Package 'TempStable'

January 20, 2025

Type Package

Title A Collection of Methods to Estimate Parameters of Different Tempered Stable Distributions

Version 0.2.2

Description A collection of methods to estimate parameters of different tempered stable distributions (TSD). Currently, there are seven different tempered stable distributions to choose from: Tempered stable subordinator distribution, classical TSD, generalized classical TSD, normal TSD, modified TSD, rapid decreasing TSD, and Kim-Rachev TSD.
The package also provides functions to compute density and probability functions and tools to run Monte Carlo simulations.
This package has already been used for the estimation of tempered stable distributions (Massing (2023) <arXiv:2303.07060>).
The following references form the theoretical background for various functions in this package. References for each function are explicitly listed in its documentation:
Bianchi et al. (2010) <doi:10.1007/978-88-470-1481-7_4>
Bianchi et al. (2011) <doi:10.1137/S0040585X97984632>
Carrasco (2017) <doi:10.1017/S0266466616000025>
Feuerverger (1981) <doi:10.1111/j.2517-6161.1981.tb01143.x>
Hansen et al. (1996) <doi:10.1080/07350015.1996.10524656>
Hansen (1982) <doi:10.2307/1912775>
Hofert (2011) <doi:10.1145/2043635.2043638>
Kawai & Masuda (2011) <doi:10.1016/j.cam.2010.12.014>
Kim et al. (2008) <doi:10.1016/j.jbankfin.2007.11.004>
Kim et al. (2009) <doi:10.1007/978-3-7908-2050-8_5>
Kim et al. (2010) <doi:10.1016/j.jbankfin.2010.01.015>
Kuechler & Tappe (2013) <doi:10.1016/j.spa.2013.06.012>
Rachev et al. (2011) <doi:10.1002/9781118268070>.

URL `https://github.com/TMoek/TempStable`

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 4.2), methods

Imports copula (>= 1.1-2), doParallel (>= 1.0.12), foreach (>= 1.5.0), gsl (>= 2.1-8), hypergeo (>= 1.2-13), moments (>= 0.14), numDeriv (>= 2016.8-1), stabledist (>= 0.7-1), StableEstim (>= 2.1), rootSolve (>= 1.8), VGAM (>= 1.1-7)

RoxygenNote 7.2.3

Suggests knitr (>= 1.4), rmarkdown (>= 2.17), testthat (>= 3.1), V8 (>= 4.2)

Config/testthat/edition 3

BugReports https://github.com/TMoek/TempStable/issues

NeedsCompilation no

Author Till Massing [cre, aut],
Cedric Maximilian Juessen [aut]

Maintainer Till Massing <till.massing@uni-due.de>

Repository CRAN

Date/Publication 2023-10-24 13:40:13 UTC

Contents

`charCTS` *Characteristic function of the classical tempered stable (CTS) distribution*

Description

Theoretical characteristic function (CF) of the classical tempered stable distribution. See Kuechler & Tappe (2013) for details.

Usage

```
charCTS(
  t,
  alpha = NULL,
  deltap = NULL,
  deltam = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  functionOrigin = "massing"
)
```

Arguments

`t`	A vector of real numbers where the CF is evaluated.
`alpha`	Stability parameter. A real number between 0 and 2.
`deltap`	Scale parameter for the right tail. A real number > 0.
`deltam`	Scale parameter for the left tail. A real number > 0.
`lambdap`	Tempering parameter for the right tail. A real number > 0.
`lambdam`	Tempering parameter for the left tail. A real number > 0.
`mu`	A location parameter, any real number.
`theta`	Parameters stacked as a vector.
`functionOrigin`	A string. Either "massing", or "kim10".

Details

`theta` denotes the parameter vector `(alpha, deltap, deltam,lambdap, lambdam, mu)`. Either provide the parameters individually OR provide `theta`. Characteristic function shown here is from Massing (2023).

$$\varphi_{CTS}(t;\theta) := E_\theta\left[\mathrm{e}^{\mathrm{i}tX}\right] = \exp\left(\mathrm{i}t\mu + \delta_+\Gamma(-\alpha)\left((\lambda_+ - \mathrm{i}t)^\alpha - \lambda_+^\alpha + \mathrm{i}t\alpha\lambda_+^{\alpha-1}\right)\right.$$
$$\left. + \delta_-\Gamma(-\alpha)\left((\lambda_- + \mathrm{i}t)^\alpha - \lambda_-^\alpha - \mathrm{i}t\alpha\lambda_-^{\alpha-1}\right)\right)$$

Origin of functions Since the parameterisation can be different for this characteristic function in different approaches, the respective approach can be selected with `functionOrigin`. For the estimation function `TemperedEstim` and therefore also the Monte Carlo function `TemperedEstim_Simulation` and the calculation of the density function `dMTS` only the approach of Massing (2023) can be selected. If you want to use the approach of Kim et al. (2010) for these functions, you have to clone the package from GitHub and adapt the functions accordingly.

massing From Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'.

kim10 From Kim et al. (2010) 'Tempered stable and tempered infinitely divisible GARCH models'.

Value

The CF of the classical tempered stable distribution.

References

Kim, Y. S.; Rachev, S. T.; Bianchi, M. L. & Fabozzi, F. J.(2010), 'Tempered stable and tempered infinitely divisible GARCH models', doi:10.1016/j.jbankfin.2010.01.015

Kuechler, U. & Tappe, S. (2013), 'Tempered stable distributions and processes' doi:10.1016/j.spa.2013.06.012

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'

Examples

```
x <- seq(-10,10,0.25)
y <- charCTS(x,1.5,1,1,1,1,0)
```

`charGTS` *Characteristic function of the generalized classical tempered stable (GTS) distribution.*

Description

Theoretical characteristic function (CF) of the generalized classical tempered stable distribution. See Rachev et al. (2011) for details. The GTS is a more generalized version of the CTS charCTS, as alpha = alphap = alpham for CTS. The characteristic function is given - with a small adjustment - by Rachev et al. (2011):

Usage

```
charGTS(
  t,
  alphap = NULL,
  alpham = NULL,
  deltap = NULL,
  deltam = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL
)
```

Arguments

`t`	A vector of real numbers where the CF is evaluated.
`alphap, alpham`	Stability parameter. A real number between 0 and 2.
`deltap, deltam`	Scale parameter. A real number > 0.
`lambdap, lambdam`	Tempering parameter. A real number > 0.
`mu`	A location parameter, any real number.
`theta`	Parameters stacked as a vector.

Details

`theta` denotes the parameter vector (`alphap`, `alpham`, `deltap`,`deltam`, `lambdap`, `lambdam`, `mu`). Either provide the parameters individually OR provide `theta`. Characteristic function shown here is from Rachev et al. (2011).

$$\varphi_{GTS}(t;\theta) := E_\theta\left[\mathrm{e}^{\mathrm{i}tX}\right] = \exp\left(\mathrm{i}t\mu - \mathrm{i}t\Gamma(1-\alpha_+)\left(\delta_+\lambda_+^{\alpha_+-1}\right)\right.$$

$$+\mathrm{i}t\Gamma(1-\alpha_-)\left(\delta_-\lambda_-^{\alpha_--1}\right)$$

$$+\delta_+\Gamma(-\alpha_+)\left((\lambda_+-\mathrm{i}t)^{\alpha_+}-\lambda_+^{\alpha_+}\right)$$

$$\left.+\delta_-\Gamma(-\alpha_-)\left((\lambda_-+\mathrm{i}t)^{\alpha_-}-\lambda_-^{\alpha_-}\right)\right)$$

Value

The CF of the the generalized classical tempered stable distribution.

References

Rachev, S. T.; Kim, Y. S.; Bianchi, M. L. & Fabozzi, F. J. (2011), 'Financial models with Lévy processes and volatility clustering' doi:10.1002/9781118268070

Examples

```
x <- seq(-5,5,0.25)
y <- charGTS(x,0.3,0.2,1,1,1,1,0)
```

---

charKRTS *Characteristic function of the Kim-Rachev tempered stable distribution*

Description

Theoretical characteristic function (CF) of the Kim-Rachev tempered stable distribution.

Usage

```
charKRTS(
  t,
  alpha = NULL,
  kp = NULL,
  km = NULL,
  rp = NULL,
  rm = NULL,
  pp = NULL,
  pm = NULL,
  mu = NULL,
  theta = NULL
)
```

Arguments

`t`	A vector of real numbers where the CF is evaluated.
`alpha`	Stability parameter. A real number between 0 and 1.
`kp`, `km`, `rp`, `rm`	Parameter of KR-distribution. A real number `>0`.
`pp`, `pm`	Parameter of KR-distribution. A real number `>-alpha`.
`mu`	A location parameter, any real number.
`theta`	Parameters stacked as a vector.

Details

The CF of the RDTS distribution is given by (Rachev et al. (2011))

$$\varphi_{KRTS}(t;\theta) := E_\theta\left[\mathrm{e}^{\mathrm{i}tX}\right] = \exp\left(\mathrm{i}t\mu - \mathrm{i}t\Gamma(1-\alpha)\left(\frac{k_+r_+}{p_++1} - \frac{k_-r_-}{p_-+1}\right)\right.$$
$$\left. + k_+H(\mathrm{i}t;\alpha,r_+,p_+) + k_-H(-\mathrm{i}t;\alpha,r_-,p_-)\right),$$

where

$$H(x;\alpha,r,p) = \frac{\Gamma(-\alpha)}{p}\left(F(p,-\alpha;1+p;rx)-1\right)$$

`F` denotes the hypergeometric Function.

Value

The CF of the the Kim-Rachev tempered stable distribution.

References

Rachev, Svetlozar T. & Kim, Young Shin & Bianchi, Michele L. & Fabozzi, Frank J. (2011) 'Financial models with Lévy processes and volatility clustering' doi:10.1002/9781118268070

Examples

```
x <- seq(-5,5,0.25)
y <- charKRTS(x,0.5,1,1,1,1,1,1,0)
```

`charMTS` *Characteristic function of the modified tempered stable distribution*

Description

Theoretical characteristic function (CF) of the modified tempered stable distribution.

