Package 'rriskDistributions'

May 9, 2026

Type Package

Title Fitting Distributions to Given Data or Known Quantiles

Description Collection of functions for fitting distributions to given data or by known quantiles. Two main functions fit.perc() and fit.cont() provide users a GUI that allows to choose a most appropriate distribution without any knowledge of the R syntax. Note, this package is a part of the 'rrisk' project.

License GPL (>= 3)

LazyLoad yes

URL http://www.bfr.bund.de/cd/52158

Depends R(>= 2.11.0)

Imports stats, graphics, mc2d, eha, msm, tcltk, tkrplot

SystemRequirements Tcl/Tk (>= 8.5), Tktable (>= 2.9)

Version 2.1.2

Date 2017-03-22

Encoding UTF-8

Collate 'rriskDistributions-package.R'
'rriskDistributions_functions.R'

RoxygenNote 6.0.1

NeedsCompilation no

Author Natalia Belgorodski [aut] (STAT-UP Statistical Consulting),
Matthias Greiner [aut, cre] (Federal Institute for Risk Assessment, Germany),
Kristin Tolksdorf [aut] (Federal Institute for Risk Assessment, Germany),
Katharina Schueller [aut] (STAT-UP Statistical Consulting),
Matthias Flor [ctb] (Federal Institute for Risk Assessment, Germany),
Lutz Göhring [ctb] (Lutz Göhring Consulting)

Maintainer Matthias Greiner <matthias.greiner@bfr.bund.de>

Repository CRAN

Date/Publication 2017-03-24 13:17:43 UTC

Contents

`rriskDistributions-package`

Fitting distributions to given data or known quantiles

Description

This packages provides a collection of functions for estimation parameters of continuous or discrete distributions (related to the `rrisk` project) to given data or to known quantiles.

Details

This package is a part of the rrisk project and contains functions for fitting distributions to given data or by known quantiles. This package does not depend on the whole rrisk project and can be used separately. The rrisk project can be downloaded from http://www.bfr.bund.de/cd/52158.

The main functions fit.perc and fit.cont call a GUI that allows users to choose an appropriate distribution family to given data or to known quantiles without any knowledge of the R syntax.

Note

Fitting by given quantiles: a typical application is the definition of a distribution based on expert opinion on some quantiles (e.g., the 2.5th, median and 97.5th) of the trial to be modelled. rrisk has a functionality, to fit all continuous or discrete distributions simultaneously without urging the user to specify the distribution family in advance.

Author(s)

Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting),
Matthias Greiner <matthias.greiner@bfr.bund.de> (Federal Institute for Risk Assessment, Germany),
Kristin Tolksdorf <kristin.tolksdorf@bfr.bund.de> (Federal Institute for Risk Assessment, Germany),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting)

Examples

```
q <- stats::qweibull(p = c(0.025, 0.5, 0.975), shape = 2, scale = 3)
get.weibull.par(q = q)
q <- stats::qweibull(p = c(0.025, 0.5, 0.975), shape = 0.01, scale = 1)
get.weibull.par(q = q)

p <- c(0.025, 0.50, 0.975)
q <- c(9.68, 29.2, 50.98)
fit.results <- rriskFitdist.perc(p, q, show.output = FALSE)
plotDiagnostics.perc(fit.results)

p <- c(0.25, 0.50, 0.75)
q <- c(9.68, 29.2, 50.98)
fit.results <- rriskFitdist.perc(p, q, show.output = FALSE)
plotDiagnostics.perc(fit.results)
plotDiagnostics.perc(fit.results, tolPlot = 2)

## Not run:
  if( class(tcltk::tclRequire("Tktable")) == "tclObj" ) {
    res.fitcont <- fit.cont(data2fit = rnorm(100))
    res.fitcont
  }
  if( class(tcltk::tclRequire("Tktable")) == "tclObj" ) {
    res.fitperc <- fit.perc()
    res.fitperc
```

```
  }

## End(Not run)
```

`fit.cont` *GUI for fitting continuous distributions to given data*

Description

This function provides a GUI for choosing a most appropriate continuous distribution fitted to given data.

Usage

```
fit.cont(data2fit)
```

Arguments

`data2fit`	numerical vector, data to be fitted.

Value

Returns chosen continuous distribution, estimated parameters and data, on which the fitting is based.

Note

This function is used for defining a Monte-Carlo random variate item (`mcrv`) in the `rrisk` project.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Kristin Tolksdorf <kristin.tolksdorf@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

Examples

```
## Not run:
  if ( class(tcltk::tclRequire("Tktable")) == "tclObj" ) {
    res1 <- fit.cont(data2fit = rgamma(374, 4, 0.08))
    res1

    res2 <- fit.cont(data2fit = rbeta(300, shape1 = 1, shape2 = 2))
    res2

    res3 <- fit.cont(data2fit = mc2d::rtriang(300, min = 1, mode = 3, max = 10))
    res3
```

```
  res4 <- fit.cont(data2fit = stats::rnorm(300))
  res4
}

## End(Not run)
```

`fit.perc` *Choosing distribution by given quantiles*

Description

This function provides a GUI for choosing a most appropriate continuous distribution for known quantiles.

Usage

```
fit.perc(p = c(0.025, 0.5, 0.975), q = stats::qnorm(p), show.output = FALSE,
  tolPlot = 0.1, tolConv = 0.001, fit.weights = rep(1, length(p)))
```

Arguments

`p`	numerical vector of probabilities.
`q`	numerical vector of quantiles.
`show.output`	logical, if `TRUE` the output of the fitting functions `get.distribution.par` will be shown.
`tolPlot`	single positive numerical value giving a tolerance for plotting graphical diagnostics. If the sums of the differences between the distribution percentiles and the given percentiles are smaller than this value, the distribution will be plotted.
`tolConv`	positive numerical value, the absolute convergence tolerance for reaching zero.
`fit.weights`	numerical vector of the same length as a probabilities vector `p` containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.

Details

The argument `tolPlot` defines a tolerance for plotting graphical diagnostics. If the sums of the differences between the percentiles of the estimated distribution and the given percentiles are smaller than this value, the distribution will be plotted.

The items of the probability vector `p` should lie between 0 and 1.

Value

Returns a named list containing a chosen distribution, its estimated parameters and the data on which the estimation is based.

Note

This function is used for defining a Monte-Carlo random variate item (mcrv) in the rrisk project.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Kristin Tolksdorf <kristin.tolksdorf@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

Examples

```
## Not run:
    chosenDistr1 <- fit.perc()
    chosenDistr1

    chosenDistr2 <- fit.perc(tolPlot = 5)
    chosenDistr2

    chosenDistr3 <- fit.perc(p = c(0.3, 0.8, 0.9), q = c(10, 20, 40))
    chosenDistr3

    chosenDistr4 <- fit.perc(p = c(0.3, 0.8, 0.9), q = c(10, 30, 40))
    chosenDistr4

    chosenDistr5 <- fit.perc(p = c(0.3, 0.8, 0.9), q = c(10, 30, 40), tolPlot = 10)
    chosenDistr5

    ## Fitting a PERT distribution
    p <- c(0.025, 0.5, 0.6, 0.975)
    q <- round(mc2d::qpert(p = p, min = 0, mode = 3, max = 10, shape = 5), digits = 2)
    chosenDistr6 <- fit.perc(p = p, q = q, tolPlot = 10)
    chosenDistr6

## End(Not run)
```

---

get.beta.par *Fitting parameters of a Beta distribution from two or more quantiles*

Description

get.beta.par returns the parameters of estimated beta distribution where the pth percentiles match with the quantiles q.

Usage

```
get.beta.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE, plot = TRUE,
     tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

p	numeric, single value or vector of probabilities.
q	numeric, single value or vector of quantiles corresponding to p.
show.output	logical, if TRUE the optim result will be printed (default value is TRUE).
plot	logical, if TRUE the graphical diagnostics will be plotted (default value is TRUE).
tol	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is 0.001).
fit.weights	numerical vector of the same length as a probabilities vector p containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities, the number of quantiles and the numbers of weightings must be identical and should be at least two. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.beta.par uses the R function optim with the method L-BFGS-B. If this method fails the optimization method CG will be invoked.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a Beta distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the eror massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good til very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See pbeta for distribution implementation details.

Examples

