.numeric(INTL$INTL.Adjusted)

returndata <- matrix(data = c(diff(log(pr1)),diff(log(pr2))),
                     ncol = 2, nrow = (length(pr1)-1))
st <- fitmnts( returndata = returndata,
               n = 2,
               alphaNtheta = c(ntsparam["alpha"], ntsparam["theta"])  )
w <- c(0.3, 0.7)
eta <- 0.01

mctVaR_MNTS(1, eta, w, st) #MCT-VaR for IBM
mctVaR_MNTS(2, eta, w, st) #MCT-VaR for INTL
```

---

moments_NTS *moments_NTS*

Description

moments_NTS calculates mean, variance, skewness, and excess kurtosis of the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$.

Usage

```
moments_NTS(param)
```

Arguments

param	A vector of the NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$.

Value

First 4 moments (Mean, Variance, Skewness, Excess Kurtosis) of NTS distribution. The mean is always the same as the parameter μ.

References

Kim, Y.S, K-H Roh, R. Douady (2020) Tempered Stable Processes with Time Varying Exponential Tails https://arxiv.org/pdf/2006.07669.pdf

Examples

```
library("temStaR")
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
moments_NTS(param = ntsparam)
```

---

moments_stdNTS *moments_stdNTS*

Description

moments_stdNTS calculates mean, variance, skewness, and excess kurtosis of the standard NTS distribution with parameters (α, θ, β).

Usage

```
moments_stdNTS(param)
```

Arguments

param	A vector of the standard NTS parameters (α, θ, β).

Value

First 4 moments (Mean, Variance, Skewness, Excess Kurtosis) of NTS distribution. Of course, the mean and variance are always 0 and 1, respectively.

References

Kim, Y.S, K-H Roh, R. Douady (2020) Tempered Stable Processes with Time Varying Exponential Tails https://arxiv.org/pdf/2006.07669.pdf

Examples

```
library("temStaR")
alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
moments_stdNTS(param = ntsparam)
```

`pmarginalmnts` *pmarginalmnts*

Description

`pmarginalmnts` calculates the marginal cdf of the n-th element of the multivariate NTS distributed random variable.

Usage

```
pmarginalmnts(x, n, st)
```

Arguments

`x`	the x such that $F(x) = P(X_n < x)$
`n`	the n-th element to be calculated.
`st`	Structure of parameters for the n-dimensional NTS distribution.

`pmnts` *pmnts*

Description

`pmnts` calculates the cdf values of the multivariate NTS distribution: $F(x_1, \cdots, x_n) = P(x_n < R_1, \cdots, x_n < R_n)$. The multivariate NTS random vector $R = (R_1, \cdots, R_n)$ is defined

$R = \mu + diag(\sigma)X$,

where

X follows $stdNTS_n(\alpha, \theta, \beta, \Sigma)$

Usage

```
pmnts(x, st, subTS = NULL)
```

Arguments

`x`	array of the $(x_1, \cdots, x_n)$
`st`	Structure of parameters for the n-dimensional NTS distribution. `st$ndim` : dimension `st$mu` : μ mean vector (column vector) of the input data. `st$sigma` : σ standard deviation vector (column vector) of the input data. `st$alpha` : α of the std NTS distribution (X). `st$theta` : θ of the std NTS distribution (X). `st$beta` : β vector (column vector) of the std NTS distribution (X). `st$Rho` : ρ matrix of the std NTS distribution (X).
`numofsample`	number of samples.

Value

Simulated NTS random vectors

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library(mvtnorm)
library(temStaR)

strPMNTS <- list(ndim = 2,
              mu = c( 0.5, -1.5 ),
              sigma = c( 2, 3 ),
              alpha = 0.1,
              theta = 3,
              beta =  c( 0.1, -0.3 ),
              Rho = matrix( data = c(1.0, 0.75, 0.75, 1.0),
                            nrow = 2, ncol = 2)
)
pmnts(c(0.6, -1.0), st = strPMNTS)


strPMNTS <- list(ndim = 2,
                 mu = c( 0, 0, 0 ),
                 sigma = c( 1, 1, 1 ),
                 alpha = 0.1,
                 theta = 3,
                 beta =  c( 0.1, -0.3, 0 ),
                 Rho = matrix(
                     data = c(1.0, 0.75, 0.1, 0.75, 1.0, 0.2, 0.1, 0.2, 1.0),
                     nrow = 3, ncol = 3)
)
pmnts(c(0,0,0), st = strPMNTS)
dmnts(c(0,0,0), st = strPMNTS)
```

`pnts` *pnts*

Description

`pnts` calculates cdf of the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If only three parameters are given, it calculates cdf of the standard NTS distribution with parameter (α, θ, β) If a time parameter value is given, it calculates cdf of the profess $F(x) = P((X(t+s) - X(s)) < x)$, where X is the NTS process generated by the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$.