Usage

```
charMTS(
  t,
  alpha = NULL,
  delta = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  functionOrigin = "kim08"
)
```

Arguments

`t`	A vector of real numbers where the CF is evaluated.
`alpha`	Stability parameter. A real number between 0 and 2.
`delta`	Scale parameter. A real number > 0.
`lambdap, lambdam`	Tempering parameter. A real number > 0.

mu	A location parameter, any real number.
theta	Parameters stacked as a vector.
functionOrigin	A string. Either "kim09", "rachev11" or "kim08". Default is "kim08".

Details

theta denotes the parameter vector (alpha, delta,lambdap, lambdam, mu). Either provide the parameters individually OR provide theta. Characteristic function shown here is from Kim et al. (2008).

$$\varphi_{MTS}(t;\theta) := E_\theta\left[\mathrm{e}^{\mathrm{i}tX}\right] = \exp\left(\mathrm{i}t\mu + G_R\left(t;\alpha,\delta,\lambda_+,\lambda_-\right) + G_R\left(t;\alpha,\delta,\lambda_+,\lambda_-\right)\right),$$

where

$$G_R\left(t;\alpha,\delta,\lambda_+,\lambda_-\right) = \frac{\sqrt{\pi}\delta\Gamma(-\frac{\alpha}{2})}{2^{\frac{\alpha+3}{2}}}\left((\lambda_+^2+t^2)^{\frac{\alpha}{2}} - \lambda_+^\alpha + (\lambda_-^2+t^2)^{\frac{\alpha}{2}} - \lambda_-^\alpha\right)$$

$$G_I\left(t;\alpha,\delta,\lambda_+,\lambda_-\right) = \frac{\mathrm{i}t\delta\Gamma(\frac{1-\alpha}{2})}{2^{\frac{\alpha+1}{2}}}\left(\lambda_+^{\alpha-1}F\left(1,\frac{1-\alpha}{2};\frac{3}{2};-\frac{t^2}{\lambda_+^2}\right)\right.$$
$$\left.-\lambda_-^{\alpha-1}F\left(1,\frac{1-\alpha}{2};\frac{3}{2};-\frac{t^2}{\lambda_-^2}\right)\right)$$

F is the hypergeometric function.

Origin of functions Since the parameterisation can be different for this characteristic function in different approaches, the respective approach can be selected with functionOrigin. For the estimation function TemperedEstim and therefore also the Monte Carlo function TemperedEstim_Simulation and the calculation of the density function dMTS only the approach of Kim et al. (2008) or Rachev et al. (2011) can be selected. If you want to use the approach of Kim et al. (2009) for these functions, you have to clone the package from GitHub and adapt the functions accordingly.

kim09 From Kim et al. (2009) 'The modified tempered stable distribution, GARCH-models and option pricing'. Here alpha is in (-Inf,1) except 0.5.

kim08 From Kim et al. (2008) 'Financial market models with Levy processes and time-varying volatility'. Without further coding, this is the selected function for estimation function from this package.

rachev11 From Rachev et al. (2011) 'Financial Models with Levy Processes and time-varying volatility'. Similar to kim08

Value

The CF of the the modified tempered stable distribution.

References

Kim, Y. S.; Rachev, S. T.; Bianchi, M. L. & Fabozzi, F. J. (2008), 'Financial market models with lévy processes and time-varying volatility' doi:10.1016/j.jbankfin.2007.11.004

Kim, Y. S.; Rachev, S. T.; Bianchi, M. L. & Fabozzi, F. J. (2009), 'A New Tempered Stable Distribution and Its Application to Finance' doi:10.1007/9783790820508_5

Rachev, S. T.; Kim, Y. S.; Bianchi, M. L. & Fabozzi, F. J. (2011), 'Financial models with Lévy processes and volatility clustering' doi:10.1002/9781118268070

Examples

```
x <- seq(-5,5,0.1)
y <- charMTS(x, 0.5,1,1,1,0)
```

---

charNTS *Characteristic function of the normal tempered stable (NTS) distribution*

Description

Theoretical characteristic function (CF) of the normal tempered stable distribution. See Rachev et al. (2011) for details.

Usage

```
charNTS(
  t,
  alpha = NULL,
  beta = NULL,
  delta = NULL,
  lambda = NULL,
  mu = NULL,
  theta = NULL
)
```

Arguments

t	A vector of real numbers where the CF is evaluated.
alpha	Stability parameter. A real number between 0 and 1.
beta	Skewness parameter. Any real number.
delta	Scale parameter. A real number > 0.
lambda	Tempering parameter. A real number > 0.
mu	A location parameter, any real number.
theta	A vector of all other arguments.

Details

`theta` denotes the parameter vector `(alpha, beta, delta, lambda,mu)`. Either provide the parameters individually OR provide `theta`.

$$\varphi_{NTS}(t;\theta) = E\left[\mathrm{e}^{\mathrm{i}tZ}\right] = \exp\left(\mathrm{i}t\mu + \delta\Gamma(-\alpha)\left((\lambda - \mathrm{i}t\beta + t^2/2)^\alpha - \lambda^\alpha\right)\right)$$

Value

The CF of the normal tempered stable distribution.

References

Massing, T. (2022), 'Parametric Estimation of Tempered Stable Laws'

Rachev, Svetlozar T. & Kim, Young Shin & Bianchi, Michele L. & Fabozzi, Frank J. (2011) 'Financial models with Lévy processes and volatility clustering' doi:10.1002/9781118268070

Examples

```
x <- seq(-10,10,0.25)
y <- charNTS(x,0.5,1,1,1,0)
```

---

charRDTS *Characteristic function of the rapidly decreasing tempered stable (RDTS) distribution*

Description

Theoretical characteristic function (CF) of the rapidly decreasing tempered stable distribution.

Usage

```
charRDTS(
  t,
  alpha = NULL,
  delta = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL
)
```

Arguments

`t`	A vector of real numbers where the CF is evaluated.
`alpha`	Stability parameter. A real number between 0 and 2.
`delta`	Scale parameter. A real number > 0.
`lambdap, lambdam`	Tempering parameter. A real number > 0.
`mu`	A location parameter, any real number.
`theta`	Parameters stacked as a vector.

Details

The CF of the RDTS distribution is given by (Rachev et al. (2011)):

$$\varphi_{RDTS}(t;\theta) := E_\theta\left[\mathrm{e}^{\mathrm{i}tX}\right] = \exp\left(\mathrm{i}t\mu + \delta(G(\mathrm{i}t;\alpha,\lambda_+) + G(-\mathrm{i}t;\alpha,\lambda_-))\right),$$

where

$$G(x;\alpha,r,\lambda) = 2^{-\frac{\alpha}{2}-1}\lambda^\alpha\Gamma\left(-\frac{\alpha}{2}\right)\left(M\left(-\frac{\alpha}{2},\frac{1}{2};\frac{x^2}{2\lambda^2}\right)-1\right)$$
$$+2^{-\frac{\alpha}{2}-\frac{1}{2}}\lambda^{\alpha-1}x\Gamma\left(\frac{1-\alpha}{2}\right)\left(M\left(\frac{1-\alpha}{2},\frac{3}{2};\frac{x^2}{2\lambda^2}\right)-1\right).$$

`M` stands for the confluent hypergeometric function.

Value

The CF of the the rapidly decreasing tempered stable distribution.

References

Rachev, Svetlozar T. & Kim, Young Shin & Bianchi, Michele L. & Fabozzi, Frank J. (2011) 'Financial models with Lévy processes and volatility clustering' doi:10.1002/9781118268070

Examples

```
x <- seq(-5,5,0.25)
y <- charRDTS(x,0.5,1,1,1,0)
```

---

charTSS — *Characteristic function of the tempered stable subordinator*

Description

Theoretical characteristic function (CF) of the distribution of the tempered stable subordinator. See Kawai & Masuda (2011) for details.

Usage

```
charTSS(t, alpha = NULL, delta = NULL, lambda = NULL, theta = NULL)
```

Arguments

`t`	A vector of real numbers where the CF is evaluated.
`alpha`	Stability parameter. A real number between 0 and 1.
`delta`	Scale parameter. A real number > 0.
`lambda`	Tempering parameter. A real number > 0.
`theta`	Parameters stacked as a vector.

Details

`theta` denotes the parameter vector `(alpha, delta, lambda)`. Either provide the parameters `alpha`, `delta`, `lambda` individually OR provide `theta`.

$$\varphi_{TSS}(t;\theta) := E_\theta\left[\mathrm{e}^{\mathrm{i}tY}\right] = \exp\left(\delta\Gamma(-\alpha)\left((\lambda - \mathrm{i}t)^\alpha - \lambda^\alpha\right)\right)$$

Value

The CF of the tempered stable subordinator distribution.

References

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'

Kawai, R. & Masuda, H. (2011), 'On simulation of tempered stable random variates' doi:10.1016/j.cam.2010.12.014

Kuechler, U. & Tappe, S. (2013), 'Tempered stable distributions and processes' doi:10.1016/j.spa.2013.06.012

Examples

```
x <- seq(-10,10,0.25)
y <- charTSS(x,0.5,1,1)
```

---

dCTS — *Density function of the classical tempered stable (CTS) distribution*

Description

The probability density function (PDF) of the classical tempered stable distributions is not available in closed form. Relies on fast Fourier transform (FFT) applied to the characteristic function.

Usage

```
dCTS(
  x,
  alpha = NULL,
  deltap = NULL,
  deltam = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  dens_method = "FFT",
  a = -20,
  b = 20,
  nf = 2048,
  ...
)
```

Arguments

x	A numeric vector of quantiles.
alpha	Stability parameter. A real number between 0 and 2.
deltap	Scale parameter for the right tail. A real number > 0.
deltam	Scale parameter for the left tail. A real number > 0.
lambdap	Tempering parameter for the right tail. A real number > 0.
lambdam	Tempering parameter for the left tail. A real number > 0.
mu	A location parameter, any real number.
theta	Parameters stacked as a vector.
dens_method	Algorithm for numerical evaluation. Choose between "FFT" (default) and "Conv".
a	Starting point of FFT, if dens_method == "FFT". -20 by default.
b	Ending point of FFT, if dens_method == "FFT". 20 by default.
nf	Pieces the transformation is divided in. Limited to power-of-two size. 2048 by default.
...	Possibility to modify charCTS().

Details

theta denotes the parameter vector (alpha, deltap, deltam,lambdap, lambdam, mu). Either provide the parameters individually OR provide theta. Methods include the FFT or alternatively by convolving two totally positively skewed tempered stable distributions, see Massing (2022).