```
q <- stats::qbeta(p = c(0.025, 0.5, 0.975), shape1 = 2, shape2 = 3)
old.par <- graphics::par(mfrow = c(2, 3))
get.beta.par(q = q)
get.beta.par(q = q, scaleX = c(0.001, 0.999))
get.beta.par(q = q, fit.weights = c(10, 1, 10))
get.beta.par(q = q, fit.weights = c(1, 10, 1))
get.beta.par(q = q, fit.weights = c(100, 1, 100))
get.beta.par(q = q, fit.weights = c(1, 100, 1))
graphics::par(old.par)

q <- stats::qbeta(p = c(0.025, 0.5, 0.975), shape1 = 1, shape2 = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.beta.par(q = q)
get.beta.par(q = q, fit.weights = c(10, 1, 10))
get.beta.par(q = q, fit.weights = c(1, 10, 1))
get.beta.par(q = q, fit.weights = c(100, 1, 100))
get.beta.par(q = q, fit.weights = c(1, 100, 1))
graphics::par(old.par)

q <- stats::qbeta(p = c(0.025, 0.5, 0.975), shape1 = 0.3, shape2 = 0.1)
old.par <- graphics::par(mfrow = c(2, 3))
get.beta.par(q = q)
get.beta.par(q = q, fit.weights = c(10, 1, 10))
get.beta.par(q = q, fit.weights = c(1, 10, 1))
get.beta.par(q = q, fit.weights = c(100, 1, 100))
get.beta.par(q = q, fit.weights = c(1, 100, 1))
```

```
graphics::par(old.par)

## example with only two quantiles
q <- stats::qbeta(p = c(0.025, 0.975), shape1 = 2, shape2 = 3)
old.par <- graphics::par(mfrow = c(2, 3))
get.beta.par(p = c(0.025, 0.975), q = q)
get.beta.par(p = c(0.025, 0.975), q = q, fit.weights = c(10, 1))
get.beta.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 10))
get.beta.par(p = c(0.025, 0.975), q = q, fit.weights = c(100, 1))
get.beta.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 100))
graphics::par(old.par)
```

`get.cauchy.par` *Fitting parameters of a Cauchy distribution from two or more quantiles*

Description

`get.cauchy.par` returns the parameters of a Cauchy distribution where the pth percentiles match with the quantiles `q`.

Usage

```
get.cauchy.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
  plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

`p`	numeric, single value or vector of probabilities.
`q`	numeric, single value or vector of quantiles corresponding to p.
`show.output`	logical, if `TRUE` the `optim` result will be printed (default value is `TRUE`).
`plot`	logical, if `TRUE` the graphical diagnostics will be plotted (default value is `TRUE`).
`tol`	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is `0.001`).
`fit.weights`	numerical vector of the same length as a probabilities vector `p` containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
`scaleX`	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if `plot = TRUE`). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
`...`	further arguments passed to the functions `plot` and `points` (relevant only if `plot = TRUE`).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least two. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. `get.cauchy.par` uses the R function `optim` with the method `L-BFGS-B`.

If `show.output = TRUE` the output of the function `optim` will be shown. The item `convergence` equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item `value` displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function `optim` are accepted if the achieved value of the minimized function (output component `value` of `optim`) is smaller than the argument `tol`.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector `fit.weights` should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a Cauchy distribution or missing values (`NA`'s) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the eror massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good til very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See pcauchy for distribution implementation details.

Examples

```
q <- stats::qcauchy(p = c(0.025, 0.5, 0.975), location = 0, scale = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.cauchy.par(q = q)
get.cauchy.par(q = q, scaleX = c(0.5, 0.5))
get.cauchy.par(q = q, fit.weights = c(1, 10, 1), scaleX = c(0.5, 0.5))
get.cauchy.par(q = q, fit.weights = c(1, 100, 1), scaleX = c(0.5, 0.5))
get.cauchy.par(q = q, fit.weights = c(10, 1, 10), scaleX = c(0.5, 0.5))
get.cauchy.par(q = q, fit.weights = c(100, 1, 100), scaleX = c(0.5, 0.5))
graphics::par(old.par)

q <- stats::qcauchy(p = c(0.025, 0.5, 0.975), location = 3, scale = 5)
old.par <- graphics::par(mfrow = c(2, 3))
get.cauchy.par(q = q, scaleX = c(0.5, 0.5))
get.cauchy.par(q = q, fit.weights = c(1, 10, 1), scaleX = c(0.5, 0.5))
get.cauchy.par(q = q, fit.weights = c(1, 100, 1), scaleX = c(0.5, 0.5))
get.cauchy.par(q = q, fit.weights = c(10, 1, 10), scaleX = c(0.5, 0.5))
get.cauchy.par(q = q, fit.weights = c(100, 1, 100), scaleX = c(0.5, 0.5))
graphics::par(old.par)

q <- stats::qcauchy(p = c(0.025, 0.5, 0.975), location = 0.1, scale = 0.1)
old.par <- graphics::par(mfrow = c(2, 3))
get.cauchy.par(q = q, scaleX = c(0.5, 0.5))
get.cauchy.par(q = q, fit.weights = c(1, 10, 1), scaleX = c(0.5, 0.5))
get.cauchy.par(q = q, fit.weights = c(1, 100, 1), scaleX = c(0.5, 0.5))
get.cauchy.par(q = q, fit.weights = c(10, 1, 10), scaleX = c(0.5, 0.5))
get.cauchy.par(q = q, fit.weights = c(100, 1, 100), scaleX = c(0.5, 0.5))
graphics::par(old.par)

## example with only two quantiles
q <- stats::qcauchy(p = c(0.025, 0.975), location = 0.1, scale = 0.1)
old.par <- graphics::par(mfrow = c(2, 3))
get.cauchy.par(p = c(0.025, 0.975), q = q, scaleX = c(0.5, 0.5))
get.cauchy.par(p = c(0.025, 0.975), q = q, fit.weights = c(10, 1), scaleX = c(0.5, 0.5))
get.cauchy.par(p = c(0.025, 0.975), q = q, fit.weights = c(100, 1), scaleX = c(0.5, 0.5))
get.cauchy.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 10), scaleX = c(0.5, 0.5))
get.cauchy.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 100), scaleX = c(0.5, 0.5))
graphics::par(old.par)
```

`get.chisq.par` *Fitting parameter of a chi-square distribution from one or more quantiles*

Description

get.chisq.par returns the parameters of a chi-square distribution where the pth percentiles match with the quantiles q.

Usage

```
get.chisq.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
  plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

p	numeric, single value or vector of probabilities.
q	numeric, single value or vector of quantiles corresponding to p.
show.output	logical, if TRUE the optim result will be printed (default value is TRUE).
plot	logical, if TRUE the graphical diagnostics will be plotted (default value is TRUE).
tol	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is 0.001).
fit.weights	numerical vector of the same length as a probabilities vector p containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)). for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE)

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least one. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.chisq.par uses the R function optim with the method L-BFGS-B.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted, if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values. The function which

will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a chi-square distribution or missing values (NA's), if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See pchisq for distribution implementation details.

Examples

```
q <- stats::qchisq(p = c(0.025, 0.5, 0.975), df = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.chisq.par(q = q)
get.chisq.par(q = q, fit.weights = c(10, 1, 10))
get.chisq.par(q = q, fit.weights = c(100, 1, 100))
get.chisq.par(q = q, fit.weights = c(1, 10, 1))
get.chisq.par(q = q, fit.weights = c(1, 100, 1))
graphics::par(old.par)

q <- stats::qchisq(p = c(0.025, 0.5, 0.975), df = 0.1)
old.par <- graphics::par(mfrow = c(2, 3))
get.chisq.par(q = q, scaleX = c(0.1, 0.1))
get.chisq.par(q = q, fit.weights = c(10, 1, 10))
get.chisq.par(q = q, fit.weights = c(100, 1, 100))
get.chisq.par(q = q, fit.weights = c(1, 10, 1))
get.chisq.par(q = q, fit.weights = c(1, 100, 1))
```

```
graphics::par(old.par)

q <- stats::qchisq(p = c(0.025, 0.5, 0.975), df = 20)
old.par <- graphics::par(mfrow = c(2, 3))
get.chisq.par(q = q)
get.chisq.par(q = q, fit.weights = c(10, 1, 10))
get.chisq.par(q = q, fit.weights = c(100, 1, 100))
get.chisq.par(q = q, fit.weights =c(1, 10, 1))
get.chisq.par(q = q, fit.weights =c(1, 100, 1))
graphics::par(old.par)

## example with only one quantile
q <- stats::qchisq(p = c(0.025), df = 20)
old.par <- graphics::par(mfrow = c(1, 3))
get.chisq.par(p = c(0.025), q = q)
get.chisq.par(p = c(0.025), q = q, fit.weights = 10)
get.chisq.par(p = c(0.025), q = q, fit.weights = 100)
graphics::par(old.par)
```

`get.chisqnc.par` *Fitting parameters of a non-central chi-square distribution from one or more quantiles*

Description

`get.chisqnc.par` returns the parameters of a non-central chi-square distribution where the pth percentiles match with the quantiles `q`.

Usage

```
get.chisqnc.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
  plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

`p`	numeric, single value or vector of probabilities.
`q`	numeric, single value or vector of quantiles corresponding to p.
`show.output`	logical, if `TRUE` the `optim` result will be printed (default value is `TRUE`).
`plot`	logical, if `TRUE` the graphical diagnostics will be plotted (default value is `TRUE`).
`tol`	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is `0.001`).
`fit.weights`	numerical vector of the same length as a probabilities vector `p` containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.

scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities, the number of quantiles and the number of weights must be identical and should be at least one. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.chisqnc.par uses the R function optim with the method L-BFGS-B.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a non-central chi-square distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the eror massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See pchisq for distribution implementation details.