Usage

```
pnts(xdata, ntsparam, dz = 2^-8, m = 2^12)
```

Arguments

xdata	An array of x
ntsparam	A vector of the NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. For the NTS process case it is a vector of parameters $(\alpha, \theta, \beta, \gamma, \mu, t)$. A vector of the standard NTS parameters (α, θ, β).

Value

Cumulative probability of the NTS distribution

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library("temStaR")

alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
x <- seq(from = -6, to = 6, length.out = 101)
p <- pnts(x, ntsparam)
plot(x,p,type = 'l')

alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
x <- seq(from = -2, to = 2, by = 0.01)
p <- pnts(x, ntsparam)
plot(x,p,type = 'l')


#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
x <- seq(from = -0.02, to = 0.02, length.out = 101)
p <- pnts(x, ntsparam)
plot(x,p,type = 'l')
```

`portfolioCVaRmnts` *portfolioCVaRmnts*

Description

Calculate portfolio conditional value at risk (expected shortfall) on the NTS market model

Usage

```
portfolioCVaRmnts(strPMNTS, w, eta)
```

Arguments

`strPMNTS`	Structure of parameters for the n-dimensional NTS distribution. `strPMNTS$ndim` : dimension `strPMNTS$mu` : μ mean vector (column vector) of the input data. `strPMNTS$sigma` : σ standard deviation vector (column vector) of the input data. `strPMNTS$alpha` : α of the std NTS distribution (X). `strPMNTS$theta` : θ of the std NTS distribution (X). `strPMNTS$beta` : β vector (column vector) of the std NTS distribution (X). `res$Rho` : ρ matrix (Correlation) of the std NTS distribution (X). `res$Sigma` : Covariance Σ matrix of return data r.
`w`	Capital allocation weight vector.
`eta`	significanlt level

Value

portfolio value at risk on the NTS market model

`portfolioVaRmnts` *portfolioVaRmnts*

Description

Calculate portfolio value at risk on the NTS market model

Usage

```
portfolioVaRmnts(strPMNTS, w, eta)
```

Arguments

`strPMNTS`	Structure of parameters for the n-dimensional NTS distribution. `strPMNTS$ndim` : dimension `strPMNTS$mu` : μ mean vector (column vector) of the input data. `strPMNTS$sigma` : σ standard deviation vector (column vector) of the input data. `strPMNTS$alpha` : α of the std NTS distribution (X). `strPMNTS$theta` : θ of the std NTS distribution (X). `strPMNTS$beta` : β vector (column vector) of the std NTS distribution (X). `res$Rho` : ρ matrix (Correlation) of the std NTS distribution (X). `res$Sigma` : Covariance Σ matrix of return data r.
`w`	Capital allocation weight vector.
`eta`	significanlt level

Value

portfolio value at risk on the NTS market model

`qmarginalmnts` *qmarginalmnts*

Description

`qmarginalmnts` calculates the quantile value of the n-th element of the multivariate NTS distributed random variable.

Usage

```
qmarginalmnts(u, n, st)
```

Arguments

`u`	vector of probabilities.
`n`	the n-th element to be calculated.
`st`	Structure of parameters for the n-dimensional NTS distribution.

`qnts` *qnts*

Description

`qnts` calculates quantile of the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If only three parameters are given, it calculates quantile of the standard NTS distribution with parameter (α, θ, β) If a time parameter value is given, it calculates quantile of NTS profess. That is it finds x such that $u = P((X(t + s) - X(s)) < x)$, where X is the NTS process generated by the NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$.

Usage

```
qnts(u, ntsparam)
```

Arguments

`u`	vector of probabilities.
`ntsparam`	A vector of the NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. For the NTS process case it is a vector of parameters $(\alpha, \theta, \beta, \gamma, \mu, t)$. A vector of standard NTS parameters (α, θ, β).

Value

The quantile function of the NTS distribution

Examples

```
library("temStaR")

alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
u <- c(0.01,0.05,0.25,0.5, 0.75, 0.95, 0.99)
q <- qnts(u, ntsparam)

alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
u <- c(0.01,0.05,0.25,0.5, 0.75, 0.95, 0.99)
q <- qnts(u, ntsparam)


#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
```

```
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
u <- c(0.01,0.05,0.25,0.5, 0.75, 0.95, 0.99)
q <- qnts(u, ntsparam)
```

---

rmnts *rmnts*

Description

`rmnts` generates random vector following the n dimensional NTS distribution using subordination.

$r = \mu + diag(\sigma)X,$

where

X follows $stdNTS_n(\alpha, \theta, \beta, \Sigma)$

Usage