The "FFT" method is automatically selected for Mac users, as the "Conv" method causes problems.

Value

As x is a numeric vector, the return value is also a numeric vector of densities.

References

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'

Examples

```
x <- seq(0,15,0.25)
y <- dCTS(x,0.6,1,1,1,1,1,NULL,"FFT",-20,20,2048)
plot(x,y)
```

`dGTS` *Density function of generalized classical tempered stable distribution*

Description

The probability density function (PDF) of the generalized classical tempered stable (GTS) distributions is not available in closed form. Relies on fast Fourier transform (FFT) applied to the characteristic function.

Usage

```
dGTS(
  x,
  alphap = NULL,
  alpham = NULL,
  deltap = NULL,
  deltam = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  dens_method = "FFT",
  a = -20,
  b = 20,
  nf = 2048
)
```

Arguments

`x`	A numeric vector of positive quantiles.
`alphap, alpham`	Stability parameter. A real number between 0 and 2.
`deltap, deltam`	Scale parameter. A real number > 0.
`lambdap, lambdam`	Tempering parameter. A real number > 0.
`mu`	A location parameter, any real number.
`theta`	Parameters stacked as a vector.
`dens_method`	A method to get the density function. Here, only "FFT" is available.
`a`	Starting point of FFT, if `dens_method == "FFT"`. -20 by default.
`b`	Ending point of FFT, if `dens_method == "FFT"`. 20 by default.
`nf`	Pieces the transformation is divided in. Limited to power-of-two size. Default is 2048.

Value

As `q` is a numeric vector, the return value is also a numeric vector of probabilities.

Examples

```
x <- seq(-5,5,0.25)
y <- dGTS(x,0.3,0.2,1,1,1,1,0)
```

---

dKRTS *Density Function of the Kim-Rachev tempered stable distribution*

Description

The probability density function (PDF) of the Kim-Rachev tempered stable distributions is not available in closed form. Relies on fast Fourier transform (FFT) applied to the characteristic function.

Usage

```
dKRTS(
  x,
  alpha = NULL,
  kp = NULL,
  km = NULL,
  rp = NULL,
  rm = NULL,
  pp = NULL,
  pm = NULL,
  mu = NULL,
  theta = NULL,
  dens_method = "FFT",
  a = -20,
  b = 20,
  nf = 256
)
```

Arguments

`x`	A numeric vector of positive quantiles.
`alpha`	Stability parameter. A real number between 0 and 1.
`kp`, `km`, `rp`, `rm`	Parameter of KR-distribution. A real number `>0`.
`pp`, `pm`	Parameter of KR-distribution. A real number `>-alpha`.
`mu`	A location parameter, any real number.
`theta`	Parameters stacked as a vector.
`dens_method`	Algorithm for numerical evaluation. Here you can only choose `"FFT"`.
`a`	Starting point of FFT, if `dens_method == "FFT"`. -20 by default.
`b`	Ending point of FFT, if `dens_method == "FFT"`. 20 by default.
`nf`	Pieces the transformation is divided in. Limited to power-of-two size. 256 by default.

Details

`theta` denotes the parameter vector (`alpha, kp, km,rp, rm, pp. pm, mu`). Either provide the parameters individually OR provide `theta`.

For examples, compare with `dCTS()`.

Value

The CF of the the Kim-Rachev tempered stable distribution.

---

dMTS — *Density function of the modified tempered stable (MTS) distribution*

Description

`theta` denotes the parameter vector (`alpha, delta, lambdap,lambdam, mu`). The probability density function (PDF) of the modified tempered stable distributions is not available in closed form. Relies on fast Fourier transform (FFT) applied to the characteristic function.

Usage

```
dMTS(
  x,
  alpha = NULL,
  delta = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  dens_method = "FFT",
  a = -20,
  b = 20,
  nf = 256
)
```

Arguments

`x`	A numeric vector of quantiles.
`alpha`	Stability parameter. A real number between 0 and 2.
`delta`	Scale parameter. A real number > 0.
`lambdap, lambdam`	Tempering parameter. A real number > 0.
`mu`	A location parameter, any real number.
`theta`	Parameters stacked as a vector.
`dens_method`	A method to get the density function. Here, only "FFT" is available.

a	Starting point of FFT, if `dens_method == "FFT"`. -20 by default.
b	Ending point of FFT, if `dens_method == "FFT"`. 20 by default.
nf	Pieces the transformation is divided in. Limited to power-of-two size. 256 by default.

Details

For examples, compare with `dCTS()`.

Value

As `x` is a numeric vector, the return value is also a numeric vector of densities.

`dNTS` *Density function of the normal tempered stable (NTS) distribution*

Description

The probability density function (PDF) of the normal tempered stable distributions is not available in closed form. Relies on fast Fourier transform (FFT) applied to the characteristic function.

Usage

```
dNTS(
  x,
  alpha = NULL,
  beta = NULL,
  delta = NULL,
  lambda = NULL,
  mu = NULL,
  theta = NULL,
  dens_method = "FFT",
  a = -20,
  b = 20,
  nf = 2048
)
```

Arguments

x	A numeric vector of quantile.
alpha	A real number between 0 and 1.
beta	Any real number.
delta	A real number > 0.
lambda	A real number > 0.
mu	A location parameter, any real number.

theta	A vector of all other arguments.
dens_method	Currently, useless param, as it does nothing and FFT is always used.
a	Starting point of FFT, if dens_method == "FFT". -20 by default.
b	Ending point of FFT, if dens_method == "FFT". 20 by default.
nf	Pieces the transformation is divided in. Limited to power-of-two size.

Details

theta denotes the parameter vector (alpha, beta, delta, lambda,mu). Either provide the parameters individually OR provide theta. Currently, the only method is FFT.

Value

As x is a numeric vector, the return value is also a numeric vector of densities.

References

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'

Examples

```
x <- seq(0,15,0.25)
y <- dNTS(x,0.8,1,1,1,1)
plot(x,y)
```

---

dRDTS *Density function of the rapidly decreasing tempered stable (CTS) distribution*

Description

The probability density function (PDF) of the rapidly decreasing tempered stable distributions is not available in closed form. Relies on fast Fourier transform (FFT) applied to the characteristic function.

Usage

```
dRDTS(
  x,
  alpha = NULL,
  delta = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  dens_method = "FFT",
```

```
  a = -20,
  b = 20,
  nf = 256
)
```

Arguments

x	A numeric vector of quantiles.
alpha	Stability parameter. A real number between 0 and 2.
delta	Scale parameter for the left tail. A real number > 0.
lambdap	Tempering parameter for the right tail. A real number > 0.
lambdam	Tempering parameter for the left tail. A real number > 0.
mu	A location parameter, any real number.
theta	Parameters stacked as a vector.
dens_method	Algorithm for numerical evaluation. Choose "FFT".
a	Starting point of FFT, if dens_method == "FFT". -20 by default.
b	Ending point of FFT, if dens_method == "FFT". 20 by default.
nf	Pieces the transformation is divided in. Limited to power-of-two size. 256 by default.

Details

theta denotes the parameter vector (alpha, delta,lambdap, lambdam, mu). Either provide the parameters individually OR provide theta. Methods include only the the Fast Fourier Transform (FFT).

For examples, compare with dCTS().

Value

As x is a numeric vector, the return value is also a numeric vector of densities.

References

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'

---

dTSS *Density function of the tempered stable subordinator (TSS) distribution*

Description

The probability density function (PDF) of tempered stable subordinator distribution. It can be computed via the stable distribution (see details) using the stabledist package.

Usage

```
dTSS(x, alpha = NULL, delta = NULL, lambda = NULL, theta = NULL)
```

Arguments

x	A numeric vector of positive quantiles.
alpha	Stability parameter. A real number between 0 and 1.
delta	Scale parameter. A real number > 0.
lambda	Tempering parameter. A real number > 0.
theta	Parameters stacked as a vector.

Details

`theta` denotes the parameter vector `(alpha, delta, lambda)`. Either provide the parameters `alpha`, `delta`, `lambda` individually OR provide `theta`.

$$f_{TSS}(y;\theta) = \mathrm{e}^{-\lambda y-\lambda^\alpha \delta \Gamma(-\alpha)} f_{S(\alpha,\delta)}(y),$$

where

$$f_{S(\alpha,\delta)}$$

is the density of the stable subordinator.

Value

As `x` is a numeric vector, the return value is also a numeric vector of probability densities.

References

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'

Kawai, R. & Masuda, H. (2011), 'On simulation of tempered stable random variates' doi:10.1016/j.cam.2010.12.014

Examples

```
x <- seq(0,15,0.25)
y <- dTSS(x,0.5,1,0.3)
plot(x,y)
```

`parallelizeMCsimulation` *Function to parallelize the Monte Carlo Simulation*

Description

Since the Monte Carlo Simulation is very computationally intensive, it may be worthwhile to split it across all available processor cores. To do this, simply pass all the parameters from the `TemperedEstim_Simulation()` function to this function in the same way.

Usage

```
parallelizeMCsimulation(
  ParameterMatrix,
  MCparam = 10000,
  SampleSizes = c(200),
  saveOutput = FALSE,
  cores = 2,
  SeedOptions = NULL,
  iterationDisplayToFileSystem = FALSE,
  ...
)
```

Arguments

`ParameterMatrix`
: The matrix is to be composed of vectors, row by row. Each vector must fit the pattern of theta of the `TemperedType`. Compared to the function `TemperedEstim_Simulation()`, the matrix here may contain only one parameter vector.

`MCparam`
: Number of Monte Carlo simulation for each couple of parameter, default=100; integer

`SampleSizes`
: Sample sizes to be used to simulate the data. By default, we use 200 (small sample size). Vector of integer. Compared to the function `TemperedEstim_Simulation()`, the vector here may contain only one integer.

`saveOutput`
: Logical flag: In the function `TemperedEstim_Simulation()` the argument can be true. Then an external csv file is created. Here the argument must be false. The output of the values works in this function exclusively via the return of the function.

`cores`
: size of cluster for parallelization. Positive Integer.

`SeedOptions`
: is an argument what can be used in `TemperedEstim_Simulation()` but must be NULL here.

`iterationDisplayToFileSystem`
: creates a text file in your file system that displays the current iteration of the simulation.

