# Package ‘tsallisqexp’

October 17, 2024

**Type** Package

**Title** Tsallis q-Exp Distribution

**Version** 0.9-5

**Description** Tsallis distribution also known as the q-exponential family distribution. Provide distribution d, p, q, r functions, fitting and testing functions. Project initiated by Paul Higbie and based on Cosma Shalizi's code.

**Depends** R (>= 3.0.0)

**License** GPL (>= 2)

**NeedsCompilation** no

**Author** Christophe Dutang [aut, cre] (<https://orcid.org/0000-0001-6732-1501>), Cosma Shalizi [aut] (<https://orcid.org/0000-0002-9195-1308>)

**Maintainer** Christophe Dutang <dutangc@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-10-16 22:20:02 UTC

## Contents



---

`tsal` *The Tsallis Distribution*

---

### Description

Density function, distribution function, quantile function, random generation.

## Usage

```
dtsal(x, shape=1, scale=1, q=tsal.q.from.shape(shape),
kappa=tsal.kappa.from.ss(shape,scale),
log=FALSE)

ptsal(x, shape=1, scale=1, q=tsal.q.from.shape(shape),
kappa=tsal.kappa.from.ss(shape,scale),
lower.tail=TRUE, log.p=FALSE)

qtsal(p,  shape=1, scale=1, q=tsal.q.from.shape(shape),
kappa=tsal.kappa.from.ss(shape,scale),
lower.tail=TRUE, log.p=FALSE)

rtsal(n, shape=1, scale=1, q=tsal.q.from.shape(shape),
kappa=tsal.kappa.from.ss(shape,scale))

tsal.mean(shape, scale, q=tsal.q.from.shape(shape),
kappa=tsal.kappa.from.ss(shape,scale))
```

## Arguments

| | |
|---|---|
| `x` | vector of quantiles. |
| `q` | vector of quantiles or a shape parameter. |
| `p` | vector of probabilities. |
| `n` | number of observations. If `length(n) > 1`, the length is taken to be the number required. |
| `shape` | shape parameter. |
| `scale`, `kappa` | scale parameters. |
| `log`, `log.p` | logical; if TRUE, probabilities p are given as log(p). |
| `lower.tail` | logical; if TRUE (default), probabilities are $P[X \le x]$, otherwise, $P[X > x]$. |

## Details

The Tsallis distribution is defined by the following density

$$f(x) = \frac{1}{\kappa}(1 - (1 - q)x/\kappa)^{1/(1-q)}$$

for all $x$. It is convenient to introduce a re-parameterization $shape = -1/(1-q)$, $scale = shape * \kappa$ which makes the relationship to the Pareto clearer, and eases estimation. If we have both shape/scale and q/kappa parameters, the latter over-ride.

## Value

`dtsal` gives the density, `ptsal` gives the distribution function, `qtsal` gives the quantile function, and `rtsal` generates random deviates. `tsal.mean` computes the expected value.

The length of the result is determined by `n` for `rtsal`, and is the maximum of the lengths of the numerical parameters for the other functions.

### Author(s)

Cosma Shalizi (original R code), Christophe Dutang (R packaging)

### References

*Maximum Likelihood Estimation for q-Exponential (Tsallis) Distributions*, C. Shalizi, http://bactra.org/research/tsallis-MLE/ and arxiv.org: 0701854.

### Examples

```
#####
# (1) density function
x <- seq(0, 5, length=24)

cbind(x, dtsal(x, 1/2, 1/4))

#####
# (2) distribution function

cbind(x, ptsal(x, 1/2, 1/4))
```

---

`tsal.boot` *Bootstraps methods for Tsallis Distributions*

---

### Description

Bootstrap functions.

