# Package ‘tggd’

March 25, 2025

**Type** Package

**Title** The Standard Distribution Functions for the Truncated Generalised Gamma Distribution

**Version** 0.1.3

**Date** 2025-03-24

**Maintainer** Aaron Robotham <aaron.robotham@uwa.edu.au>

**Description** Density, distribution function, quantile function and random generation for the Truncated Generalised Gamma Distribution (also in log10(x) and ln(x) space).

**License** GPL-3

**Depends** R (>= 3.00), gsl

**NeedsCompilation** no

**Author** Aaron Robotham [aut, cre], Steven Murray [aut]

**Repository** CRAN

**Date/Publication** 2025-03-25 08:00:02 UTC

## Contents

---

`tggd` *The Truncated Generalised Gamma Distribution*

---

### Description

Density, distribution function, quantile function and random generation for the Truncated Generalised Gamma Distribution in linear space.

## Usage

```
dtggd(x, scale=1e14, a=-1, b=1, xmin=1e10, log = FALSE)
ptggd(q, scale=1e14, a=-1, b=1, xmin=1e10, lower.tail = TRUE, log.p = FALSE)
qtggd(p, scale=1e14, a=-1, b=1, xmin=1e10, lower.tail = TRUE, log.p = FALSE,
res.approx=1e-2)
rtggd(n, scale=1e14, a=-1, b=1, xmin=1e10, res.approx=1e-2)
tggd_mode(scale=1e14, a=-1, b=1, xmin=1e10)
tggd_mean(scale=1e14, a=-1, b=1, xmin=1e10)
tggd_var(scale=1e14, a=-1, b=1, xmin=1e10)
tggd_sd(scale=1e14, a=-1, b=1, xmin=1e10)
```

## Arguments

| | |
|---|---|
| `x, q` | Vector of quantiles. |
| `p` | Vector of probabilities. |
| `n` | Number of observations. If length(n) > 1, the length is taken to be the number required. |
| `scale` | Vector of values for scale, which controls the transition regime between the power law slope and the exponential cut-off of the TGGD. This is analogous to the scale parameter for the standard Gamma distribution (see `GammaDist`). |
| `a` | Vector of values for a, which controls the power law slope of the TGGD. |
| `b` | Vector of values for b, which controls the exponential cutoff of the TGGD. |
| `xmin` | Vector of values for xmin, which controls the lower limit at which to trancate the TGGD. |
| `res.approx` | The resolution used to create the inverted CDF required to map probability integrals back onto quantiles. |
| `log, log.p` | Logical; if TRUE, probabilities/densities p are returned as log(p). |
| `lower.tail` | Logical; if TRUE (default), probabilities are P[X <= x], otherwise, P[X > x]. |

## Details

This distribution function is described in detail in Murray, Robotham and Power 2016. The PDF is given by:

$$f(x;a,b,s,m)=\frac{b(\frac{x}{s})^a\exp(-(\frac{x}{s})^b)}{s\Gamma(\frac{a+1}{b},(\frac{m}{s})^b)}$$

where, from the argument list above, we use x=x, a=a, b=b, s=scale and m=xmin. $\Gamma$ is the upper incomplete Gamma function as defined by the gsl `gamma_inc` function, using the same argument ordering, where `gamma_inc(a,x)==pgamma(x,a,lower=FALSE)*gamma(x)` for a>0. `gamma_inc` is used because it allows for the computation of upper incomplete integrals in cases where a<=0.

## Value

dtggd gives the density, ptggd gives the distribution function, qtggd gives the quantile function, and rtggd generates random deviates. tggd_mode gives the location of the distribution mode. tggd_mean gives the location of the distribution mean. tggd_var gives the value of the distribution variance. tggd_sd gives the value of the distribution standard deviation.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rtggd, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

## Author(s)

Aaron Robotham, Steven Murray

## References

Murray, Robotham and Power (2016)

## See Also

`GammaDist` regarding the Gamma distribution. `RNG` about random number generation in R. `Distributions` for other standard distributions.