`...`
: The function works only if all necessary arguments from the function `TemperedEstim_Simulation()` are passed. See description and details.

Details

In this function exactly the arguments must be passed, which are also needed for the function `TemperedEstim_Simulation()`. However, a few functions of `TemperedEstim_Simulation()` are not possible here. The restrictions are described in more detail for the individual arguments.

In addition to the arguments of function `TemperedEstim_Simulation()`, the argument "cores" can be assigned an integer value. This value determines how many different processes are to be parallelized. If value is `NULL`, R tries to read out how many cores the processor has and passes this value to "cores".

During the simulation, the progress of the simulation can be viewed in a file in the workspace named "IterationControlForParallelization.txt".

Value

The return object is a list of 2. Results of the simulation are listed in `$outputMat`.

---

pCTS — *Cumulative probability function of the classical tempered stable (CTS) distribution*

Description

The cumulative probability distribution function (CDF) of the classical tempered stable distribution.

Usage

```
pCTS(
  q,
  alpha = NULL,
  deltap = NULL,
  deltam = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  a = -40,
  b = 40,
  nf = 2^13,
  ...
)
```

Arguments

`q`	A numeric vector of quantiles.
`alpha`	Stability parameter. A real number between 0 and 2.
`deltap`	Scale parameter for the right tail. A real number > 0.

deltam	Scale parameter for the left tail. A real number > 0.
lambdap	Tempering parameter for the right tail. A real number > 0.
lambdam	Tempering parameter for the left tail. A real number > 0.
mu	A location parameter, any real number.
theta	Parameters stacked as a vector.
a	Starting point of FFT, if `dens_method == "FFT"`. -20 by default.
b	Ending point of FFT, if `dens_method == "FFT"`. 20 by default.
nf	Pieces the transformation is divided in. Limited to power-of-two size.
...	Possibility to modify `stats::integrate()`.

Details

`theta` denotes the parameter vector `(alpha, deltap, deltam,lambdap, lambdam, mu)`. Either provide the parameters individually OR provide `theta`. The function integrates the PDF numerically with `integrate()`.

Value

As `q` is a numeric vector, the return value is also a numeric vector of probabilities.

See Also

See also the `dCTS()` density-function.

Examples

```
x <- seq(-5,5,0.25)
y <- pCTS(x,0.5,1,1,1,1,1)
plot(x,y)
```

---

pGTS — *Cumulative probability function of the generalized classical tempered stable (GTS) distribution*

Description

The cumulative probability distribution function (CDF) of the generalized classical tempered stable distribution.

Usage

```
pGTS(
  q,
  alphap = NULL,
  alpham = NULL,
  deltap = NULL,
  deltam = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  dens_method = "FFT",
  a = -40,
  b = 40,
  nf = 2048,
  ...
)
```

Arguments

q	A numeric vector of quantiles.
alphap, alpham	Stability parameter. A real number between 0 and 2.
deltap	Scale parameter for the right tail. A real number > 0.
deltam	Scale parameter for the left tail. A real number > 0.
lambdap	Tempering parameter for the right tail. A real number > 0.
lambdam	Tempering parameter for the left tail. A real number > 0.
mu	A location parameter, any real number.
theta	Parameters stacked as a vector.
dens_method	A method to get the density function. Here, only "FFT" is available.
a	Starting point of FFT, if dens_method == "FFT". -20 by default.
b	Ending point of FFT, if dens_method == "FFT". 20 by default.
nf	Pieces the transformation is divided in. Limited to power-of-two size.
...	Possibility to modify stats::integrate().

Details

theta denotes the parameter vector (alphap, alpham, deltap,deltam, lambdap, lambdam, mu). Either provide the parameters individually OR provide theta. The function integrates the PDF numerically with integrate().

Value

As q is a numeric vector, the return value is also a numeric vector of probabilities.

See Also

See also the dGTS() density-function.

Examples

```
x <- seq(-1,1,1)
y <- pGTS(x,0.5,1.5,1,1,1,1,1)
```

---

pKRTS — *Cumulative probability distribution function of the Kim-Rachev tempered stable (KRTS) distribution*

Description

The cumulative probability distribution function (CDF) of the Kim-Rachev tempered stable distribution.

Usage

```
pKRTS(
  q,
  alpha = NULL,
  kp = NULL,
  km = NULL,
  rp = NULL,
  rm = NULL,
  pp = NULL,
  pm = NULL,
  mu = NULL,
  theta = NULL,
  dens_method = "FFT",
  a = -40,
  b = 40,
  nf = 2048,
  ...
)
```

Arguments

q	A vector of real numbers where the CF is evaluated.
alpha	Stability parameter. A real number between 0 and 2.
kp, km, rp, rm	Parameter of KR-distribution. A real number >0.
pp, pm	Parameter of KR-distribution. A real number >-alpha.
mu	A location parameter, any real number.

theta	Parameters stacked as a vector.
dens_method	Algorithm for numerical evaluation. Currently, only "FFT" available.
a	Starting point of FFT, if dens_method == "FFT". -40 by default.
b	Ending point of FFT, if dens_method == "FFT". 40 by default.
nf	Pieces the transformation is divided in. Limited to power-of-two size.
...	Possibility to modify stats::integrate().

Details

theta denotes the parameter vector (alpha, kp, km,rp, rm, pp. pm, mu)). Either provide the parameters individually OR provide theta. The function integrates the PDF numerically with integrate().

Value

As q is a numeric vector, the return value is also a numeric vector of probabilities.

See Also

See also the dKRTS() density-function.

---

pMTS *Cumulative probability function of the modified tempered stable (MTS) distribution*

Description

The cumulative probability distribution function (CDF) of the modified tempered stable distribution.

Usage

```
pMTS(
  q,
  alpha = NULL,
  delta = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  dens_method = "FFT",
  a = -40,
  b = 40,
  nf = 2048,
  ...
)
```

Arguments

`q`	A vector of real numbers where the CF is evaluated.
`alpha`	Stability parameter. A real number between 0 and 2.
`delta`	Scale parameter. A real number > 0.
`lambdap`, `lambdam`	Tempering parameter. A real number > 0.
`mu`	A location parameter, any real number.
`theta`	Parameters stacked as a vector.
`dens_method`	A method to get the density function. Here, only "FFT" is available.
`a`	Starting point of FFT, if `dens_method == "FFT"`. -20 by default.
`b`	Ending point of FFT, if `dens_method == "FFT"`. 20 by default.
`nf`	Pieces the transformation is divided in. Limited to power-of-two size.
`...`	Possibility to modify `stats::integrate()`.

Details

`theta` denotes the parameter vector `(alpha, delta,lambdap, lambdam, mu)`. Either provide the parameters individually OR provide `theta`. The function integrates the PDF numerically with `integrate()`.

Value

As `q` is a numeric vector, the return value is also a numeric vector of probabilities.

`pNTS` *Cumulative probability function of the normal tempered stable (NTS) distribution*

Description

The cumulative probability distribution function (CDF) of the normal tempered stable distribution.

Usage

```
pNTS(
  q,
  alpha = NULL,
  beta = NULL,
  delta = NULL,
  lambda = NULL,
  mu = NULL,
  theta = NULL,
  a = -40,
  b = 40,
  nf = 2^11,
  ...
)
```

Arguments

q	A numeric vector of quantile.
alpha	A real number between 0 and 1.
beta	Any real number.
delta	A real number > 0.
lambda	A real number > 0.
mu	A location parameter, any real number.
theta	A vector of all other arguments.
a	Starting point integrate density function. -40 by default.
b	Ending point of integrate density function. 40 by default.
nf	Pieces the fast Fourier transformation is divided in. Limited to power-of-two size. 2^11 by default.
...	Change parameters in dNTS()

Details

theta denotes the parameter vector (alpha, beta, delta, lambda,mu). Either provide the parameters individually OR provide theta. The function integrates the PDF numerically with integrate().

Value

As q is a numeric vector, the return value is also a numeric vector of probabilities.

See Also

See also the dNTS() density-function.

Examples

```
x <- seq(-5,5,0.25)
y <- pNTS(x,0.5,1,1,1,1)
plot(x,y)
```

---

pRDTS *Cumulative probability function of the rapidly decreasing tempered stable (RDTS) distribution*

Description

The cumulative probability distribution function (CDF) of the rapidly decreasing tempered stable distribution.

Usage

```
pRDTS(
  q,
  alpha = NULL,
  delta = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  dens_method = "FFT",
  a = -130,
  b = 130,
  nf = 2048,
  ...
)
```

Arguments

q	A numeric vector of quantiles.
alpha	Stability parameter. A real number between 0 and 2.
delta	Scale parameter for the left tail. A real number > 0.
lambdap	Tempering parameter for the right tail. A real number > 0.
lambdam	Tempering parameter for the left tail. A real number > 0.
mu	A location parameter, any real number.
theta	Parameters stacked as a vector.
dens_method	Algorithm for numerical evaluation. Currently, only "FFT" available.
a	Starting point of FFT, if dens_method == "FFT". -20 by default.
b	Ending point of FFT, if dens_method == "FFT". 20 by default.
nf	Pieces the transformation is divided in. Limited to power-of-two size.
...	Possibility to modify stats::integrate().

Details

theta denotes the parameter vector (alpha, delta,lambdap, lambdam, mu). Either provide the parameters individually OR provide theta. The function integrates the PDF numerically with integrate().

Value

As q is a numeric vector, the return value is also a numeric vector of probabilities.

See Also

See also the dRDTS() density-function.

pTSS *Cumulative probability distribution function of the tempered stable subordinator distribution*

Description

The cumulative probability distribution function (CDF) of the tempered stable subordinator distribution.

Usage

```
pTSS(
  q,
  alpha = NULL,
  delta = NULL,
  lambda = NULL,
  theta = NULL,
  pmethod = "integrate",
  N = 8192,
  ...
)
```

Arguments

q	A numeric vector of positive quantiles.
alpha	Stability parameter. A real number between 0 and 1.
delta	Scale parameter. A real number > 0.
lambda	Tempering parameter. A real number > 0.
theta	Parameters stacked as a vector.
pmethod	A string. If not "integrate", the function `chartocdf()` will be triggered.
N	is a power of two & N >= 1024. if `pmethod != "integrate"`. 8192 by default. Relevant for
...	Possibility to modify `stats::integrate()`.