### Usage

```
tsal.bootstrap.errors(dist=NULL, reps=500, confidence=0.95,
    n=if(is.null(dist)) 1 else dist$n,
    shape=if(is.null(dist)) 1 else dist$shape,
    scale=if(is.null(dist)) 1 else dist$scale,
    q = if(is.null(dist)) tsal.q.from.shape(shape) else dist$q,
    kappa = if(is.null(dist)) tsal.kappa.from.ss(shape,scale) else dist$kappa,
    method = if(is.null(dist)) "mle.equation" else dist$method,
    xmin = if(is.null(dist)) 0 else dist$xmin)


tsal.total.magnitude(dist=NULL, n=if(is.null(dist)) 1 else dist$n,
    shape=if(is.null(dist)) 1 else dist$shape,
    scale=if(is.null(dist)) 1 else dist$scale,
```

```
q = if(is.null(dist)) tsal.q.from.shape(shape) else dist$q,
kappa = if(is.null(dist)) tsal.kappa.from.ss(shape,scale) else dist$kappa,
xmin = if(is.null(dist)) 0 else dist$xmin,
mult = 1)
```

## Arguments

| | |
|---|---|
| `dist` | distribution (as a list of the sort produced by tsal.fit) |
| `reps` | number of bootstrap replicates. |
| `confidence` | confidence level for confidence intervals. |
| `n` | original sample size. |
| `shape, q` | shape parameters (over-riding those of the distribution, if one was given). |
| `scale, kappa` | scale parameters (over-riding those of the distribution, if one was given). |
| `method` | fitting method (over-riding that used in the original fit, if one was given), see `tsal.fit`. |
| `xmin` | minimum x-value (left-censoring threshold). |
| `mult` | multiplier of size (if the base units of the data are not real units). |

## Details

`tsal.bootstrap.errors` finds biases and standard errors for parameter estimates by parametric bootstrapping, and simple confidence intervals Simulate, many times, drawing samples from the estimated distribution, of the same size as the original data; re-estimate the parameters on the simulated data. The distribution of the re-estimates around the estimated parameters is approximately the same as the distribution of the estimate around the true parameters. This function invokes the estimating-equation MLE, but it would be easy to modify to use other methods. Confidence intervals (CI) are calculated for each parameter separately, using a simple pivotal interval (see, e.g., Wasserman, _All of Statistics_, Section 8.3). Confidence regions for combinations of parameters would be a tedious, but straightforward, extension.

`tsal.total.magnitude` estimates the total magnitude of a tail-sampled population given that we have n samples from the tail of a distribution, i.e., only values >= xmin were retained, provide an estimate of the total magnitude (summed values) of the population. Then it estimates the number of objects, observed and un-observed, as n/pr(X >= xmin) and then multiply by the mean.

## Value

`tsal.bootstrap.errors` returns a structured list, containing the actual parameter settings used, the estimated biases, the estimated standard errors, the lower confidence limits, the upper confidence limits, the sample size, the number of replicates, the confidence level, and the fitting method.

`tsal.total.magnitude` returns a list, giving estimated total magnitude and estimated total population size.

## Author(s)

Cosma Shalizi (original R code), Christophe Dutang (R packaging)

### References

*Maximum Likelihood Estimation for q-Exponential (Tsallis) Distributions*, `http://bactra.org/research/tsallis-MLE/` and `https://arxiv.org/abs/math/0701854`.

### Examples

```
#####
# (1) fit
x <- rtsal(20, 1/2, 1/4)
tsal.loglik(x, 1/2, 1/4)

tsal.fit(x, method="mle.equation")
tsal.fit(x, method="mle.direct")
tsal.fit(x, method="leastsquares")
```

---

## `tsal.fit` *Fitting Tsallis Distributions*

---

### Description

Loglikelihood and fit functions.

