

Package ‘TempStable’

March 27, 2023

Type Package

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

Version 0.1.1

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 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](https://arxiv.org/abs/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:

Carrasco (2017) <[doi:10.1017/S0266466616000025](https://doi.org/10.1017/S0266466616000025)>

Feuerverger (1981) <[doi:10.1111/j.2517-6161.1981.tb01143.x](https://doi.org/10.1111/j.2517-6161.1981.tb01143.x)>

Hansen et al. (1996) <[doi:10.1080/07350015.1996.10524656](https://doi.org/10.1080/07350015.1996.10524656)>

Hansen (1982) <[doi:10.2307/1912775](https://doi.org/10.2307/1912775)>

Kawai & Masuda (2011) <[doi:10.1016/j.cam.2010.12.014](https://doi.org/10.1016/j.cam.2010.12.014)>

Kim et al. (2008) <[doi:10.1016/j.jbankfin.2007.11.004](https://doi.org/10.1016/j.jbankfin.2007.11.004)>

Kuechler & Tappe (2013) <[doi:10.1016/j.spa.2013.06.012](https://doi.org/10.1016/j.spa.2013.06.012)>

Rachev et al. (2011) <[doi:10.1002/9781118268070](https://doi.org/10.1002/9781118268070)>.

URL <https://cedricjuessen.github.io/TempStable/>

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 4.2), methods

Imports doParallel (>= 1.0.12), foreach (>= 1.5.0), 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)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

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

Maintainer Cedric Maximilian Juessen <cedric.juessen@vwl.uni-due.de>

Repository CRAN

Date/Publication 2023-03-27 08:40:06 UTC

R topics documented:

charCTS	2
charNTS	4
charTSS	5
dCTS	6
dNTS	7
dTSS	8
parallelizeMCsimulation	9
pCTS	11
pNTS	12
pTSS	14
qCTS	15
qNTS	16
qTSS	17
rCTS	19
rNTS	20
rTSS	21
TemperedEstim	22
TemperedEstim_Simulation	27
TempStable	31
Index	33

charCTS	<i>Characteristic function of the classical tempered stable (CTS) distribution</i>
---------	--

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
)
```

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.

Details

theta denotes the parameter vector (alpha, deltap, deltam, lambdap, lambdam, mu). Either provide the parameters individually OR provide theta.

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

Value

The CF of the tempered stable subordinator distribution.

References

Massing, T. (2023), 'Parametric Estimation of Tempered Stable Laws'
 Kuechler, U. & Tappe, S. (2013), 'Tempered stable distributions and processes' [doi:10.1016/j.spa.2013.06.012](https://doi.org/10.1016/j.spa.2013.06.012)

Examples

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

charNTS	<i>Characteristic function of the normal tempered stable (NTS) distribution</i>
---------	---

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.

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

Value

The CF of the normal tempered stable distribution.

References

Massing, T. (2022), 'Parametric Estimation of Tempered Stable Laws'
 Rachev, S., Kim, Y., Bianchi, M. & Fabozzi, F. (2011), 'Financial Models with Levy Processes and Volatility Clustering' [doi:10.1002/9781118268070](https://doi.org/10.1002/9781118268070)

Examples

```
x <- seq(-10,10,0.25)
y <- charNTS(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.

$$\varphi_{TSS}(t; \theta) := E_{\theta} [e^{itY}] = \exp(\delta\Gamma(-\alpha)((\lambda - it)^{\alpha} - \lambda^{\alpha}))$$

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](https://doi.org/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](https://doi.org/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 classic 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.