Details

`theta` denotes the parameter vector `(alpha, delta, lambda)`. Either provide the parameters `alpha`, `delta`, `lambda` individually OR provide `theta`. The function integrates the PDF numerically with `integrate()`.

Value

As `q` is a numeric vector, the return value is also a numeric vector of probabilities.

See Also

See also the dTSS() density-function.

Examples

```
x <- seq(0,15,0.5)
y <- pTSS(x,0.7,1.354,0.3)
plot(x,y)
```

---

qCTS *Quantile function of the classical tempered stable (CTS)*

Description

The quantile function of the classical tempered stable (CTS).

Usage

```
qCTS(
  p,
  alpha = NULL,
  deltap = NULL,
  deltam = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  qmin = NULL,
  qmax = NULL,
  ...
)
```

Arguments

p	A numeric vector of probabilities. Each probability must be a real number >0 and <1.
alpha	Stability parameter. A real number between 0 and 2.
deltap	Scale parameter for the right tail. A real number > 0.
deltam	Scale parameter for the left tail. A real number > 0.
lambdap	Tempering parameter for the right tail. A real number > 0.
lambdam	Tempering parameter for the left tail. A real number > 0.
mu	A location parameter, any real number.
theta	Parameters stacked as a vector.
qmin, qmax	Limits of the interval. Will be computed if ==NULL.
...	Modify pTSS() and stats::uniroot().

Details

`theta` denotes the parameter vector (`alpha`, `deltap`, `deltam`,`lambdap`, `lambdam`, `mu`). Either provide the parameters individually OR provide `theta`. The function searches for a root between `qmin` and `qmax` with `uniroot`. Boundaries can either be supplied by the user or a built-in approach using the stable distribution is used.

Value

As `p` is a numeric vector, the return value is also a numeric vector of quantiles.

See Also

See also the `pCTS()` probability function.

Examples

```
qCTS(0.5,1.5,1,1,1,1,1)
```

`qNTS` *Quantile function of the normal tempered stable (NTS)*

Description

The quantile function of the normal tempered stable (NTS).

Usage

```
qNTS(
  p,
  alpha = NULL,
  beta = NULL,
  delta = NULL,
  lambda = NULL,
  mu = NULL,
  theta = NULL,
  qmin = NULL,
  qmax = NULL,
  ...
)
```

Arguments

p	A numeric vector of probabilities. Each probability must be a real number >0 and <1.
alpha	A real number between 0 and 1.
beta	A gap holder.
delta	A real number > 0.
lambda	A real number >= 0.
mu	A location parameter, any real number.
theta	A vector of all other arguments.
qmin, qmax	Limits of the interval. Will be computed if ==NULL.
...	Modify pNTS() and stats::uniroot().

Details

theta denotes the parameter vector (alpha, beta, delta, lambda,mu). Either provide the parameters individually OR provide theta. The function searches for a root between qmin and qmax with uniroot. Boundaries can either be supplied by the user or a built-in approach using the stable distribution is used.

Value

As p is a numeric vector, the return value is also a numeric vector of quantiles.

See Also

See also the pNTS() probability function.

Examples

```
qNTS(0.1,0.5,1,1,1,1)
qNTS(0.3,0.6,1,1,1,1,NULL)
```

---

qTSS *Quantile function of the tempered stable subordinator distribution*

Description

The quantile function of the tempered stable subordinator distribution.

Usage

```
qTSS(
  p,
  alpha = NULL,
  delta = NULL,
  lambda = NULL,
  theta = NULL,
  qmin = NULL,
  qmax = NULL,
  ...
)
```

Arguments

p	A numeric vector of probabilities. Each probability must be a real number >0 and <1.
alpha	Stability parameter. A real number between 0 and 1.
delta	Scale parameter. A real number > 0.
lambda	Tempering parameter. A real number > 0.
theta	Parameters stacked as a vector.
qmin, qmax	Limits of the interval. Will be computed if ==NULL.
...	Modify pTSS() and stats::uniroot().

Details

theta denotes the parameter vector (alpha, delta, lambda). Either provide the parameters alpha, delta, lambda individually OR provide theta. The function searches for a root between qmin and qmax with uniroot. Boundaries can either be supplied by the user or a built-in approach using the stable distribution is used.

Value

As p is a numeric vector, the return value is also a numeric vector of quantiles.

See Also

See also the pTSS() probability function.

Examples

```
qTSS(0.5,0.5,5,0.01)
qTSS(0.5,0.9,1,10,NULL)
```

rCTS *Function to generate random variates of CTS distribution.*

Description

Generates n random numbers distributed according to the classical tempered stable (CTS) distribution.

Usage

```
rCTS(
  n,
  alpha = NULL,
  deltap = NULL,
  deltam = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  methodR = "TM",
  k = 10000,
  c = 1
)
```

Arguments

n	sample size (integer).
alpha	Stability parameter. A real number between 0 and 2.
deltap	Scale parameter for the right tail. A real number > 0.
deltam	Scale parameter for the left tail. A real number > 0.
lambdap	Tempering parameter for the right tail. A real number > 0.
lambdam	Tempering parameter for the left tail. A real number > 0.
mu	A location parameter, any real number.
theta	Parameters stacked as a vector.
methodR	A String. Either "TM","AR" or "SR".
k	integer: the level of truncation, if `methodR == "SR"`. 10000 by default.
c	A real number. Only relevant for `methodR == "AR"`. 1 by default.

Details

`theta` denotes the parameter vector `(alpha, deltap, deltam,lambdap, lambdam, mu)`. Either provide the parameters individually OR provide `theta`. "AR" stands for the approximate Acceptance-Rejection Method and "SR" for a truncated infinite shot noise series representation. "TM" stands for Two Methods as two different methods are used depending on which will be faster. "TM" works only for alpha < 1. In this method the function copula::retstable() is called. For alpha < 1, "TM" is the default method, while "AR" for alpha > 1 is the default method.

It is recommended to check the generated random numbers once for each distribution using the density function. If the random numbers are shifted, e.g. for the method "SR", it may be worthwhile to increase k.

For more details, see references.

Value

Generates n random numbers of the CTS distribution.

References

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'

Kawai, R & Masuda, H (2011), 'On simulation of tempered stable random variates' doi:10.1016/j.cam.2010.12.014

Hofert, M (2011), 'Sampling Exponentially Tilted Stable Distributions' doi:10.1145/2043635.2043638

See Also

copula::retstable() as "TM" uses this function.

Examples

```
rCTS(10,0.5,1,1,1,1,1,NULL,"SR",10)
rCTS(10,0.5,1,1,1,1,1,NULL,"aAR")
```

---

rGTS *Function to generate random variates of GTS distribution.*

Description

Generates n random numbers distributed according to the generalized classical tempered stable (GTS) distribution.

Usage

```
rGTS(
  n,
  alphap = NULL,
  alpham = NULL,
  deltap = NULL,
  deltam = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
```

```
    theta = NULL,
    methodR = "AR",
    k = 10000,
    c = 1
)
```

Arguments

n	sample size (integer).
alphap, alpham	Stability parameter. A real number between 0 and 2.
deltap	Scale parameter for the right tail. A real number > 0.
deltam	Scale parameter for the left tail. A real number > 0.
lambdap	Tempering parameter for the right tail. A real number > 0.
lambdam	Tempering parameter for the left tail. A real number > 0.
mu	A location parameter, any real number.
theta	Parameters stacked as a vector.
methodR	A String. Either "TM","AR" or "SR".
k	integer: the level of truncation, if `methodR == "SR"`. 10000 by default.
c	A real number. Only relevant for `methodR == "AR"`. 1 by default.

Details

`theta` denotes the parameter vector `(alphap, alpham, deltap,deltam, lambdap, lambdam, mu)`. Either provide the parameters individually OR provide `theta`. "AR" stands for the approximate Acceptance-Rejection Method and "SR" for a truncated infinite shot noise series representation.

It is recommended to check the generated random numbers once for each distribution using the density function. If the random numbers are shifted, e.g. for the method "SR", it may be worthwhile to increase k.

For more details, see references.

Value

Generates `n` random numbers of the CTS distribution.

References

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'

Kawai, R & Masuda, H (2011), 'On simulation of tempered stable random variates' doi:10.1016/j.cam.2010.12.014

Hofert, M (2011), 'Sampling Exponentially Tilted Stable Distributions' doi:10.1145/2043635.2043638

See Also

`copula::retstable()` as "TM" uses this function and `rCTS()`.

Examples

```
rGTS(2,1.5,0.5,1,1,1,1,0,NULL,"SR")
rGTS(2,1.5,0.5,1,1,1,1,1,NULL,"aAR")
```

---

rKRTS *Function to generate random variates of KRTS distribution.*

Description

Generates n random numbers distributed according to the Kim-Rachev tempered stable (KRTS) distribution.

Usage

```
rKRTS(
  n,
  alpha = NULL,
  kp = NULL,
  km = NULL,
  rp = NULL,
  rm = NULL,
  pp = NULL,
  pm = NULL,
  mu = NULL,
  theta = NULL,
  methodR = "SR",
  k = 10000
)
```

Arguments

n	sample size (integer).
alpha	Stability parameter. A real number between 0 and 2.
kp, km, rp, rm	Parameter of KR-distribution. A real number >0.
pp, pm	Parameter of KR-distribution. A real number >-alpha.
mu	A location parameter, any real number.
theta	Parameters stacked as a vector.
methodR	A String. Only "SR" is available here.
k	integer: the level of truncation, if methodR == "SR". 10000 by default.

Details

theta denotes the parameter vector (alpha, kp, km,rp, rm, pp. pm, mu). Either provide the parameters individually OR provide theta. "SR" stands for a truncated infinite shot noise series representation. Currently, this method is the only implemented to generate random variates. The series representation is given by Bianchi et a. (2010).

It is recommended to check the generated random numbers once for each distribution using the density function. If the random numbers are shifted, e.g. for the method "SR", it may be worthwhile to increase k.

For more details, see references.

Value

Generates n random numbers of the KRTS distribution.

References

Bianchi, M. L.; Rachev, S. T.; Kim, Y. S. & Fabozzi, F. J. (2010), 'Tempered stable distributions and processes in finance: Numerical analysis' doi:10.1007/9788847014817

Examples

```
rKRTS(1,0.5,1,1,1,1,1,1,0,NULL,"SR")
```

---

rMTS *Function to generate random variates of MTS distribution*

Description

Generates n random numbers distributed according to the modified tempered stable (MTS) distribution.