## Examples

```
r <- rtggd(100,a=-2)
hist(log10(r))

##Ideally the output below should equal 1, in practice it will be very close:
qtggd(ptggd(r))/r

#These should be the same:
integrate(dtggd,lower=1e10,upper=1e11,a=-1.5,b=0.7,xmin=1e10)
ptggd(1e11,a=-1.5,b=0.7,xmin=1e10)

#This should be very close to 1 (for a true PDF):
ptggd(1e18,a=-1.5,b=0.7,xmin=1e10)

#To show the link to the log10 (called log) and ln variants (and the slight inaccuracies)
#these outputs should be a sequence from 0 to 1 (by=0.1):
ptggd(10^qtggd_log(seq(0,1,by=0.1)))
ptggd(exp(qtggd_ln(seq(0,1,by=0.1))))
```

## tggd_ln *The Truncated Generalised Gamma Distribution*

### Description

Density, distribution function, quantile function and random generation for the Truncated Generalised Gamma Distribution in natural log (ln) space. Specifically, if exp(x) is drawn from a TGGD distribution (in real space), these functions give the distribution of x, using the same parameter values.

### Usage

```
dtggd_ln(x, scale=log(1e14), a=-1, b=1, xmin=log(1e10), log = FALSE)
ptggd_ln(q, scale=log(1e14), a=-1, b=1, xmin=log(1e10), lower.tail = TRUE, log.p = FALSE)
qtggd_ln(p, scale=log(1e14), a=-1, b=1, xmin=log(1e10), lower.tail = TRUE, log.p = FALSE,
res.approx=1e-2)
rtggd_ln(n, scale=log(1e14), a=-1, b=1, xmin=log(1e10), res.approx=1e-2)
tggd_mode_ln(scale=log(1e14), a=-1, b=1, xmin=log(1e10))
```

### Arguments

| | |
|---|---|
| `x, q` | Vector of quantiles. |
| `p` | Vector of probabilities. |
| `n` | Number of observations. If length(n) > 1, the length is taken to be the number required. |
| `scale` | Vector of values for scale, which controls the transition regime between the power law slope and the exponential cut-off of the TGGD. This is analogous to the scale parameter for the standard Gamma distribution (see `GammaDist`). |
| `a` | Vector of values for a, which controls the power law slope of the TGGD. |
| `b` | Vector of values for b, which controls the exponential cutoff of the TGGD. |
| `xmin` | Vector of values for xmin, which controls the lower limit at which to trancate the TGGD. |
| `res.approx` | The resolution used to create the inverted CDF required to map probability integrals back onto quantiles. |
| `log, log.p` | Logical; if TRUE, probabilities/densities p are returned as log(p). |
| `lower.tail` | Logical; if TRUE (default), probabilities are P[X <= x], otherwise, P[X > x]. |

### Details

This distribution function is described in detail in Murray, Robotham and Power 2016.

$$f(x;a,b,s,m) = \frac{b\exp((x-s)^{(a+1)})\exp(-\exp(b(x-s)))}{s\Gamma(\frac{a+1}{b},\exp(m-s)^{b})}$$

where, from the argument list above, we use x=x, a=a, b=b, s=scale and m=xmin. $\Gamma$ is the upper incomplete Gamma function as defined by the gsl `gamma_inc` function, using the same argument ordering, where `gamma_inc(a,x)==pgamma(x,a,lower=FALSE)*gamma(x)` for a>0. `gamma_inc` is used because it allows for the computation of upper incomplete integrals in cases where a<=0.

## Value

dtggd_ln gives the density, ptggd_ln gives the distribution function, qtggd_ln gives the quantile function, and rtggd_ln generates random deviates. tggd_mode_ln gives the location of the distribution mode.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rtggd_ln, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

## Note

The intended application of the log-space version of the TGGD is to provide the correct distribution when variates are drawn from a real-space TGGD, but there are priors on their uncertainty which operate in logspace (eg. a lognormal distribution). The likelihood of a given set of parameters is incorrect in such a case if the real-space version is used without an adjustment to the Jacobian.

## Author(s)

Aaron Robotham, Steven Murray

## References

Murray, Robotham and Power (2016)

## See Also

`GammaDist` regarding the Gamma distribution. `RNG` about random number generation in R. `Distributions` for other standard distributions.