Examples

```
q <- stats::qchisq(p = c(0.025, 0.5, 0.975), df = 2, ncp = 4)
old.par <- graphics::par(mfrow = c(2, 3))
get.chisqnc.par(q = q)
get.chisqnc.par(q = q, scaleX = c(0.1, 0.9999999))
get.chisqnc.par(q = q, fit.weights = c(100, 1, 100))
get.chisqnc.par(q = q, fit.weights = c(10, 1, 10))
get.chisqnc.par(q = q, fit.weights = c(1, 100, 1))
get.chisqnc.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qchisq(p = c(0.025, 0.5, 0.975), df = 0.1, ncp = 0.4)
old.par <- graphics::par(mfrow = c(2, 3))
get.chisqnc.par(q = q)
get.chisqnc.par(q = q, fit.weights = c(100, 1, 100))
get.chisqnc.par(q = q, fit.weights = c(10, 1, 10))
get.chisqnc.par(q = q, fit.weights = c(1, 100, 1))
get.chisqnc.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qchisq(p = c(0.025, 0.5, 0.975), df = 1, ncp = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.chisqnc.par(q = q)
get.chisqnc.par(q = q, fit.weights = c(100, 1, 100))
get.chisqnc.par(q = q, fit.weights = c(10, 1, 10))
get.chisqnc.par(q = q, fit.weights = c(1, 100, 1))
get.chisqnc.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

## example with only two quantile
q <- stats::qchisq(p = c(0.025, 0.95), df = 20, ncp = 20)
old.par <- graphics::par(mfrow = c(2, 3))
get.chisqnc.par(p = c(0.025, 0.975), q = q)
get.chisqnc.par(p = c(0.025, 0.975), q = q, fit.weights = c(100, 1))
get.chisqnc.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 100))
get.chisqnc.par(p = c(0.025, 0.975), q = q, fit.weights = c(10, 1))
get.chisqnc.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 10))
graphics::par(old.par)
```

get.exp.par *Fitting parameters of an exponential distribution from one or more quantiles*

Description

get.exp.par returns the parameters of an exponential distribution where the pth percentiles match with the quantiles q.

Usage

```
get.exp.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
   plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

p	numeric, single value or vector of probabilities.
q	numeric, single value or vector of quantiles corresponding to p.
show.output	logical, if TRUE the optim result will be printed (default value is TRUE).
plot	logical, if TRUE the graphical diagnostics will be plotted (default value is TRUE).
tol	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is 0.001).
fit.weights	numerical vector of the same length as a probabilities vector p containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least one. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.exp.par uses the R function optim with the method L-BFGS-B. If this method fails the optimization method BFGS will be invoked.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values. The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of an exponential distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the eror massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good til very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See pexp for distribution implementation details.

Examples

```
q <- stats::qexp(p = c(0.025, 0.5, 0.975), rate = 2)
old.par <- graphics::par(mfrow = c(2, 3))
get.exp.par(q = q)
get.exp.par(q = q, fit.weights = c(100, 1, 100))
get.exp.par(q = q, fit.weights = c(10, 1, 10))
get.exp.par(q = q, fit.weights = c(1, 100, 1))
get.exp.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)
```

```
q <- stats::qexp(p = c(0.025, 0.5, 0.975), rate = 0.1)
old.par <- graphics::par(mfrow = c(2, 3))
get.exp.par(q = q)
get.exp.par(q = q, fit.weights = c(100, 1, 100))
get.exp.par(q = q, fit.weights = c(10, 1, 10))
get.exp.par(q = q, fit.weights = c(1, 100, 1))
get.exp.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qexp(p = c(0.025, 0.5, 0.975), rate = 0.001)
old.par <- graphics::par(mfrow = c(2, 3))
get.exp.par(q = q)
get.exp.par(q = q, fit.weights = c(100, 1, 100))
get.exp.par(q = q, fit.weights = c(10, 1, 10))
get.exp.par(q = q, fit.weights = c(1, 100, 1))
get.exp.par(q = q, tol = 0.2, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qexp(p = c(0.025, 0.5, 0.975), rate = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.exp.par(q = q)
get.exp.par(q = q, fit.weights = c(100, 1, 100))
get.exp.par(q = q, fit.weights = c(10, 1, 10))
get.exp.par(q = q, fit.weights = c(1, 100, 1))
get.exp.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

## example with only one quantile
q <- stats::qexp(p = c(0.025), rate = 2)
old.par <- graphics::par(mfrow = c(1, 3))
get.exp.par(p = c(0.025), q = q)
get.exp.par(p = c(0.025), q = q, fit.weights = 10)
get.exp.par(p = c(0.025), q = q, fit.weights = 100)
graphics::par(old.par)
```

`get.f.par` *Fitting parameters of a F distribution from two or more quantiles*

Description

`get.f.par` returns the parameters of a F distribution where the pth percentiles match with the quantiles q.

Usage

```
get.f.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
  plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

p	numeric, single value or vector of probabilities.
q	numeric, single value or vector of quantiles corresponding to p.
show.output	logical, if TRUE the optim result will be printed (default value is TRUE).
plot	logical, if TRUE the graphical diagnostics will be plotted (default value is TRUE).
tol	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is 0.001).
fit.weights	numerical vector of the same length as a probabilities vector p containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least two. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.f.par uses the R function optim with the method L-BFGS-B. If this method fails the optimization method BFGS will be invoked.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a F distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See `stats::pf` for distribution implementation details.

Examples

```
q <- stats::qf(p = c(0.025, 0.5, 0.975), df1 = 2, df2 = 10)
old.par <- graphics::par(mfrow = c(2, 3))
get.f.par(q = q, scaleX = c(0.1, 0.5))
get.f.par(q = q, fit.weights = c(100, 1, 100), scaleX = c(0.1, 0.5))
get.f.par(q = q, fit.weights = c(10, 1, 10), scaleX = c(0.1, 0.5))
get.f.par(q = q, fit.weights = c(1, 100, 1), scaleX = c(0.1, 0.5))
get.f.par(q = q, fit.weights = c(1, 10, 1), scaleX = c(0.1, 0.5))
graphics::par(old.par)

q <- stats::qf(p = c(0.025, 0.5, 0.975), df1 = 0.2, df2 = 0.3)
old.par <- graphics::par(mfrow = c(2, 3))
get.f.par(q = q, scaleX = c(0.1, 0.2))
get.f.par(q = q, fit.weights = c(100, 1, 100), scaleX = c(0.1, 0.999))
get.f.par(q = q, fit.weights = c(10, 1, 10), scaleX = c(0.1, 0.2))
get.f.par(q = q, fit.weights = c(1, 100, 1), scaleX = c(0.1, 0.9999))
get.f.par(q = q, fit.weights = c(1, 10, 1), scaleX = c(0.1, 0.9999))
graphics::par(old.par)

q <- stats::qf(p = c(0.025, 0.5, 0.975), df1 = 1, df2 = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.f.par(q = q, scaleX = c(0.1, 0.2))
get.f.par(q = q, fit.weights = c(100, 1, 100), scaleX = c(0.1, 0.2))
get.f.par(q = q, fit.weights = c(10, 1, 10), scaleX = c(0.1, 0.2))
get.f.par(q = q, fit.weights = c(1, 100, 1), scaleX = c(0.1, 0.2))
get.f.par(q = q, fit.weights = c(1, 10, 1), scaleX = c(0.1, 0.2))
graphics::par(old.par)
```

```
## example with only two quantiles
q <- stats::qf(p = c(0.025, 0.975), df1 = 2, df2 = 3)
old.par <- graphics::par(mfrow = c(1, 3))
get.f.par(p = c(0.025, 0.975), q = q)
get.f.par(p = c(0.025, 0.975), q = q, fit.weights = c(100, 1))
get.f.par(p = c(0.025, 0.975), q = q, fit.weights = c(10, 1))
graphics::par(old.par)
```

`get.gamma.par` *Fitting parameters of a gamma distribution from two or more quantiles*

Description

`get.gamma.par` returns the parameters of a gamma distribution where the pth percentiles match with the quantiles `q`.