Usage

```
rMTS(
  n,
  alpha = NULL,
  delta = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  methodR = "SR",
  k = 10000
)
```

Arguments

`n`	sample size (integer).
`alpha`	Stability parameter. A real number between 0 and 2.
`delta`	Scale parameter. A real number > 0.
`lambdap`, `lambdam`	Tempering parameter. A real number > 0.
`mu`	A location parameter, any real number.
`theta`	Parameters stacked as a vector.
`methodR`	A String. Either "TM", "AR" or "SR".
`k`	integer: the level of truncation, if `methodR == "SR"`. 10000 by default.

Details

Currently, random variants can only be generated using the series representation given by Bianchi et al. (2011).

It is recommended to check the generated random numbers once for each distribution using the density function. If the random numbers are shifted, e.g. for the method "SR", it may be worthwhile to increase k.

Value

Generates `n` random numbers of the CTS distribution.

References

Bianchi, M. L.; Rachev, S. T.; Kim, Y. S. & Fabozzi, F. J. (2011), 'Tempered infinitely divisible distributions and processes' doi:10.1137/S0040585X97984632

Examples

```
rMTS(2,0.5,1,1,1,0,NULL,"SR")
```

`rNTS` *Function to generate random variates of NTS distribution.*

Description

Generates `n` random numbers distributed according of the normal tempered stable distribution.

Usage

```
rNTS(
  n,
  alpha = NULL,
  beta = NULL,
  delta = NULL,
  lambda = NULL,
  mu = NULL,
  theta = NULL,
  methodR = "TM",
  k = 10000
)
```

Arguments

n	sample size (integer).
alpha	A real number between 0 and 1.
beta	A gap holder.
delta	A real number > 0.
lambda	A real number > 0.
mu	A location parameter, any real number.
theta	A vector of all other arguments.
methodR	A String. Either "TM","AR" or "SR". "TM" by default.
k	integer: the number of replications, if methodR == "SR". 10000 by default.

Details

theta denotes the parameter vector (alpha, beta, delta, lambda,mu). Either provide the parameters individually OR provide theta. Works by a normal variance-mean mixture with a TSS distribution. Method parameter is for the method of simulating the TSS random variable, see the rTSS() function. "AR" stands for the Acceptance-Rejection Method and "SR" for a truncated infinite shot noise series representation. "TM" stands for Two Methods as two different methods are used depending on which will be faster. In this method the function copula::retstable() is called. "TM" is the standard method used. For more details, see references.

It is recommended to check the generated random numbers once for each distribution using the density function. If the random numbers are shifted, e.g. for the method "SR", it may be worthwhile to increase k.

For more details, see references.

Value

Generates n random numbers.

References

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'

Kawai, R & Masuda, H (2011), 'On simulation of tempered stable random variates' doi:10.1016/j.cam.2010.12.014

Hofert, M (2011), 'Sampling Exponentially Tilted Stable Distributions' doi:10.1145/2043635.2043638

See Also

See also the `rTSS()` function. `copula::retstable()` as "TM" uses this function.

Examples

```
rNTS(100, 0.5, 1,1,1,1)
rNTS(10, 0.6, 0,1,1,0)
rNTS(10, 0.5, 1,1,1,1, NULL, "SR", 100)
```

`rRDTS` *Function to generate random variates of RDTS distribution.*

Description

Generates `n` random numbers distributed according to the rapidly decreasing tempered stable (CTS) distribution.

Usage

```
rRDTS(
  n,
  alpha = NULL,
  delta = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  methodR = "SR",
  k = 10000
)
```

Arguments

`n`	sample size (integer).
`alpha`	Stability parameter. A real number between 0 and 2.
`delta`	Scale parameter for the left tail. A real number > 0.
`lambdap`	Tempering parameter for the right tail. A real number > 0.

lambdam	Tempering parameter for the left tail. A real number > 0.
mu	A location parameter, any real number.
theta	Parameters stacked as a vector.
methodR	A String. Only "SR" works currently.
k	integer: the level of truncation, if methodR == "SR". 10000 by default.

Details

theta denotes the parameter vector (alpha, delta,lambdap, lambdam, mu). Either provide the parameters individually OR provide theta. "SR" stands for a truncated infinite shot noise series representation. Kim et al. (2010) showed how to simulate random variates with SR-method for the RDTS distribution. For more details, see references.

It is recommended to check the generated random numbers once for each distribution using the density function. If the random numbers are shifted, e.g. for the method "SR", it may be worthwhile to increase k.

Value

Generates n random numbers of the RDTS distribution.

References

Kim, Young Shi & Rachev, Svetlozar T. & Leonardo Bianchi, Michele & Fabozzi, Frank J. (2010), 'Tempered stable and tempered infinitely divisible GARCH models' doi:10.1016/j.jbankfin.2010.01.015

Examples

```
rCTS(10,0.5,1,1,1,1,1,NULL,"SR",10)
rCTS(10,0.5,1,1,1,1,1,NULL,"aAR")
```

---

rTSS — *Function to generate random variates of the TSS distribution.*

Description

Generates n random numbers distributed according of the tempered stable subordinator distribution.

Usage

```
rTSS(
  n,
  alpha = NULL,
  delta = NULL,
  lambda = NULL,
  theta = NULL,
  methodR = "TM",
  k = 10000
)
```

Arguments

`n`	sample size (integer).
`alpha`	Stability parameter. A real number between 0 and 1.
`delta`	Scale parameter. A real number > 0.
`lambda`	Tempering parameter. A real number > 0.
`theta`	Parameters stacked as a vector.
`methodR`	A String. Either "TM", "AR" or "SR".
`k`	integer: the level of truncation, if `methodR == "SR"`. 10000 by default.

Details

`theta` denotes the parameter vector `(alpha, delta, lambda)`. Either provide the parameters `alpha`, `delta`, `lambda` individually OR provide `theta`. "AR" stands for the Acceptance-Rejection Method and "SR" for a truncated infinite shot noise series representation. "TM" stands for Two Methods as two different methods are used depending on which will be faster. In this method the function `copula::retstable()` is called. "TM" is the standard method used.

It is recommended to check the generated random numbers once for each distribution using the density function. If the random numbers are shifted, e.g. for the method "SR", it may be worthwhile to increase k.

For more details, see references.

Value

Generates `n` random numbers.

References

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'

Kawai, R & Masuda, H (2011), 'On simulation of tempered stable random variates' doi:10.1016/j.cam.2010.12.014

Hofert, M (2011), 'Sampling Exponentially Tilted Stable Distributions' doi:10.1145/2043635.2043638

See Also

`copula::retstable()` as "TM" uses this function.

Examples

```
rTSS(100,0.5,1,1)
rTSS(100,0.5,1,1,NULL,"SR",50)
```

TemperedEstim *Estimation function*

Description

Main estimation function for the tempered stabled distributions offered within this package. It allows the user to select the preferred estimation method and several related options.

Usage

```
TemperedEstim(
  TemperedType = c("CTS", "TSS", "NTS", "MTS", "GTS", "KRTS", "RDTS"),
  EstimMethod = c("ML", "GMM", "Cgmm", "GMC"),
  data,
  theta0 = NULL,
  ComputeCov = FALSE,
  HandleError = TRUE,
  eps = 1e-06,
  algo = NULL,
  regularization = NULL,
  WeightingMatrix = NULL,
  t_scheme = NULL,
  alphaReg = NULL,
  t_free = NULL,
  nb_t = NULL,
  subdivisions = NULL,
  IntegrationMethod = NULL,
  randomIntegrationLaw = NULL,
  s_min = NULL,
  s_max = NULL,
  ncond = NULL,
  IterationControl = NULL,
  ...
)
```

Arguments

`TemperedType`	A String. Either "CTS", "TSS", "NTS", "MTS", "GTS", "KRTS", "RDTS".
`EstimMethod`	A String. Either "ML", "GMM", "Cgmm", or "GMC".
`data`	Data used to perform the estimation: numeric vector of length n.
`theta0`	A vector of numeric values corresponding to the pattern of the `TemperedType`.
`ComputeCov`	Logical flag: If set to TRUE, the asymptotic covariance matrix is computed. `FALSE` by default.
`HandleError`	Logical flag: If set to `TRUE` and if an error occurs during the estimation procedure, the computation will carry on and NA will be returned. Useful for Monte Carlo simulations.`TRUE` by default.

eps — Numerical error tolerance. 1e-06 by default.

algo — algorithm: For GMM: "2SGMM" is the two step GMM proposed by Hansen (1982). "CueGMM" and "ITGMM" are respectively the continuous updated and the iterative GMM proposed by Hansen, Eaton et Yaron (1996) and adapted to the continuum case. For GMC: "2SGMC", "CueGMC". For Cgmm: "2SCgmm", "CueCgmm", ...

regularization — regularization scheme to be used for moment methods, one of "Tikhonov" (Tikhonov), "LF" (Landweber-Fridmann) and "cut-off" (spectral cut-off).

WeightingMatrix — type of weighting matrix used to compute the objective function for the GMM and GMC methods, one of "OptAsym" (the optimal asymptotic), "DataVar" (the data driven, only for GMM) and "Id" (the identity matrix).

t_scheme — scheme used to select the points for the GMM method where the moment conditions are evaluated, one of "equally" (equally placed), "NonOptAr" (non optimal arithmetic placement), "uniformOpt" (uniform optimal placement), "ArithOpt" (arithmetic optimal placement), "Var Opt" (optimal variance placement) and "free" (users need to pass their own set of points in ...).

alphaReg — value of the regularisation parameter; numeric. Example Value could be ==0.01.

t_free — sequence, if t_scheme=="free".

nb_t — integer, if you set t_scheme <- "equally". nb_t could be == 20 for example.

subdivisions — Number of subdivisions used to compute the different integrals involved in the computation of the objective function for the Cgmm method (to minimise); numeric.