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)
```

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.

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

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](https://doi.org/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	<i>Cumulative probability function of the classic tempered stable (CTS) distribution</i>
------	--

Description

The cumulative probability distribution function (CDF) of the classic 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)
plot(x,y)
```

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)
```

pTSS	<i>Cumulative probability distribution function of the tempered stable subordinator distribution</i>
------	--

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 <code>chartocdf()</code> will be triggered.
N	is a power of two & $N \geq 1024$. if <code>pmethod != "integrate"</code> . 8192 by default. Relevant for
...	Possibility to modify <code>stats::integrate()</code> .

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 classic tempered stable (CTS)

Description

The quantile function of the classic 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 <code>pTSS()</code> and <code>stats::uniroot()</code> .

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 (CTS).

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 <code>==NULL</code> .
...	Modify <code>pNTS()</code> and <code>stats::uniroot()</code> .

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 <code>pTSS()</code> and <code>stats::uniroot()</code> .

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 classic tempered stable (CTS) distribution.

Usage

```
rCTS(
  n,
  alpha = NULL,
  deltap = NULL,
  deltam = NULL,
  lambdap = NULL,
  lambdam = NULL,
  mu = NULL,
  theta = NULL,
  methodR = "AR",
  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 "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. "AR" is the standard method used. 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](https://doi.org/10.1016/j.cam.2010.12.014)

Examples

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

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 = "AR",
  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 "AR" or "SR". "AR" 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. "AR" is the standard method used.

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](https://doi.org/10.1016/j.cam.2010.12.014)

See Also

See also the [rTSS\(\)](#) 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)
```

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 = "AR",
  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 "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. "AR" is the standard method used. 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](https://doi.org/10.1016/j.cam.2010.12.014)

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 stable subordinator distribution (TSS), the classical tempered stable distribution (CTS), and the normal tempered stable distribution (NTS). It allows the user to select the preferred estimation method and several related options.

Usage

```

TemperedEstim(
  TemperedType = c("Classic", "Subordinator", "Normal"),
  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,
  subdivisions = NULL,
  IntegrationMethod = NULL,
  randomIntegrationLaw = NULL,
  s_min = NULL,
  s_max = NULL,
  ncond = NULL,
  IterationControl = NULL,
  ...
)

```

Arguments

TemperedType	A String. Either "Classic", "Subordinator", or "Normal"
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".
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. Use [charTSS\(\)](#), [charCTS\(\)](#), or [charNTS\(\)](#).

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 Feueverger (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 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

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 = "Classic", EstimMethod = "ML",
              data = rCTS(2,1.5,1,1,1,1,0),
              theta0 = c(1.5,1,1,1,1,0) - 0.1);
TemperedEstim("Subordinator", "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("Normal", "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("Subordinator", "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("Classic", "Subordinator", "Normal"),
  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,
  subdivisions = NULL,
  IntegrationMethod = NULL,
  randomIntegrationLaw = NULL,
  s_min = NULL,
  s_max = NULL,
  ncond = NULL,
  IterationControl = NULL,
  methodR = "AR",
  ...
)
```

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 "Classic", "Subordinator", or "Normal".
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 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".
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. It selects the method in rCTS/rNTS/rTSS. "AR" by default.
...	Other arguments to be passed to the estimation function.

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`.

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 = "Classic", Estimfct = "ML",
                          saveOutput = FALSE)

TemperedEstim_Simulation(ParameterMatrix = rbind(c(1.5,1,1,1,1,0)),
                          SampleSizes = c(4), MCparam = 4,
                          TemperedType = "Classic", 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 = "Classic", 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 = "Classic", 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 MCParm also 1000.

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

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


Index

charCTS, [2](#)
charCTS(), [24](#)
charNTS, [4](#)
charNTS(), [24](#)
charTSS, [5](#)
charTSS(), [24, 32](#)

dCTS, [6](#)
dCTS(), [12, 32](#)
dNTS, [7](#)
dNTS(), [13](#)
dTSS, [8](#)
dTSS(), [15](#)

parallelizeMCSimulation, [9](#)
parallelizeMCSimulation(), [30, 32](#)
pCTS, [11](#)
pCTS(), [16](#)
pNTS, [12](#)
pNTS(), [17](#)
pTSS, [14](#)
pTSS(), [16, 18](#)

qCTS, [15](#)
qNTS, [16](#)
qTSS, [17](#)

rCTS, [19](#)
rNTS, [20](#)
rTSS, [21](#)
rTSS(), [21](#)

stats::uniroot(), [16–18](#)

TemperedEstim, [22](#)
TemperedEstim(), [32](#)
TemperedEstim_Simulation, [27](#)
TemperedEstim_Simulation(), [9, 10, 32](#)
TempStable, [31](#)