## Examples

```
r <- rtggd_ln(100,a=-2)
hist(r)

##Ideally the output below should equal 0, in practice it will be very close:
qtggd_ln(ptggd_ln(r))-r

#These should be the same:
integrate(dtggd_ln,lower=log(1e10),upper=log(1e11),a=-1.5,b=0.7,xmin=log(1e10))
ptggd_ln(log(1e11),a=-1.5,b=0.7,xmin=log(1e10))

#This should be very close to 1 (for a true PDF):
ptggd_ln(log(1e18),a=-1.5,b=0.7,xmin=log(1e10))
```

```
#To show the link to the linear and log10 (called log) variants (and the slight
#inaccuracies) these outputs should be a sequence from 0 to 1 (by=0.1):
ptggd_log(log10(qtggd(seq(0,1,by=0.1))))
ptggd_log(qtggd_ln(seq(0,1,by=0.1))/log(10))
```

## tggd_log *The Truncated Generalised Gamma Distribution*

### Description

Density, distribution function, quantile function and random generation for the Truncated Generalised Gamma Distribution in log base 10 (log10) space. Specifically, if 10^x is drawn from a TGGD distribution (in real space), these functions give the distribution of x, using the same parameter values.

### Usage

```
dtggd_log(x, scale=14, a=-1, b=1, xmin=10, log = FALSE)
ptggd_log(q, scale=14, a=-1, b=1, xmin=10, lower.tail = TRUE, log.p = FALSE)
qtggd_log(p, scale=14, a=-1, b=1, xmin=10, lower.tail = TRUE, log.p = FALSE,
res.approx=1e-2)
rtggd_log(n, scale=14, a=-1, b=1, xmin=10, res.approx=1e-2)
tggd_mode_log(scale=14, a=-1, b=1, xmin=10)
```

### Arguments

| | |
|---|---|
| `x, q` | Vector of quantiles. |
| `p` | Vector of probabilities. |
| `n` | Number of observations. If length(n) > 1, the length is taken to be the number required. |
| `scale` | Vector of values for scale, which controls the transition regime between the power law slope and the exponential cut-off of the TGGD. This is analogous to the scale parameter for the standard Gamma distribution (see `GammaDist`). |
| `a` | Vector of values for a, which controls the power law slope of the TGGD. |
| `b` | Vector of values for b, which controls the exponential cutoff of the TGGD. |
| `xmin` | Vector of values for xmin, which controls the lower limit at which to trancate the TGGD. |
| `res.approx` | The resolution used to create the inverted CDF required to map probability integrals back onto quantiles. |
| `log, log.p` | Logical; if TRUE, probabilities/densities p are returned as log(p). |
| `lower.tail` | Logical; if TRUE (default), probabilities are P[X <= x], otherwise, P[X > x]. |

## Details

This distribution function is described in detail in Murray, Robotham and Power 2016. The PDF is given by:

$$f(x;a,b,s,m) = \frac{\ln(10).b(10^{(x-s)})^{(a+1)}\exp(-10^{(b(x-s))})}{s\Gamma(\frac{a+1}{b},(10^{(m-s)})^b)}$$

where, from the argument list above, we use x=x, a=a, b=b, s=scale and m=xmin. $\Gamma$ is the upper incomplete Gamma function as defined by the gsl `gamma_inc` function, using the same argument ordering, where `gamma_inc(a,x)==pgamma(x,a,lower=FALSE)*gamma(x)` for a>0. `gamma_inc` is used because it allows for the computation of upper incomplete integrals in cases where a<=0.

## Value

dtggd_log gives the density, ptggd_log gives the distribution function, qtggd_log gives the quantile function, and rtggd_log generates random deviates. tggd_mode_log gives the location of the distribution mode.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rtggd_log, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

## Note

The intended application of the log-space version of the TGGD is to provide the correct distribution when variates are drawn from a real-space TGGD, but there are priors on their uncertainty which operate in logspace (eg. a lognormal distribution). The likelihood of a given set of parameters is incorrect in such a case if the real-space version is used without an adjustment to the Jacobian.

## Author(s)

Aaron Robotham, Steven Murray

## References

Murray, Robotham and Power (2016) Baldry et al, 2012, MNRAS, 421, 621

## See Also

`GammaDist` regarding the Gamma distribution. `RNG` about random number generation in R. `Distributions` for other standard distributions.