IntegrationMethod — Numerical integration method to be used to approximate the (vectorial) integrals for the Cgmm method. Users can choose between "Uniform" discretization or the "Simpson"'s rule (the 3-point Newton-Cotes quadrature rule).

randomIntegrationLaw — Probability measure associated to the Hilbert space spanned by the moment conditions for the Cgmm method.

s_min, s_max — Lower and Upper bounds of the interval where the moment conditions are considered for the Cgmm method; numeric.

ncond — Integer. Number of moment conditions (until order ncond) for the GMC method. Must not be less than 3 for TSS, 6 for CTS, 5 for NTS.

IterationControl — only used if algo = "IT..." or algo = "Cue..." to control the iterations. See Details.

... — Other arguments to be passed to the estimation function or the asymptotic confidence level.

Details

TemperedType Detailed documentation of the individual tempered stable distributions can be viewed in the respective characteristic function. With the parameter 'TemperedTyp' you can choose the tempered stable distribution you want to use. Here is a list of distribution you can choose from:

TSS Tempered stabel subordinator: See charTSS() for details.

CTS Classical tempered stable distribution: See `charCTS()` for details.

GTS Generalized classical tempered stable distribution: See `charGTS()` for details.

NTS Normal tempered stable distribution: See `charNTS()` for details.

MTS Modified tempered stable distribution: See `charMTS()` for details.

RDTS Rapid decreasing tempered stable distribution: See `charRDTS()` for details.

KRTS Kim-Rachev tempered stable distribution: See `charKRTS()` for details.

Estimfct Additional parameters are needed for different estimation functions. These are listed below for each function. The list of additional parameters starts after the parameter `eps` in the parameter list.

For ML: See usage of Maximum likelihood estimation in Kim et al. (2008). No additional parameters are needed.

For GMM: Generalized Method of Moments by Feuerverger (1981). The parameters `algo`, `alphaReg`, `regularization`, `WeightingMatrix`, and `t_scheme` must be specified.

Parameter `t_scheme`: One of the most important features of this method is that it allows the user to choose how to place the points where the moment conditions are evaluated. One can choose among 6 different options. Depending on the option, further parameters have to be passed.

"equally": equally placed points in `min_t,max_t`. When provided, user's `min_t` and `max_t` will be used (when `Coinstrained == FALSE`).

"NonOptAr": non optimal arithmetic placement.

"uniformOpt": uniform optimal placement.

"ArithOpt": arithmetic optimal placement.

"Var Opt": optimal variance placement as explained above.

"free": user needs to pass own set of points in `t_free`.

Parameter `WeightingMatrix`: One can choose among 3 different options:

"OptAsym": the optimal asymptotic choice.

"DataVar": the covariance matrix of the data provided.

"Id": the identity matrix.

For Cgmm: Continuum Generalized Methods of Moments by Carrasco & Kotchoni (2017). The parameters `algo`, `alphaReg`, `subdivisions`, `IntegrationMethod`, `randomIntegrationLaw`, `s_min`, and `s_max` must be specified.

For GMC: Generalized Method of Cumulants (GMC) by Massing, T. (2022). The parameters `algo`, `alphaReg`, `regularization`, `WeightingMatrix`, and `ncond` must be specified.

Estim-Class Class storing all the information about the estimation method; output of this function.

Slots of the return class

par: Object of class "`numeric`"; Value of the estimated parameters.

par0: Object of class "`numeric`"; Initial guess for the parameters.

vcov: Object of class "`matrix`" representing the covariance matrix.

confint: Object of class "`matrix`" representing the confidence interval computed at a specific level (attribute of the object).

data: Object of class "`numeric`" used to compute the estimation.

sampleSize: Object of class "`numeric`" ; length of the data.

others: Object of class "`list`" ; more information about the estimation method.

duration: Object of class "`numeric`" ; duration in seconds.

failure: Object of class "`numeric`" representing the status of the procedure: 0 failure or 1 success.

method: Object of class "`character`" description of the parameter used in the estimation.

IterationControl If `algo = "IT..."` or `algo = "Cue..."` the user can control each iteration by setting up the list IterationControl which contains the following elements:

NbIter maximum number of iteration. The loop stops when NBIter is reached; default = 10.

PrintIterlogical if set to TRUE, the value of the current parameter estimation is printed to the screen at each iteration; default = TRUE.

RelativeErrMax the loop stops if the relative error between two consecutive estimation steps is smaller than RelativeErrMax; default = 1e-3.

Since this package is structurally based on the **"StableEstim" package by Tarak Kharrat and Georgi N. Boshnakov**, more detailed documentation can be found in their documentation.

Value

Object of a estim-class. See details for more information.

References

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'

Kim, Y. s., Rachev, S. T., Bianchi, M. L. & Fabozzi, F. J. (2008), 'Financial market models with levy processes and time-varying volatility' doi:10.1016/j.jbankfin.2007.11.004

Hansen, L. P. (1982), 'Large sample properties of generalized method of moments estimators' doi:10.2307/1912775

Hansen, L. P.; Heaton, J. & Yaron, A. (1996), 'Finite-Sample Properties of Some Alternative GMM Estimators' doi:10.1080/07350015.1996.10524656

Carrasco, M. & Kotchoni, R. (2017), 'Efficient estimation using the characteristic function' doi:10.1017/S0266466616000025

Kuechler, U. & Tappe, S. (2013), 'Tempered stable distribution and processes' doi:10.1016/j.spa.2013.06.012

Feuerverger, A. & McDunnough, P. (1981), 'On the efficiency of empirical characteristic function procedures' doi:10.1111/j.25176161.1981.tb01143.x

See Also

`https://github.com/GeoBosh/StableEstim/blob/master/R/Simulation.R`

Examples

```
TemperedEstim(TemperedType = "CTS", EstimMethod = "ML",
              data = rCTS(2,1.5,1,1,1,1,0),
              theta0 = c(1.5,1,1,1,1,0) - 0.1);
TemperedEstim("TSS", "GMM", rTSS(20,0.5,1,1), algo = "2SGMM",
             alphaReg = 0.01, regularization = "cut-off",
             WeightingMatrix = "OptAsym", t_scheme = "free",
             t_free = seq(0.1,2,length.out = 12));
TemperedEstim("NTS", "Cgmm", rNTS(20,0.5,1,1,1,0), algo = "2SCgmm",
             alphaReg = 0.01, subdivisions = 50,
             IntegrationMethod = "Uniform", randomIntegrationLaw = "unif",
             s_min = 0, s_max= 1);
TemperedEstim("TSS", "GMC", rTSS(20, 0.5, 1, 1), algo = "2SGMC",
             alphaReg = 0.01, WeightingMatrix = "OptAsym",
             regularization = "cut-off", ncond = 8, theta0 = c(0.5,1,1));
```

TemperedEstim_Simulation

Monte Carlo Simulation

Description

Runs Monte Carlo simulation for a selected estimation method. The function can save results in a file.

Usage

```
TemperedEstim_Simulation(
  ParameterMatrix,
  SampleSizes = c(200, 1600),
  MCparam = 100,
  TemperedType = c("CTS", "TSS", "NTS", "MTS", "GTS", "KRTS", "RDTS"),
  Estimfct = c("ML", "GMM", "Cgmm", "GMC"),
  HandleError = TRUE,
  saveOutput = FALSE,
  SeedOptions = NULL,
  eps = 1e-06,
  algo = NULL,
  regularization = NULL,
  WeightingMatrix = NULL,
  t_scheme = NULL,
  alphaReg = NULL,
  t_free = NULL,
  nb_t = NULL,
  subdivisions = NULL,
  IntegrationMethod = NULL,
  randomIntegrationLaw = NULL,
```

```
  s_min = NULL,
  s_max = NULL,
  ncond = NULL,
  IterationControl = NULL,
  methodR = "TM",
  ...
)
```

Arguments

ParameterMatrix
: The matrix is to be composed of vectors, row by row. Each vector must fit the pattern of theta of the `TemperedType`.

SampleSizes
: Sample sizes to be used to simulate the data. By default, we use 200 (small sample size) and 1600 (large sample size); vector of integer.

MCparam
: Number of Monte Carlo simulation for each couple of parameter, default=100; integer

TemperedType
: A String. Either "CTS", "TSS", "NTS", "MTS", "GTS", "KRTS", "RDTS".

Estimfct
: The estimation function to be used. A String. Either "ML", "GMM", "Cgmm", or "GMC".

HandleError
: Logical flag: if set to TRUE, the simulation doesn't stop when an error in the estimation function is encountered. A vector of (size 4) NA is saved and the the simulation carries on. See details.

saveOutput
: Logical flag: if set to TRUE, a csv file (for each couple of parameter) with the the estimation information is saved in the current directory. See details.

SeedOptions
: List to control the seed generation. See details.

eps
: Numerical error tolerance. `1e-06` by default.

algo
: algorithm: For GMM: `"2SGMM"` is the two step GMM proposed by Hansen (1982). `"CueGMM"` and `"ITGMM"` are respectively the continuous updated and the iterative GMM proposed by Hansen, Eaton et Yaron (1996) and adapted to the continuum case. For GMC: `"2SGMC"`, `"CueGMC"`. For Cgmm: `"2SCgmm"`, `"CueCgmm"`, ...

regularization
: regularization scheme to be used for moment methods, one of `"Tikhonov"` (Tikhonov), `"LF"` (Landweber-Fridmann) and `"cut-off"` (spectral cut-off).

WeightingMatrix
: type of weighting matrix used to compute the objective function for the GMM and GMC methods, one of `"OptAsym"` (the optimal asymptotic), `"DataVar"` (the data driven, only for GMM) and `"Id"` (the identity matrix).

t_scheme
: scheme used to select the points for the GMM method where the moment conditions are evaluated, one of `"equally"` (equally placed), `"NonOptAr"` (non optimal arithmetic placement), `"uniformOpt"` (uniform optimal placement), `"ArithOpt"` (arithmetic optimal placement) , `"Var Opt"` (optimal variance placement) and `"free"` (users need to pass their own set of points in ...).

alphaReg
: value of the regularisation parameter; numeric. Example Value could be ==0.01.

t_free
: sequence, if `t_scheme=="free"`.

`nb_t` integer, if you set `t_scheme <- "equally"`. nb_t could be == 20 for example.

`subdivisions` Number of subdivisions used to compute the different integrals involved in the computation of the objective function for the Cgmm method (to minimise); numeric.

`IntegrationMethod` Numerical integration method to be used to approximate the (vectorial) integrals for the Cgmm method. Users can choose between "Uniform" discretization or the "Simpson"'s rule (the 3-point Newton-Cotes quadrature rule).