### Usage

```
tsal.loglik(x, shape, scale, q=tsal.q.from.shape(shape),
kappa=tsal.kappa.from.ss(shape,scale), xmin=0)

tsal.fit(x, xmin=0, method=c("mle.equation", "mle.direct", "leastsquares"), ...)

#
# Note that this function ONLY works with the shape-scale parameterization
# Inputs: shape, scale, left-censoring threshold

tsal.fisher(shape, scale, xmin=0)
```

### Arguments

| | |
|---|---|
| `x` | vector of quantiles. |
| `shape, q` | shape parameters. |
| `scale, kappa` | scale parameters. |
| `xmin` | minimum x-value. |

| | |
|---|---|
| `method` | A character string for the estimation method: `"mle.equation"` (default), `"mle.direct"`, `"leastsquares"`. |
| `...` | further arguments to be passed to the estimation method. |

## Details

`tsal.loglik` computes the loglikelihood of a sample `x`.

`tsal.fisher` calculates the Fisher information matrix, for asymptotic variances and covariances of the maximum likelihood estimates of shape and scale First row/column corresponds to shape, second to scale Convergence to the asymptotic normal distribution can be slow, so for limited data you should bootstrap.

`tsal.fit` estimates parameters by solving maximum likelihood equations when `method="mle.equation"`, by minimizing the log-likelihood (directly) when `method="mle.direct"`, by minimizing the square difference between the empirical and theoretical distribution functions. This function is a wrapper for the actual methods: `tsal.fit.mle.equation` (solve maximum likelihood estimating equations); `tsal.fit.mle.direct` (numerical likelihood maximization); and `tsal.fit.leastsquares` (least-squares curve-fitting to the empirical distribution); prettying up the results in all cases.

## Value

`tsal.loglik` returns the loglikelihood as a numeric.

`tsal.fit` returns `NA` when estimation aborts or a list with components (`type`, `q`, `kappa`, `shape`, `scale`, `loglik`, `n`, `xmin`, `method`) when estimation succeeds.

## Author(s)

Cosma Shalizi (original R code), Christophe Dutang (R packaging)

## References

*Maximum Likelihood Estimation for q-Exponential (Tsallis) Distributions*, http://bactra.org/research/tsallis-MLE/ and https://arxiv.org/abs/math/0701854.

## Examples

```
#####
# (1) fit
x <- rtsal(20, 1/2, 1/4)
tsal.loglik(x, 1/2, 1/4)

tsal.fit(x, method="mle.equation")
tsal.fit(x, method="mle.direct")
tsal.fit(x, method="leastsquares")
```

`tsal.tail` *The Tsallis Distribution with a censoring parameter (tail-conditional)*

### Description

Density function, distribution function, quantile function, random generation.

### Usage

```
dtsal.tail(x, shape=1,scale=1, q=tsal.q.from.shape(shape),
kappa=tsal.kappa.from.ss(shape,scale), xmin=0,
log=FALSE)

ptsal.tail(x, shape=1, scale=1, q=tsal.q.from.shape(shape),
kappa=tsal.kappa.from.ss(shape,scale), xmin=0,
lower.tail=TRUE, log.p=FALSE)

qtsal.tail(p,  shape=1, scale=1, q=tsal.q.from.shape(shape),
kappa=tsal.kappa.from.ss(shape,scale), xmin=0,
lower.tail=TRUE, log.p=FALSE)

rtsal.tail(n, shape=1, scale=1, q=tsal.q.from.shape(shape),
kappa=tsal.kappa.from.ss(shape,scale), xmin=0)
```

### Arguments

| | |
|---|---|
| `x` | vector of quantiles. |
| `q` | vector of quantiles or a shape parameter. |
| `p` | vector of probabilities. |
| `n` | number of observations. If `length(n) > 1`, the length is taken to be the number required. |
| `shape` | shape parameter. |
| `scale, kappa` | scale parameters. |
| `xmin` | minimum x-value. |
| `log, log.p` | logical; if TRUE, probabilities p are given as log(p). |
| `lower.tail` | logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$. |

### Details

The Tsallis distribution with a censoring parameter is the distribution of a Tsallis distributed random variable conditionnaly on $x > xmin$. The density is defined as

$$f(x) = \frac{C}{\kappa}(1 - (1 - q)x/\kappa)^{1/(1-q)}$$

for all $x > xmin$ where $C$ is the appropriate constant so that the integral of the density equals 1. That is $C$ is the survival probability of the classic Tsallis distribution at $x = xmin$. It is convenient to introduce a re-parameterization $shape = -1/(1-q)$, $scale = shape * \kappa$ which makes the relationship to the Pareto clearer, and eases estimation. If we have both shape/scale and q/kappa parameters, the latter over-ride.