Usage

```
get.gamma.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
   plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

`p`	numeric, single value or vector of probabilities.
`q`	numeric, single value or vector of quantiles corresponding to p.
`show.output`	logical, if `TRUE` the `optim` result will be printed (default value is `TRUE`).
`plot`	logical, if `TRUE` the graphical diagnostics will be plotted (default value is `TRUE`).
`tol`	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is `0.001`).
`fit.weights`	numerical vector of the same length as a probabilities vector `p` containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
`scaleX`	numerical vector of the length 2 containing values (from the open interval (0, 1)). for scaling quantile-axis (relevant only if `plot = TRUE`). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
`...`	further arguments passed to the functions `plot` and `points` (relevant only if `plot = TRUE`).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least two. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. `get.gamma.par` uses the R function `optim` with the method `L-BFGS-B`. If this method fails the optimization method `BFGS` will be invoked.

If `show.output = TRUE` the output of the function `optim` will be shown. The item `convergence` equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item `value` displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function `optim` are accepted if the achieved value of the minimized function (output component `value` of `optim`) is smaller than the argument `tol`.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector `fit.weights` should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a gamma distribution or missing values (`NA`'s) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See pgamma for distribution implementation details.

Examples

```
q <- stats::qgamma(p = c(0.025, 0.5, 0.975), shape = 10, rate = 10)
old.par <- graphics::par(mfrow = c(2, 3))
get.gamma.par(q = q)
get.gamma.par(q = q, scaleX = c(0.00001, 0.9999))
get.gamma.par(q = q, fit.weights = c(100, 1, 100))
get.gamma.par(q = q, fit.weights = c(10, 1, 10))
get.gamma.par(q = q, fit.weights = c(1, 100, 1))
get.gamma.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qgamma(p = c(0.025, 0.5, 0.975), shape = 0.1, rate = 0.1)
old.par <- graphics::par(mfrow = c(2, 3))
get.gamma.par(q = q)
get.gamma.par(q = q, fit.weights = c(100, 1, 100))
get.gamma.par(q = q, fit.weights = c(10, 1, 10))
get.gamma.par(q = q, fit.weights = c(1, 100, 1))
get.gamma.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qgamma(p = c(0.025, 0.5, 0.975), shape = 1, rate = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.gamma.par(q = q)
get.gamma.par(q = q, fit.weights = c(100, 1, 100))
get.gamma.par(q = q, fit.weights = c(10, 1, 10))
get.gamma.par(q = q, fit.weights = c(1, 100, 1))
get.gamma.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

## example with only two quantiles
q <- stats::qgamma(p = c(0.025, 0.975), shape = 10, rate = 10)
old.par <- graphics::par(mfrow = c(2, 3))
get.gamma.par(p = c(0.025, 0.975), q = q)
get.gamma.par(p = c(0.025, 0.975), q = q, fit.weights = c(100, 1))
get.gamma.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 100))
get.gamma.par(p = c(0.025, 0.975), q = q, fit.weights = c(10, 1))
get.gamma.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 10))
graphics::par(old.par)
```

`get.gompertz.par` *Fitting parameters of a Gompertz distribution from two or more quantiles*

Description

get.gompertz.par returns the parameters of a Gompertz distribution where the pth percentiles match with the quantiles q.

Usage

```
get.gompertz.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
   plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

p	numeric, single value or vector of probabilities.
q	numeric, single value or vector of quantiles corresponding to p.
show.output	logical, if TRUE the optim result will be printed (default value is TRUE).
plot	logical, if TRUE the graphical diagnostics will be plotted (default value is TRUE).
tol	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is 0.001).
fit.weights	numerical vector of the same length as a probabilities vector p containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least two. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.gompertz.par uses the R function optim with the method L-BFGS-B. If this method fails the optimization method BFGS will be invoked.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector `fit.weights` should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a Gompertz distribution or missing values (`NA`'s) if the distribution cannot fit the specified quantiles.

Note

Comply with a parametrization of this distribution. The definition of this distribution in the literature is not unique.

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See `pgompertz` from the package **eha** for distribution implementation details.

Examples

```
q <- eha::qgompertz(p = c(0.025, 0.5, 0.975), shape = 2, scale = 5)
old.par <- graphics::par(mfrow = c(2, 3))
get.gompertz.par(q = q)
get.gompertz.par(q = q, fit.weights = c(100, 1, 100))
get.gompertz.par(q = q, fit.weights = c(10, 1, 10))
get.gompertz.par(q = q, fit.weights = c(1, 100, 1))
get.gompertz.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)
```

```
q <- eha::qgompertz(p = c(0.025, 0.5, 0.975), shape = 0.2, scale = 0.5)
old.par <- graphics::par(mfrow = c(2, 3))
get.gompertz.par(q = q)
get.gompertz.par(q = q, fit.weights = c(100, 1, 100))
get.gompertz.par(q = q, fit.weights = c(10, 1, 10))
get.gompertz.par(q = q, fit.weights = c(1, 100, 1))
get.gompertz.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- eha::qgompertz(p = c(0.025, 0.5, 0.975), shape = 1, scale = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.gompertz.par(q = q)
get.gompertz.par(q = q, fit.weights = c(100, 1, 100))
get.gompertz.par(q = q, fit.weights = c(10, 1, 10))
get.gompertz.par(q = q, fit.weights = c(1, 100, 1))
get.gompertz.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

## example with only two quantiles
q <- eha::qgompertz(p = c(0.025, 0.975), shape = 2, scale = 5)
old.par <- graphics::par(mfrow = c(2, 3))
get.gompertz.par(p = c(0.025, 0.975), q = q)
get.gompertz.par(p = c(0.025, 0.975), q = q, fit.weights = c(100, 1), scaleX = c(0.0001, 0.9999))
get.gompertz.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 100))
get.gompertz.par(p = c(0.025, 0.975), q = q, fit.weights = c(10, 1))
get.gompertz.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 10))
graphics::par(old.par)
```

`get.hyper.par` *Fitting parameters of a hypergeometric distribution from three or more quantiles*

Description

`get.hyper.par` returns the parameters of a hypergeometric distribution where the pth percentiles match with the quantiles q.

Usage

```
get.hyper.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
   plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

p	numeric, single value or vector of probabilities.
q	numeric, single value or vector of quantiles corresponding to p.
show.output	logical, if TRUE the optim result will be printed (default value is TRUE).
plot	logical, if TRUE the graphical diagnostics will be plotted (default value is TRUE).

tol	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is 0.001).
fit.weights	numerical vector of the same length as a probabilities vector p containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least three. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.hyper.par uses the R function optim with the method L-BFGS-B.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a hypergeometric distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a

"similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See phyper for distribution implementation details.

Examples

```
q <- stats::qhyper(p = c(0.025, 0.5, 0.975), m = 5, n = 3, k = 3)
old.par <- graphics::par(mfrow = c(2, 3))
get.hyper.par(q = q)
get.hyper.par(q = q, tol = 1)
get.hyper.par(q = q, fit.weights = c(100, 1, 100))
get.hyper.par(q = q, fit.weights = c(10, 1, 10))
get.hyper.par(q = q, fit.weights = c(1, 100, 1))
get.hyper.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qhyper(p = c(0.025, 0.5, 0.975), m = 10, n = 5, k = 4)
old.par <- graphics::par(mfrow = c(2, 3))
get.hyper.par(q = q)
get.hyper.par(q = q, fit.weights = c(100, 1, 100))
get.hyper.par(q = q, fit.weights = c(10, 1, 10))
get.hyper.par(q = q, fit.weights = c(1, 100, 1))
get.hyper.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)
```

---

get.lnorm.par *Fitting parameters of a lognormal distribution from two or more quantiles*

Description

get.lnorm.par returns the parameters of a lognormal distribution where the pth percentiles match with the quantiles q.

Usage

```
get.lnorm.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
   plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

p	numeric, single value or vector of probabilities.
q	numeric, single value or vector of quantiles corresponding to p.
show.output	logical, if TRUE the optim result will be printed (default value is TRUE).
plot	logical, if TRUE the graphical diagnostics will be plotted (default value is TRUE).
tol	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is 0.001).
fit.weights	numerical vector of the same length as a probabilities vector p containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least two. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.lnorm.par uses the R function optim with the method L-BFGS-B. If this method fails the optimization method Nelder-Mead will be invoked.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a lognormal distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

Comply with a parametrization of this distribution. The definition of this distribution in the literature is not unique.

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See plnorm for distribution implementation details.