```
rmnts(strPMNTS, numofsample, rW = NULL, rTau = NULL)
```

Arguments

`strPMNTS`	Structure of parameters for the n-dimensional NTS distribution. `strPMNTS$ndim` : dimension `strPMNTS$mu` : μ mean vector (column vector) of the input data. `strPMNTS$sigma` : σ standard deviation vector (column vector) of the input data. `strPMNTS$alpha` : α of the std NTS distribution (X). `strPMNTS$theta` : θ of the std NTS distribution (X). `strPMNTS$beta` : β vector (column vector) of the std NTS distribution (X). `strPMNTS$Rho` : ρ matrix of the std NTS distribution (X).
`numofsample`	number of samples.

Value

Simulated NTS random vectors

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
strPMNTS <- list(ndim = 2,
                 mu = c( 0.00011, 0.00048 ),
                 sigma = c( 0.0162, 0.0231 ),
                 alpha = 1.23,
                 theta = 3.607,
                 beta =  c( -0.1209,  0.0905 ),
                 Rho = matrix( data = c(1.0, 0.55, 0.55, 1.0), nrow = 2, ncol = 2)
```

```
)
gensim <- rmnts( strPMNTS, 100 )
plot(gensim)
```

`rnts` *rnts*

Description

`rnts` generates random numbers following NTS distribution with parameters $(\alpha, \theta, \beta, \gamma, \mu)$. If only three parameters are given, it generates random numbers of standard NTS distribution with parameter (α, θ, β) If a time parameter value is given, it generates random numbers of increments of NTS profess for time interval t.

Usage

```
rnts(n, ntsparam, u = NULL)
```

Arguments

`n`	number of random numbers to be generated.
`ntsparam`	A vector of NTS parameters $(\alpha, \theta, \beta, \gamma, \mu)$. For NTS process case it is a vector of parameters $(\alpha, \theta, \beta, \gamma, \mu, t)$. A vector of standard NTS parameters (α, θ, β).

Value

NTS random numbers

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library("temStaR")

alpha <- 1.2
theta <- 1
beta <- -0.2
ntsparam <- c(alpha, theta, beta)
r <- rnts(100, ntsparam) #generate 100 NTS random numbers
plot(r)

alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
ntsparam <- c(alpha, theta, beta, gamma, mu)
r <- rnts(100, ntsparam) #generate 100 NTS random numbers
plot(r)
```

```
#Annual based parameters
alpha <- 1.2
theta <- 1
beta <- -0.2
gamma <- 0.3
mu <- 0.1
#scaling annual parameters to one day
dt <- 1/250 #one day
ntsparam <- c(alpha, theta, beta, gamma, mu, dt)
r <- rnts(100, ntsparam) #generate 100 NTS random numbers
plot(r)
```

`setPortfolioParam` *setPortfolioParam*

Description

Please use `getPortNTSParam` instead of `setPortfolioParam`.

Portfolio return with capital allocation weight is $R_p =< w, r >$, which is a weighted sum of of elements in the N-dimensional NTS random vector. R_p becomes an 1-dimensional NTS random variable. `setPortfolioParam` find the parameters of R_p.

Usage

```
\code{res <- setPortfolioParam(strPMNTS,w)}
```

Arguments

`strPMNTS`	Structure of parameters for the n-dimensional NTS distribution. `strPMNTS$ndim` : dimension `strPMNTS$mu` : μ mean vector (column vector) of the input data. `strPMNTS$sigma` : σ standard deviation vector (column vector) of the input data. `strPMNTS$alpha` : α of the std NTS distribution (X). `strPMNTS$theta` : θ of the std NTS distribution (X). `strPMNTS$beta` : β vector (column vector) of the std NTS distribution (X). `strPMNTS$Rho` : Σ matrix of the std NTS distribution (X).
`w`	Capital allocation weight vector.

Value

The weighted sum follows 1-dimensional NTS.

$R_p =< w, r >= \mu + diag(\sigma)X$,

where

X follows $stdNTS_1(\alpha, \theta, \beta, 1)$.

Hence we obtain

`res$mu` : μ mean of R_p.

`res$sigma` : σ standard deviation of R_p.

`res$alpha` : α of X.

`res$theta` : θ of X.

`res$beta` : β X.

References

Kim, Y. S. (2020) Portfolio Optimization on the Dispersion Risk and the Asymmetric Tail Risk https://arxiv.org/pdf/2007.13972.pdf

Examples

```
library("temStaR")
strPMNTS <- list(ndim = 2,
                 mu = c( 9.876552e-05, 4.747343e-04 ),
                 sigma = c( 0.01620588, 0.02309643 ),
                 alpha = 0.1888129 ,
                 theta = 0.523042,
                 beta =  c( -0.04632938,  0.04063555 ),
                 Rho = matrix( data = c(1.0, 0.469883,
                                        0.469883, 1.0),
                               nrow = 2, ncol = 2)
                 CovMtx = matrix( data = c(0.0002626304, 0.0001740779,
                                           0.0001740779, 0.0005334452),
                                  nrow = 2, ncol = 2)
)
w <- c(0.3, 0.7)
res <- setPortfolioParam(strPMNTS,w)
```

Index