## Value

`dtsal.tail` gives the density, `ptsal.tail` gives the distribution function, `qtsal.tail` gives the quantile function, and `rtsal.tail` generates random deviates.

The length of the result is determined by `n` for `rtsal.tail`, and is the maximum of the lengths of the numerical parameters for the other functions.

## Author(s)

Cosma Shalizi (original R code), Christophe Dutang (R packaging)

## References

*Maximum Likelihood Estimation for q-Exponential (Tsallis) Distributions*, http://bactra.org/research/tsallis-MLE/ and https://arxiv.org/abs/math/0701854.

## Examples

```
#####
# (1) density function
x <- seq(0, 5, length=24)

cbind(x, dtsal(x, 1/2, 1/4))

#####
# (2) distribution function

cbind(x, ptsal(x, 1/2, 1/4))
```

---

`tsal.test` *Test Tsallis Distributions*

---

## Description

Test functions.

### Usage

```
test.tsal.quantile.transform(from=0, to=1e6, shape=1, scale=1,
    q=tsal.q.from.shape(shape), kappa=tsal.kappa.from.ss(shape,scale),
    n=1e5, lwd=0.01, xmin=0, ...)

test.tsal.LR.distribution(n=100, reps=100, shape=1, scale=1,
    q=tsal.q.from.shape(shape), kappa=tsal.kappa.from.ss(shape,scale),
    xmin=0,method="mle.equation",...)
```

### Arguments

| | |
|---|---|
| `from` | lower limit for `x`. |
| `to` | upper limit for `x`. |
| `shape, q` | shape parameters. |
| `scale, kappa` | scale parameters. If we have both shape/scale and q/kappa parameters, the latter over-ride. |
| `n` | number of points at which to evaluate the function over the domain or number of sample points. |
| `lwd` | line width. |
| `xmin` | minimum x-value (left-censoring threshold). |
| `...` | further arguments to be passed to `curve` or `plot`, except `xlim` for `test.tsal.LR.distribution`. |
| `reps` | number of replicates. |
| `method` | estimation method, see `tsal.fit`. |

### Details

`plot.tsal.quantile.transform` check the accuracy of quantile/inverse quantile transformations For all parameter values and all `x`, `qtsal(ptsal(x))` should = `x`. This function displays the relative error in the transformation, which is due to numerical imprecision. This indicates roughly how far off random variates generated by the transformation method will be. If everything is going according to plan, the curve plotted should oscillate extremely rapidly between positive and negative limits which, while growing, stay quite small in absolute terms, e.g., on the order of 1e-5 when `x` is on the order of 1e9.

`plot.tsal.LR.distribution` checks likelihood ratio estimation accuracy : $2*[(Likelihoodatestimatedparameters)-(likelihoodattrueparameters)]$ should have a chi square distribution with 2 degrees of freedom, at least asymptotically for large samples.

### Value

`plot.tsal.quantile.transform` plots the relative error in the transformation.

`plot.tsal.LR.distribution` plots the likelihood ratio against a chi-square distribution and returns the result of Kolmgorov-Smirnov test against theoretical distribution.

## Author(s)

Cosma Shalizi (original R code), Christophe Dutang (R packaging)

## References

*Maximum Likelihood Estimation for q-Exponential (Tsallis) Distributions*, http://bactra.org/research/tsallis-MLE/ and https://arxiv.org/abs/math/0701854.

## Examples

```
#####
# (1) fit
x <- rtsal(20, 1/2, 1/4)
tsal.loglik(x, 1/2, 1/4)

tsal.fit(x, method="mle.equation")
tsal.fit(x, method="mle.direct")
tsal.fit(x, method="leastsquares")
```

# Index