Examples

```
q <- stats::qlnorm(p = c(0.025, 0.5, 0.975), meanlog = 4, sdlog = 0.8)
old.par <- graphics::par(mfrow = c(2, 3))
get.lnorm.par(q = q)
get.lnorm.par(q = q, fit.weights = c(100, 1, 100))
get.lnorm.par(q = q, fit.weights = c(10, 1, 10))
get.lnorm.par(q = q, fit.weights = c(1, 100, 1))
get.lnorm.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qlnorm(p = c(0.025, 0.5, 0.975), meanlog=-4, sdlog = 0.8)
old.par <- graphics::par(mfrow = c(2, 3))
get.lnorm.par(q = q)
get.lnorm.par(q = q, fit.weights = c(100, 1, 100))
get.lnorm.par(q = q, fit.weights = c(10, 1, 10))
get.lnorm.par(q = q, fit.weights = c(1, 100, 1))
get.lnorm.par(q = q, fit.weights = c(1, 10, 1))
```

```
graphics::par(old.par)

q <- stats::qlnorm(p = c(0.025, 0.5, 0.975), meanlog = 1, sdlog = 0.1)
old.par <- graphics::par(mfrow = c(2, 3))
get.lnorm.par(q = q)
get.lnorm.par(q = q, fit.weights = c(100, 1, 100))
get.lnorm.par(q = q, fit.weights = c(10, 1, 10))
get.lnorm.par(q = q, fit.weights = c(1, 100, 1), scaleX = c(0.000001, 0.99999999))
get.lnorm.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qlnorm(p = c(0.025, 0.5, 0.975), meanlog = 0.1, sdlog = 0.1)
old.par <- graphics::par(mfrow = c(2, 3))
get.lnorm.par(q = q)
get.lnorm.par(q = q, fit.weights = c(100, 1, 100))
get.lnorm.par(q = q, fit.weights = c(10, 1, 10))
get.lnorm.par(q = q, fit.weights = c(1, 100, 1))
get.lnorm.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

## example with only two quantiles
q <- stats::qlnorm(p = c(0.025, 0.975), meanlog = 4, sdlog = 0.8)
old.par <- graphics::par(mfrow = c(2, 3))
get.lnorm.par(p = c(0.025, 0.975), q = q)
get.lnorm.par(p = c(0.025, 0.975), q = q, fit.weights = c(100, 1), scaleX = c(0.1, 0.001))
get.lnorm.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 100), scaleX = c(0.1, 0.001))
get.lnorm.par(p = c(0.025, 0.975), q = q, fit.weights = c(10, 1))
get.lnorm.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 10))
graphics::par(old.par)
```

`get.logis.par` *Fitting parameters of a logistic distribution from two or more quantiles*

Description

`get.logis.par` returns the parameters of a logistic distribution where the pth percentiles match with the quantiles q.

Usage

```
get.logis.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
  plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

p	numeric, single value or vector of probabilities.
q	numeric, single value or vector of quantiles corresponding to p.
show.output	logical, if TRUE the optim result will be printed (default value is TRUE).

plot	logical, if TRUE the graphical diagnostics will be plotted (default value is TRUE).
tol	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is 0.001).
fit.weights	numerical vector of the same length as a probabilities vector p containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least two. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.logis.par uses the R function optim with the method L-BFGS-B. If this method fails the optimization method BFGS will be invoked.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values. The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a logistic distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look

very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See plogis for distribution implementation details.

Examples

```
q <- stats::qlogis(p = c(0.025, 0.5, 0.975), location = 0, scale = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.logis.par(q = q)
get.logis.par(q = q, scaleX = c(0.5, 0.5))
get.logis.par(q = q, fit.weights = c(100, 1, 100))
get.logis.par(q = q, fit.weights = c(10, 1, 10))
get.logis.par(q = q, fit.weights = c(1, 100, 1))
get.logis.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qlogis(p = c(0.025, 0.5, 0.975), location = 0, scale = 3)
old.par <- graphics::par(mfrow = c(2, 3))
get.logis.par(q = q)
get.logis.par(q = q, fit.weights = c(100, 1, 100))
get.logis.par(q = q, fit.weights = c(10, 1, 10))
get.logis.par(q = q, fit.weights = c(1, 100, 1))
get.logis.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

## example with only two quantiles
q <- stats::qlogis(p = c(0.025, 0.975), location = 0, scale = 3)
old.par <- graphics::par(mfrow = c(1, 3))
get.logis.par(p = c(0.025, 0.975), q = q)
get.logis.par(p = c(0.025, 0.975), q = q, fit.weights = c(100, 1))
get.logis.par(p = c(0.025, 0.975), q = q, fit.weights = c(10, 1))
graphics::par(old.par)
```

`get.nbinom.par` *Fitting parameters of a negative binomial distribution from two or more quantiles*

Description

`get.nbinom.par` returns the parameters of a negative binomial distribution where the `pth` percentiles match with the quantiles `q`.

Usage

```
get.nbinom.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
   plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

`p`	numeric, single value or vector of probabilities.
`q`	numeric, single value or vector of quantiles corresponding to p.
`show.output`	logical, if `TRUE` the `optim` result will be printed (default value is `TRUE`).
`plot`	logical, if `TRUE` the graphical diagnostics will be plotted (default value is `TRUE`).
`tol`	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is `0.001`).
`fit.weights`	numerical vector of the same length as a probabilities vector `p` containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
`scaleX`	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if `plot = TRUE`). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
`...`	further arguments passed to the functions `plot` and `points` (relevant only if `plot = TRUE`).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least two. Using the default `p`, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. `get.nbinom.par` uses the R function `optim` with the method `L-BFGS-B`.

If `show.output = TRUE` the output of the function `optim` will be shown. The item `convergence` equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item `value` displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function `optim` are accepted if the achieved

value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a negative binomial distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See pnbinom for distribution implementation details.

Examples

```
q <- stats::qnbinom(p = c(0.025, 0.5, 0.975), size = 10, prob = 0.5)
old.par <- graphics::par(mfrow = c(2, 3))
get.nbinom.par(q = q)
get.nbinom.par(q = q, fit.weights = c(100, 1, 100))
get.nbinom.par(q = q, fit.weights = c(1, 100, 1))
get.nbinom.par(q = q, fit.weights = c(10, 1, 10))
get.nbinom.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)
```

```
q <- stats::qnbinom(p = c(0.025, 0.5, 0.975), size = 1, prob = 0.5)
old.par <- graphics::par(mfrow = c(2, 3))
get.nbinom.par(q = q, tol = 0.01)
get.nbinom.par(q = q, fit.weights = c(100, 1, 100))
get.nbinom.par(q = q, fit.weights = c(1, 100, 1), tol = 0.01)
get.nbinom.par(q = q, fit.weights = c(10, 1, 10), tol = 0.01)
get.nbinom.par(q = q, fit.weights = c(1, 10, 1), tol = 0.01)
graphics::par(old.par)

q <- stats::qnbinom(p = c(0.025, 0.5, 0.975), size = 1, prob = 0.1)
old.par <- graphics::par(mfrow = c(2, 3))
get.nbinom.par(q = q)
get.nbinom.par(q = q, fit.weights = c(100, 1, 100))
get.nbinom.par(q = q, fit.weights = c(1, 100, 1))
get.nbinom.par(q = q, fit.weights = c(10, 1, 10))
get.nbinom.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

## example with only two quantiles
q <- stats::qnbinom(p = c(0.025, 0.975), size = 10, prob = 0.5)
old.par <- graphics::par(mfrow = c(2, 3))
get.nbinom.par(p = c(0.025, 0.975), q = q,)
get.nbinom.par(p = c(0.025, 0.975), q = q, fit.weights = c(100, 1))
get.nbinom.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 100))
get.nbinom.par(p = c(0.025, 0.975), q = q, fit.weights = c(10, 1))
get.nbinom.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 10))
graphics::par(old.par)
```

`get.norm.par` *Fitting parameters of normal distribution from two or more quantiles*

Description

`get.norm.par` returns the parameters of a normal distribution where the pth percentiles match with the quantiles `q`.

Usage

```
get.norm.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
  plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

`p`	numeric, single value or vector of probabilities.
`q`	numeric, single value or vector of quantiles corresponding to p.
`show.output`	logical, if `TRUE` the `optim` result will be printed (default value is `TRUE`).
`plot`	logical, if `TRUE` the graphical diagnostics will be plotted (default value is `TRUE`).

tol	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is 0.001).
fit.weights	numerical vector of the same length as a probabilities vector p containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least two. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.norm.par uses the R function optim with the method L-BFGS-B. If this method fails the optimization method BFGS will be invoked.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a normal distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look

very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See pnorm for distribution implementation details.

Examples

```
q <- stats::qnorm(p = c(0.025, 0.5, 0.975), mean = 12, sd = 34)
old.par <- graphics::par(mfrow = c(2, 3))
get.norm.par(q = q)
get.norm.par(q = q, scaleX = c(0.00001, 0.99999))
get.norm.par(q = q, fit.weights = c(10, 1, 10))
get.norm.par(q = q, fit.weights = c(1, 10, 1))
get.norm.par(q = q, fit.weights = c(100, 1, 100))
get.norm.par(q = q, fit.weights = c(1, 100, 1))
graphics::par(old.par)

q <- stats::qnorm(p = c(0.025, 0.5, 0.975), mean = 0, sd = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.norm.par(q = q)
get.norm.par(q = q, fit.weights = c(10, 1, 10))
get.norm.par(q = q, fit.weights = c(1, 10, 1))
get.norm.par(q = q, fit.weights = c(100, 1, 100))
get.norm.par(q = q, fit.weights = c(1, 100, 1))
graphics::par(old.par)

q <- stats::qnorm(p = c(0.025, 0.5, 0.975), mean = 0.1, sd = 0.1)
old.par <- graphics::par(mfrow = c(2, 3))
get.norm.par(q = q)
get.norm.par(q = q, fit.weights = c(10, 1, 10))
get.norm.par(q = q, fit.weights = c(1, 10, 1))
get.norm.par(q = q, fit.weights = c(100, 1, 100))
get.norm.par(q = q, fit.weights = c(1, 100, 1))
graphics::par(old.par)