## Examples

```
r <- rtggd_log(100,a=-2)
hist(r)

##Ideally the output below should equal 0, in practice it will be very close:
qtggd_log(ptggd_log(r))-r

#These should be the same:
integrate(dtggd_log,lower=10,upper=11,a=-1.5,b=0.7,xmin=10)
ptggd_log(11,a=-1.5,b=0.7,xmin=10)

#This should be very close to 1 (for a true PDF):
ptggd_log(18,a=-1.5,b=0.7,xmin=10)

#To show the link to the linear and ln variants (and the slight inaccuracies) these
#outputs should be a sequence from 0 to 1 (by=0.1):
ptggd_ln(log(qtggd(seq(0,1,by=0.1))))
ptggd_ln(qtggd_log(seq(0,1,by=0.1))*log(10))

#Here we make a double Schechter galaxy stellar mass function down to a target stellar
#mass (xmin) of log10(SM)=8.

#Using data from Baldry (2012):
#Mixture 1 has M* (scale)=10.66, a=-1.47, b=1, phi*=0.79e-3
#Mixture 2 has M* (scale)=10.66, a=-0.35, b=1, phi*=3.96e-3

#phi* is defined such that: dtggd_log(M*,M*,a,b,xmin)=phi*.log(10).exp(-1)
#for any a, b and xmin.

#We want to fit for the ratio of phi*: 0.79/3.96=0.2

#Relatively speaking, we can define new scaling values for sampling with:
M1norm=0.2/dtggd_log(10.66,10.66,-1.47,1,xmin=8)
M2norm=1/dtggd_log(10.66,10.66,-0.35,1,xmin=8)
Mtot=M1norm+M2norm
#Say we want to sample 1e5 galaxies, we can then do:
Nsamp=1e5
set.seed(100)
GalSample=c(rtggd_log(Nsamp*M1norm/Mtot,10.66,-1.47,1,xmin=8),
rtggd_log(Nsamp*M2norm/Mtot,10.66,-0.35,1,xmin=8))
temp=hist(GalSample,breaks=seq(8,12,by=0.1), plot=FALSE)
#We can then make a plot to compare to Fig 13 of Baldry (2012)
#(the lines are approximate, using trapazoid integration for the bins):
plot(temp$mids,temp$counts,log='y')
lines(seq(8,12,by=0.01), dtggd_log(seq(8,12,by=0.01),10.66,-1.47,1,xmin=8)*
Nsamp*M1norm/Mtot/10,col='blue')
lines(seq(8,12,by=0.01), dtggd_log(seq(8,12,by=0.01),10.66,-0.35,1,xmin=8)*
Nsamp*M2norm/Mtot/10,col='red')
lines(seq(8,12,by=0.01), dtggd_log(seq(8,12,by=0.01),10.66,-1.47,1,xmin=8)*
Nsamp*M1norm/Mtot/10 + dtggd_log(seq(8,12,by=0.01),10.66,-0.35,1,xmin=8)*
Nsamp*M2norm/Mtot/10,col='black')
```

```
## Not run:
#Now we can try to fit the mixed model. The trick here is we fit for the mixture using
#an additional parameter, where one component is multiplied by par[4] and the other
#1-par[4]. We define it so M1norm/Mtot=par[4] and M1norm/Mtot=1-par[4].

mixlike=function(par,data){
return(-sum(log(
dtggd_log(data,par[1],par[2],1,8)*par[4]+     #Contribution of mix 1 to the likelihood
dtggd_log(data,par[1],par[3],1,8)*(1-par[4])  #Contribution of mix 2 to the likelihood
)))
}
GSMFfit=optim(par=c(10,-2,0,0.5), fn=mixlike, data=GalSample, hessian=TRUE)
#The fit is probably not fantastic though. Generalised Gamma distributions (including
#truncated ones) display poor convergence properties using ML. Full MCMC is a better
#route when trying to fit GSMF type data. And the data certainly should *not* be binned!

#The maximum likelihood parameters:
GSMFfit$par
#The marginal errors using the diagonal of the inverse hessian:
sqrt(diag(solve(GSMFfit$hessian)))

#The M1norm/Mtot mixture output is ~0.8.
#To get back to original ratio of phi1*/phi2* (~0.2):

(GSMFfit$par[4]*dtggd_log(10.66,10.66,-1.47,1,xmin=8))/
((1-GSMFfit$par[4])*dtggd_log(10.66,10.66,-0.35,1,xmin=8))

#In general the final phi* will still need a further global normalisation, e.g.
#to count within a set window of stellar mass and volume (redshift and sky area).

## End(Not run)
```

# Index