`randomIntegrationLaw` Probability measure associated to the Hilbert space spanned by the moment conditions for the Cgmm method.

`s_min, s_max` Lower and Upper bounds of the interval where the moment conditions are considered for the Cgmm method; numeric.

`ncond` Integer. Number of moment conditions (until order `ncond`) for the GMC method. Must not be less than 3 for TSS, 6 for CTS, 5 for NTS.

`IterationControl` only used if algo = "IT..." or algo = "Cue..." to control the iterations. See Details.

`methodR` A string. Method generates random variates of TS distribution. "TM" by default. Switches automatically if the method is not applicable in this way.

`...` Other arguments to be passed to the estimation function.

Details

TemperedTyp With the parameter 'TemperedTyp' you can choose the tempered stable distribution you want to use. Here is a list of distribution you can choose from:

TSS Tempered stabel subordinator: See `charTSS()` for details.

CTS Classical tempered stable distribution: See `charCTS()` for details.

GTS Generalized classical tempered stable distribution: See `charGTS()` for details.

NTS Normal tempered stable distribution: See `charNTS()` for details.

MTS Modified tempered stable distribution: See `charMTS()` for details.

RDTS Rapid decreasing tempered stable distribution: See `charRDTS()` for details.

KRTS Kim-Rachev tempered stable distribution: See `charKRTS()` for details.

Error Handling It is advisable to set it to TRUE when user is planning to launch long simulations as it will prevent the procedure to stop if an error occurs for one sample data. The estimation function will produce a vector of NA as estimated parameters related to this (error generating) sample data and move on to the next Monte Carlo step.

Output file Setting `saveOutput` to `TRUE` will have the side effect of saving a csv file in the working directory. This file will have `MCparam*length(SampleSizes)` lines and its columns will be:

alphaT, ...: the true value of the parameters.

data size: the sample size used to generate the simulated data.

seed: the seed value used to generate the simulated data.

alphaE, ...: the estimate of the parameters.

failure: binary: 0 for success, 1 for failure.

time: estimation running time in seconds.

The file name is informative to let the user identify the value of the true parameters, the MC parameters as well as the options selected for the estimation method. The csv file is updated after each MC estimation which is useful when the simulation stops before it finishes.

SeedOptions If users does not want to control the seed generation, they could ignore this argument (default value NULL). This argument can be more useful when they wants to cut the simulation (even for one parameter value) into pieces. In that case, they can control which part of the seed vector they want to use.

MCtot: total values of MC simulations in the entire process.

seedStart: starting index in the seed vector. The vector extracted will be of size MCparam.

Estimfct Additional parameters are needed for different estimation functions. These are listed below for each function. The list of additional parameters starts after the parameter `eps` in the parameter list.

For ML: See usage of Maximum likelihood estimation in Kim et al. (2008).No additional parameters are needed.

For GMM: Generalized Method of Moments by Feuerverger (1981). The parameters `algo`, `alphaReg`, `regularization`, `WeightingMatrix`, and `t_scheme` must be specified.

Parameter `t_scheme`: One of the most important features of this method is that it allows the user to choose how to place the points where the moment conditions are evaluated. One can choose among 6 different options. Depending on the option, further parameters have to be passed.

"equally": equally placed points in `min_t,max_t`. When provided, user's `min_t` and `max_t` will be used (when `Coinstrained == FALSE`).

"NonOptAr": non optimal arithmetic placement.

"uniformOpt": uniform optimal placement.

"ArithOpt": arithmetic optimal placement.

"Var Opt": optimal variance placement as explained above.

"free": user needs to pass own set of points in `t_free`.

Parameter `WeightingMatrix`: One can choose among 3 different options:

"OptAsym": the optimal asymptotic choice.

"DataVar": the covariance matrix of the data provided.

"Id": the identity matrix.

For Cgmm: Continuum Generalized Methods of Moments by Carrasco & Kotchoni (2017). The parameters `algo`, `alphaReg`, `subdivisions`, `IntegrationMethod`, `randomIntegrationLaw`, `s_min`, and `s_max` must be specified.

For GMC: Generalized Method of Cumulants (GMC) by Massing, T. (2022). The parameters `algo`, `alphaReg`, `regularization`, `WeightingMatrix`, and `ncond` must be specified.

IterationControl If `algo = "IT..."` or `algo = "Cue..."` the user can control each iteration by setting up the list IterationControl which contains the following elements:

NbIter maximum number of iteration. The loop stops when NBIter is reached; default = 10.

PrintIterlogical if set to TRUE, the value of the current parameter estimation is printed to the screen at each iteration; default = TRUE.

RelativeErrMax the loop stops if the relative error between two consecutive estimation steps is smaller than RelativeErrMax; default = 1e-3.

methodR Random numbers must be generated for each MC study. For each distribution, different methods are available for this (partly also depending on alpha). For more information, the documentation of the respective `r...()` distribution can be called up. By default, the fastest method is selected. Since the deviation error can amplify to the edges of alpha depending on the method, it is recommended to check the generated random numbers once for each distribution using the density function before starting the simulation.

Parallelization Parallelization of the function is possible with using `parallelizeMCsimulation()`. If someone wants to parallelize the function manually, the parameter `MCparam` must be set to 1 and the parameter `SeedOption` must be changed for each iteration.

Since this package is structurally based on the **"StableEstim" package by Tarak Kharrat and Georgi N. Boshnakov**, more detailed documentation can be found in their documentation.

Value

If `saveOutput` == `FALSE`, the return object is a list of 2. Results of the simulation are listed in `$outputMat`. If `saveOutput` == `TRUE`, only a csv file is saved and nothing is returned.

References

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'

Kim, Y. s.; Rachev, S. T.; Bianchi, M. L. & Fabozzi, F. J. (2008), 'Financial market models with lévy processes and time-varying volatility' doi:10.1016/j.jbankfin.2007.11.004

Hansen, L. P. (1982), 'Large sample properties of generalized method of moments estimators' doi:10.2307/1912775

Hansen, L. P.; Heaton, J. & Yaron, A. (1996), 'Finite-Sample Properties of Some Alternative GMM Estimators' doi:10.1080/07350015.1996.10524656

Feuerverger, A. & McDunnough, P. (1981), 'On the efficiency of empirical characteristic function procedures' doi:10.1111/j.25176161.1981.tb01143.x

Carrasco, M. & Kotchoni, R. (2017), 'Efficient estimation using the characteristic function' doi:10.1017/S0266466616000025;

Kuechler, U. & Tappe, S. (2013), 'Tempered stable distribution and processes' doi:10.1016/j.spa.2013.06.012

See Also

https://github.com/GeoBosh/StableEstim/blob/master/R/Simulation.R

Examples

```
TemperedEstim_Simulation(ParameterMatrix = rbind(c(1.5,1,1,1,1,0),
                                                 c(0.5,1,1,1,1,0)),
                         SampleSizes = c(4), MCparam = 4,
                         TemperedType = "CTS", Estimfct = "ML",
```

```
                              saveOutput = FALSE)

TemperedEstim_Simulation(ParameterMatrix = rbind(c(1.5,1,1,1,1,0)),
                              SampleSizes = c(4), MCparam = 4,
                              TemperedType = "CTS", Estimfct = "GMM",
                              saveOutput = FALSE, algo = "2SGMM",
                              regularization = "cut-off",
                              WeightingMatrix = "OptAsym", t_scheme = "free",
                              alphaReg = 0.01,
                              t_free = seq(0.1,2,length.out=12))

TemperedEstim_Simulation(ParameterMatrix = rbind(c(1.45,0.55,1,1,1,0)),
                              SampleSizes = c(4), MCparam = 4,
                              TemperedType = "CTS", Estimfct = "Cgmm",
                              saveOutput = FALSE, algo = "2SCgmm",
                              alphaReg = 0.01, subdivisions = 50,
                              IntegrationMethod = "Uniform",
                              randomIntegrationLaw = "unif",
                              s_min = 0, s_max= 1)

TemperedEstim_Simulation(ParameterMatrix = rbind(c(1.45,0.55,1,1,1,0)),
                              SampleSizes = c(4), MCparam = 4,
                              TemperedType = "CTS", Estimfct = "GMC",
                              saveOutput = FALSE, algo = "2SGMC",
                              alphaReg = 0.01, WeightingMatrix = "OptAsym",
                              regularization = "cut-off", ncond = 8)
```

`TempStable` *TempStable: A collection of methods to estimate parameters of different tempered stable distributions.*

Description

A collection of methods to estimate parameters of different tempered stable distributions. Currently, there are three different tempered stable distributions to choose from: Tempered stable subordinator distribution, classical tempered stable distribution, normal tempered stable distribution. The package also provides functions to compute characteristic functions and tools to run Monte Carlo simulations.

Details

The package was developed by Till Massing and Cedric Juessen and is structurally based on the "StableEstim" package by Tarak Kharrat and Georgi N. Boshnakov.

Brief description of functions

TemperedEstim() `TemperedEstim()`computes all the information about the estimator. It allows the user to choose the preferred method and several related options.

Characteristic function, density function, probability function and other functions for every tempered stable distribution mentioned above. E.g. charTSS(), dCTS(), ...

Monte Carlo simulation: a tool to run a Monte Carlo simulation TemperedEstim_Simulation() is provided and can save output files or produce statistical summary.To parallelize this function, you can use parallelizeMCsimulation().

Examples

```
## basic example code
# Such a simulation can take a very long time. Therefore, it can make sense
# to parallelize after Monte Carlo runs. Parallelization of the simulation is
# now possible with [parallelizeMCsimulation()].

# For testing purposes, the amount of runs and parameters is greatly reduced.
# Therefore, the result is not meaningful. To start a meaningful simulation,
# the SampleSize could be, for example, 1000 and MCParam also 1000.

thetaT <- c(1.5,1,1,1,1,0)
res_CTS_ML_size4 <- TemperedEstim_Simulation(ParameterMatrix =
                                               rbind(thetaT),
                                             SampleSizes = c(4),
                                             MCparam = 4,
                                             TemperedType = "CTS",
                                             Estimfct = "ML",
                                             saveOutput = FALSE)

colMeans(sweep(res_CTS_ML_size4$outputMat[,9:14],2,thetaT), na.rm = TRUE)
```

Index