## example with only two quantiles
q <- stats::qnorm(p = c(0.025, 0.975), mean = 12, sd = 34)
old.par <- graphics::par(mfrow = c(2, 3))
get.norm.par(p = c(0.025, 0.975), q = q)
```

```
get.norm.par(p = c(0.025, 0.975), q = q, fit.weights = c(10, 1))
get.norm.par(p = c(0.025, 0.975), q = q, fit.weights = c(100, 1))
get.norm.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 10))
get.norm.par(p = c(0.025, 0.975), q = q, fit.weights = c(1, 100))
graphics::par(old.par)
```

`get.norm.sd` *Fitting standard deviation of a normal distribution from one or more quantiles and known mean*

Description

`get.norm.sd` returns the standard deviation of a normal distribution where the pth percentiles match with the quantiles `q`.

Usage

```
get.norm.sd(p = c(0.025, 0.5, 0.975), q, show.output = TRUE, plot = TRUE,
                   fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

`p`	numeric, vector of probabilities.
`q`	numeric, vector of quantiles corresponding to p.
`show.output`	logical, if `TRUE` the `optim` result will be printed (default vaule is `TRUE`).
`plot`	logical, if `TRUE` the graphical diagnostics will be plotted (default value is `TRUE`).
`fit.weights`	numerical vector of the same length as a probabilities vector `p` containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
`scaleX`	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if `plot = TRUE`). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
`...`	further arguments passed to the functions `plot` and `points` (relevant only if `plot = TRUE`).

Details

The number of probabilities and the number of quantiles must be identical and should be at least two. `get.norm.sd` uses the central limit theorem and the linear regression.

If `show.output = TRUE` the output of the function `lm` will be shown.

The items of the probability vector `p` should lie between 0 and 1.

The items of the weighting vector `fit.weights` should be positive values.

The function will be meaningful only if the quantile comes from a normal distribution.

Value

Returns an estimated standard deviation or missing value

Note

It should be noted that the data must be normally distributed, or the central limt theorem must hold for large (enough) samples sizes.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See `pnorm` for distribution implementation details.

Examples

```
q <- stats::qnorm(p = c(0.025, 0.5, 0.975), mean = 0, sd = 2)
old.par <- graphics::par(mfrow = c(2, 3))
get.norm.sd(q = q)
get.norm.sd(q = q, scaleX = c(0.0001, 0.9999))
get.norm.sd(q = q, fit.weights = c(10, 1, 10))
get.norm.sd(q = q, fit.weights = c(1, 10, 1))
get.norm.sd(q = q, fit.weights = c(100, 1, 100))
get.norm.sd(q = q, fit.weights = c(1, 100, 1))
graphics::par(old.par)

q <- stats::qnorm(p = c(0.025, 0.5, 0.975), mean = 176, sd = 15)
old.par <- graphics::par(mfrow = c(2, 3))
get.norm.sd(q = q)
get.norm.sd(q = q, fit.weights = c(10, 1, 10))
get.norm.sd(q = q, fit.weights = c(1, 10, 1))
get.norm.sd(q = q, fit.weights = c(100, 1, 100))
get.norm.sd(q = q, fit.weights = c(1, 100, 1))
graphics::par(old.par)

## The estimation model is not suitable for the following quantiles.
## Because the quantile is unsymmetrical, which could not be from a normally distributed data.
q <- c(-2, 30, 31)
old.par <- graphics::par(mfrow = c(2, 3))
get.norm.sd(q = q)
get.norm.sd(q = q, fit.weights = c(10, 1, 10))
get.norm.sd(q = q, fit.weights = c(1, 10, 1), scaleX = c(0.0001, 0.9999))
get.norm.sd(q = q, fit.weights = c(100, 1, 100))
```

```
get.norm.sd(q = q, fit.weights = c(1, 100, 1), scaleX = c(0.0001, 0.9999))
graphics::par(old.par)

## Estimating from actually exponentially distributed data
x.exp <- rexp(n = 10, rate = 5)
mean(x.exp)
stats::sd(x.exp)
q <- quantile(x.exp, c(0.025, 0.5, 0.975))
old.par <- graphics::par(mfrow = c(2, 3))
get.norm.sd(q = q)
get.norm.sd(q = q, fit.weights = c(1, 10, 1))
get.norm.sd(q = q, fit.weights = c(10, 1, 10))
get.norm.sd(q = q, fit.weights = c(1, 100, 1))
get.norm.sd(q = q, fit.weights = c(100, 1, 100))
graphics::par(old.par)

## other examples
q <- stats::qnorm(p = c(0.025, 0.5, 0.975), mean = 1, sd = 1)
get.norm.sd(q = q)

q <- stats::qnorm(p = c(0.025, 0.5, 0.975), mean = 1, sd = 0.5)
get.norm.sd(q = q)

q <- stats::qnorm(p = c(0.025, 0.5, 0.975), mean = 0.01, sd = 0.1)
get.norm.sd(q = q)
```

`get.pert.par` *Fitting parameters of a pert distribution from four or more quantiles*

Description

`get.pert.par` returns the parameters of a pert distribution where the pth percentiles match with the quantiles q.

Usage

```
get.pert.par(p = c(0.025, 0.5, 0.6, 0.975), q, show.output = TRUE,
   plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

p	numeric, single value or vector of probabilities.
q	numeric, single value or vector of quantiles corresponding to p.
show.output	logical, if TRUE the optim result will be printed (default value is TRUE).
plot	logical, if TRUE the graphical diagnostics will be plotted (default value is TRUE).
tol	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is 0.001).

fit.weights	numerical vector of the same length as a probabilities vector p containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least three. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.pert.par uses the R function optim with the method L-BFGS-B. If this method fails the optimization method BFGS will be invoked.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a pert distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See ppert from the package **mc2d** for distribution implementation details.

Examples

```
q <- mc2d::qpert(p = c(0.025, 0.5, 0.6, 0.975), min = 0, mode = 3, max = 10, shape = 5)
old.par <- graphics::par(mfrow = c(2, 3))
get.pert.par(q = q)
get.pert.par(q = q, fit.weights = c(100, 1, 1, 100))
get.pert.par(q = q, fit.weights = c(10, 1, 1, 10))
get.pert.par(q = q, fit.weights = c(1, 100, 1, 1))
get.pert.par(q = q, fit.weights = c(1, 10, 1, 1))
graphics::par(old.par)

q <- mc2d::qpert(p = c(0.025, 0.5, 0.6, 0.975), min = 1, mode = 5, max = 10, shape = 4)
old.par <- graphics::par(mfrow = c(2, 3))
get.pert.par(q = q)
get.pert.par(q = q, scaleX = c(0.0001, 0.999999))
get.pert.par(q = q, fit.weights = c(100, 1, 1, 100))
get.pert.par(q = q, fit.weights = c(10, 1, 1, 10))
get.pert.par(q = q, fit.weights = c(1, 100, 1, 1))
get.pert.par(q = q, fit.weights = c(1, 10, 1, 1))
graphics::par(old.par)

q <- mc2d::qpert(p = c(0.025, 0.5, 0.6, 0.975), min=-10, mode = 5, max = 10, shape = 4)
old.par <- graphics::par(mfrow = c(2, 3))
get.pert.par(q = q)
get.pert.par(q = q, fit.weights = c(100, 1, 1, 100))
get.pert.par(q = q, fit.weights = c(10, 1, 1, 10))
get.pert.par(q = q, fit.weights = c(1, 100, 1, 1))
get.pert.par(q = q, fit.weights = c(1, 10, 1, 1))
graphics::par(old.par)

q <- mc2d::qpert(p = c(0.025, 0.5, 0.6, 0.975), min=-10, mode = 5, max = 10, shape = 0.4)
old.par <- graphics::par(mfrow = c(2, 3))
get.pert.par(q = q)
get.pert.par(q = q, fit.weights = c(100, 1, 1, 100))
get.pert.par(q = q, fit.weights = c(10, 1, 1, 10))
get.pert.par(q = q, fit.weights = c(1, 100, 1, 1))
get.pert.par(q = q, fit.weights = c(1, 10, 1, 1))
```

```
graphics::par(old.par)
```

`get.pois.par` *Fitting parameter of Poisson distribution from one or more quantiles*

Description

`get.pois.par` returns the parameters of a Poisson distribution where the pth percentiles match with the quantiles `q`.

Usage

```
get.pois.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
   plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

`p`	numeric, single value or vector of probabilities
`q`	numeric, single value or vector of quantiles corresponding to p
`show.output`	logical, if `TRUE` the `optim` result will be printed (default value is `TRUE`)
`plot`	logical, if `TRUE` the graphical diagnostics will be plotted (default value is `TRUE`)
`tol`	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is `0.001`)
`fit.weights`	numerical vector of the same length as a probabilities vector `p` containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
`scaleX`	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if `plot = TRUE`). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
`...`	further arguments passed to the functions `plot` and `points` (relevant only if `plot = TRUE`)

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least one. Using the default `p`, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. `get.pois.par` uses the R function `optim` with the method `L-BFGS-B`.

If `show.output = TRUE` the output of the function `optim` will be shown. The item `convergence` equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item `value` displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a Poisson distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because it is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See ppois for distribution implementation details.

Examples

```
q <- stats::qpois(p = c(0.025, 0.5, 0.975), lambda = 3)
old.par <- graphics::par(mfrow = c(2, 3))
get.pois.par(q = q)
get.pois.par(q = q, fit.weights = c(100, 1, 100))
get.pois.par(q = q, fit.weights = c(10, 1, 10))
get.pois.par(q = q, fit.weights = c(1, 100, 1))
get.pois.par(q = q, fit.weights = c(1, 10, 1))
```

```
graphics::par(old.par)

q <- stats::qpois(p = c(0.025, 0.5, 0.975), lambda = 4)
old.par <- graphics::par(mfrow = c(2, 3))
get.pois.par(q = q)
get.pois.par(q = q, fit.weights = c(100, 1, 100))
get.pois.par(q = q, fit.weights = c(10, 1, 10))
get.pois.par(q = q, fit.weights = c(1, 100, 1))
get.pois.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qpois(p = c(0.025, 0.5, 0.975), lambda = 0.5)
old.par <- graphics::par(mfrow = c(2, 3))
get.pois.par(q = q, tol = 1)
get.pois.par(q = q, fit.weights = c(100, 1, 100), tol = 1)
get.pois.par(q = q, fit.weights = c(10, 1, 10), tol = 1)
get.pois.par(q = q, fit.weights = c(1, 100, 1))
get.pois.par(q = q, fit.weights = c(1, 10, 1), tol = 0.01)
graphics::par(old.par)

q <- stats::qpois(p = c(0.025, 0.5, 0.975), lambda = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.pois.par(q = q, tol = 0.01)
get.pois.par(q = q, fit.weights = c(100, 1, 100), tol = 0.01)
get.pois.par(q = q, fit.weights = c(10, 1, 10), tol = 0.01)
get.pois.par(q = q, fit.weights = c(1, 100, 1))
get.pois.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)
```

`get.t.par` *Fitting parameter of a Student's t distribution from one or more quantiles*

Description

`get.t.par` returns the parameters of a Student's t distribution where the pth percentiles match with the quantiles q.

Usage

```
get.t.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
   plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

p	numeric, single value or vector of probabilities.
q	numeric, single value or vector of quantiles corresponding to p.
show.output	logical, if TRUE the optim result will be printed (default value is TRUE).

plot	logical, if TRUE the graphical diagnostics will be plotted (default value is TRUE).
tol	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is 0.001).
fit.weights	numerical vector of the same length as a probabilities vector p containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities and the number of quantiles must be identical and should be at least one. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.t.par uses the R function optim with the method L-BFGS-B. If this method fails the optimization method BFGS will be invoked.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a Student's t distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look

very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See `stats::pt` for distribution implementation details.

Examples

```
q <- stats::qt(p = c(0.025, 0.5, 0.975), df = 10)
old.par <- graphics::par(mfrow = c(2, 3))
get.t.par(q = q)
get.t.par(q = q, fit.weights = c(100, 1, 100))
get.t.par(q = q, fit.weights = c(10, 1, 10))
get.t.par(q = q, fit.weights = c(1, 100, 1))
get.t.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qt(p = c(0.025, 0.5, 0.975), df = 0.1)
old.par <- graphics::par(mfrow = c(2, 3))
get.t.par(q = q, scaleX = c(0.5, 0.5))
get.t.par(q = q, fit.weights = c(100, 1, 100), scaleX = c(0.5, 0.5))
get.t.par(q = q, fit.weights = c(10, 1, 10), scaleX = c(0.5, 0.5))
get.t.par(q = q, fit.weights = c(1, 100, 1), scaleX = c(0.5, 0.5))
get.t.par(q = q, fit.weights = c(1, 10, 1), scaleX = c(0.5, 0.5))
graphics::par(old.par)

q <- stats::qt(p = c(0.025, 0.5, 0.975), df = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.t.par(q = q, scaleX = c(0.5, 0.5))
get.t.par(q = q, fit.weights = c(100, 1, 100), scaleX = c(0.5, 0.5))
get.t.par(q = q, fit.weights = c(10, 1, 10), scaleX = c(0.5, 0.5))
get.t.par(q = q, fit.weights = c(1, 100, 1), scaleX = c(0.5, 0.5))
get.t.par(q = q, fit.weights = c(1, 10, 1), scaleX = c(0.5, 0.5))
graphics::par(old.par)

## example with only one quantile
q <- stats::qt(p = c(0.025), df = 3)
old.par <- graphics::par(mfrow = c(1, 3))
get.t.par(p = c(0.025), q = q)
get.t.par(p = c(0.025), q = q, fit.weights = 10)
```

```
get.t.par(p = c(0.025), q = q, fit.weights = 100)
graphics::par(old.par)
```

`get.tnorm.par` *Fitting parameters of truncated normal distribution from four or more quantiles*

Description

`get.tnorm.par` returns the parameters of a truncated normal distribution where the pth percentiles match with the quantiles `q`.

Usage

```
get.tnorm.par(p = c(0.025, 0.5, 0.75, 0.975), q, show.output = TRUE,
  plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

`p`	numeric, single value or vector of probabilities.
`q`	numeric, single value or vector of quantiles corresponding to p.
`show.output`	logical, if `TRUE` the `optim` result will be printed (default value is `TRUE`).
`plot`	logical, if `TRUE` the graphical diagnostics will be plotted (default value is `TRUE`).
`tol`	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is `0.001`).
`fit.weights`	numerical vector of the same length as a probabilities vector `p` containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
`scaleX`	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if `plot = TRUE`). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
`...`	further arguments passed to the functions `plot` and `points` (relevant only if `plot = TRUE`).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least four. Using the default `p`, the four corresponding quantiles are the 2.5th percentile, the median, the 75th percentile and the 97.5th percentile, respectively. `get.tnorm.par` uses the R function `optim` with the method `L-BFGS-B`. If this method fails the optimization method `Nelder-Mead` will be invoked.

If `show.output = TRUE` the output of the function `optim` will be shown. The item `convergence`

equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item `value` displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function `optim` are accepted if the achieved value of the minimized function (output component `value` of `optim`) is smaller than the argument `tol`.

The items of the probability vector `p` should lie between 0 and 1.

The items of the weighting vector `fit.weights` should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a truncated normal distribution or missing values (`NA`'s) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See `ptnomr` for distribution implementation details.

Examples

```
q <- msm::qtnorm(p = c(0.025, 0.5, 0.75, 0.975), mean = 3, sd = 3, lower = 0, upper = 10)
old.par <- graphics::par(mfrow = c(2, 3))
get.tnorm.par(q = q)
```

```
get.tnorm.par(q = q, scaleX = c(0.1, 0.999999))
get.tnorm.par(q = q, fit.weights = c(100, 1, 1, 100))
get.tnorm.par(q = q, fit.weights = c(10, 1, 1, 10))
get.tnorm.par(q = q, fit.weights = c(1, 100, 1, 1))
get.tnorm.par(q = q, fit.weights = c(1, 10, 1, 1))
graphics::par(old.par)

q <- msm::qtnorm(p = c(0.025, 0.5, 0.75, 0.975), mean = 3, sd = 0.1, lower=-1, upper = 4)
old.par <- graphics::par(mfrow = c(2, 3))
get.tnorm.par(q = q)
get.tnorm.par(q = q, fit.weights = c(100, 1, 1, 100))
get.tnorm.par(q = q, fit.weights = c(10, 1, 1, 10))
get.tnorm.par(q = q, fit.weights = c(1, 100, 1, 1))
get.tnorm.par(q = q, fit.weights = c(1, 10, 1, 1))
graphics::par(old.par)

q <- msm::qtnorm(p = c(0.025, 0.5, 0.75, 0.975), mean = 0, sd = 1, lower=-2, upper = 2)
old.par <- graphics::par(mfrow = c(2, 3))
get.tnorm.par(q = q)
get.tnorm.par(q = q, fit.weights = c(100, 1, 1, 100))
get.tnorm.par(q = q, fit.weights = c(10, 1, 1, 10))
get.tnorm.par(q = q, fit.weights = c(1, 100, 1, 1))
get.tnorm.par(q = q, fit.weights = c(1, 10, 1, 1))
graphics::par(old.par)
```

`get.triang.par` *Fitting parameters of a triangular distribution from three or more quantiles*

Description

`get.triang.par` returns the parameters of a triangular distribution where the pth percentiles match with the quantiles `q`.

Usage

```
get.triang.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
   plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

`p`	numeric, single value or vector of probabilities.
`q`	numeric, single value or vector of quantiles corresponding to p.
`show.output`	logical, if `TRUE` the `optim` result will be printed (default value is `TRUE`).
`plot`	logical, if `TRUE` the graphical diagnostics will be plotted (default value is `TRUE`).
`tol`	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is `0.001`).

fit.weights	numerical vector of the same length as a probabilities vector p containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.
scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least three. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.triang.par uses the R function optim with the method L-BFGS-B. If this method fails the optimization method BFGS will be invoked.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a triangular distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See ptriang from the package **mc2d** for distribution implementation details.

Examples

```
q <- mc2d::qtriang(p = c(0.025, 0.5, 0.975), min = 0, mode = 3, max = 10)
old.par <- graphics::par(mfrow = c(2, 3))
get.triang.par(q = q)
get.triang.par(q = q, fit.weights = c(100, 1, 100))
get.triang.par(q = q, fit.weights = c(10, 1, 10))
get.triang.par(q = q, fit.weights = c(1, 100, 1))
get.triang.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- mc2d::qtriang(p = c(0.025, 0.5, 0.975), min = 1, mode = 5, max = 10)
old.par <- graphics::par(mfrow = c(2, 3))
get.triang.par(q = q)
get.triang.par(q = q, scaleX = c(0.00001, 0.99999))
get.triang.par(q = q, fit.weights = c(100, 1, 100))
get.triang.par(q = q, fit.weights = c(10, 1, 10))
get.triang.par(q = q, fit.weights = c(1, 100, 1))
get.triang.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

## bad fit for negative values
q <- mc2d::qtriang(p = c(0.025, 0.5, 0.975), min=-20, mode = 5, max = 10)
old.par <- graphics::par(mfrow = c(2, 3))
get.triang.par(q = q, tol = 0.1)
get.triang.par(q = q)
get.triang.par(q = q, fit.weights = c(100, 1, 100))
get.triang.par(q = q, fit.weights = c(10, 1, 10))
get.triang.par(q = q, fit.weights = c(1, 100, 1), tol = 1)
get.triang.par(q = q, fit.weights = c(1, 10, 1), tol = 1)
graphics::par(old.par)

## other examples
q <- mc2d::qtriang(p = c(0.025, 0.5, 0.975), min=-20, mode = 5, max = 10)
get.triang.par(q = q, tol = 0.3)
```

`get.unif.par` *Fitting parameters of a uniform distribution from two or more quantiles*

Description

`get.unif.par` returns the parameters of a uniform distribution where the pth percentiles match with the quantiles `q`.

Usage

```
get.unif.par(p = c(0.025, 0.975), q, plot = TRUE, scaleX = c(0.1, 0.9), ...)
```

Arguments

`p`	numeric, single value or vector of probabilities.
`q`	numeric, single value or vector of quantiles corresponding to p.
`plot`	logical, if `TRUE` the graphical diagnostics will be plotted (default value is `TRUE`)
`scaleX`	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if `plot = TRUE`). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
`...`	further arguments passed to the functions `plot` and `points` (relevant only if `plot = TRUE`)

Details

The number of probabilities and the number of quantiles must be identical and should be at least two. Using the default `p`, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively.

Parameters of the uniform distribution are estimated exactly.

The items of the probability vector `p` should lie between 0 and 1.

Value

Returns fitted parameters of a uniform distribution.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a

"similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

Examples

```
q <- stats::qunif(p = c(0.025, 0.975), min = 0, max = 5)
get.unif.par(q = q)
get.unif.par(q = q, scaleX = c(0.001, 0.999))

q <- stats::qunif(p = c(0.025, 0.975), min=-6, max = 5)
get.unif.par(q = q)
```

---

get.weibull.par *Fitting parameters of a Weibull distribution from two or more quantiles*

Description

`get.weibull.par` returns the parameters of a Weibull distribution where the pth percentiles match with the quantiles `q`.

Usage

```
get.weibull.par(p = c(0.025, 0.5, 0.975), q, show.output = TRUE,
  plot = TRUE, tol = 0.001, fit.weights = rep(1, length(p)), scaleX = c(0.1, 0.9), ...)
```

Arguments

`p`	numeric, single value or vector of probabilities.
`q`	numeric, single value or vector of quantiles corresponding to p.
`show.output`	logical, if `TRUE` the `optim` result will be printed (default value is `TRUE`).
`plot`	logical, if `TRUE` the graphical diagnostics will be plotted (default value is `TRUE`).
`tol`	numeric, single positive value giving the absolute convergence tolerance for reaching zero (default value is `0.001`).
`fit.weights`	numerical vector of the same length as a probabilities vector `p` containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.

scaleX	numerical vector of the length 2 containing values (from the open interval (0, 1)) for scaling quantile-axis (relevant only if plot = TRUE). The smaller the left value, the further the graph is extrapolated within the lower percentile, the greater the right value, the further it goes within the upper percentile.
...	further arguments passed to the functions plot and points (relevant only if plot = TRUE).

Details

The number of probabilities, the number of quantiles and the number of weightings must be identical and should be at least two. Using the default p, the three corresponding quantiles are the 2.5th percentile, the median and the 97.5th percentile, respectively. get.weibull.par uses the R function optim with the method L-BFGS-B.

If show.output = TRUE the output of the function optim will be shown. The item convergence equal to 0 means the successful completion of the optimization procedure, otherwise it indicates a convergence error. The item value displays the achieved minimal value of the functions that were minimized.

The estimated distribution parameters returned by the function optim are accepted if the achieved value of the minimized function (output component value of optim) is smaller than the argument tol.

The items of the probability vector p should lie between 0 and 1.

The items of the weighting vector fit.weights should be positive values.

The function which will be minimized is defined as a sum of squared differences between the given probabilities and the theoretical probabilities of the specified distribution evaluated at the given quantile points (least squares estimation).

Value

Returns fitted parameters of a Weibull distribution or missing values (NA's) if the distribution cannot fit the specified quantiles.

Note

It should be noted that there might be deviations between the estimated and the theoretical distribution parameters in certain circumstances. This is because the estimation of the parameters is based on a numerical optimization method and depends strongly on the initial values. In addition, one must always keep in mind that a distribution for different combinations of parameters may look very similar. Therefore, the optimization method cannot always find the "right" distribution, but a "similar" one.

If the function terminates with the error massage "convergence error occurred or specified tolerance not achieved", one may try to set the convergence tolerance to a higher value. It is yet to be noted, that good till very good fits of parameters could only be obtained for tolerance values that are smaller than 0.001.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

See Also

See pweibull for distribution implementation details.

Examples

```
q <- stats::qweibull(p = c(0.025, 0.5, 0.975), shape = 0.01, scale = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.weibull.par(q = q, scaleX = c(0.1, 0.03))
get.weibull.par(q = q, fit.weights = c(100, 1, 100), scaleX = c(0.1, 0.99))
get.weibull.par(q = q, fit.weights = c(10, 1, 10))
get.weibull.par(q = q, fit.weights = c(1, 100, 1), scaleX = c(0.1, 0.03))
get.weibull.par(q = q, fit.weights = c(1, 10, 1), scaleX = c(0.1, 0.03))
graphics::par(old.par)

q <- stats::qweibull(p = c(0.025, 0.5, 0.975), shape = 0.1, scale = 0.1)
old.par <- graphics::par(mfrow = c(2, 3))
get.weibull.par(q = q, scaleX = c(0.1, 0.05))
get.weibull.par(q = q, fit.weights = c(100, 1, 100), scaleX = c(0.00000001, 0.99999999999))
get.weibull.par(q = q, fit.weights = c(10, 1, 10), scaleX = c(0.00000001, 0.99999999999))
get.weibull.par(q = q, fit.weights = c(1, 100, 1), scaleX = c(0.00000001, 0.01))
get.weibull.par(q = q, fit.weights = c(1, 10, 1), scaleX = c(0.00000001, 0.1))
graphics::par(old.par)

q <- stats::qweibull(p = c(0.025, 0.5, 0.975), shape = 2, scale = 3)
old.par <- graphics::par(mfrow = c(2, 3))
get.weibull.par(q = q)
get.weibull.par(q = q, fit.weights = c(100, 1, 100))
get.weibull.par(q = q, fit.weights = c(10, 1, 10))
get.weibull.par(q = q, fit.weights = c(1, 100, 1))
get.weibull.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

q <- stats::qweibull(p = c(0.025, 0.5, 0.975), shape = 1, scale = 1)
old.par <- graphics::par(mfrow = c(2, 3))
get.weibull.par(q = q)
get.weibull.par(q = q, fit.weights = c(100, 1, 100))
get.weibull.par(q = q, fit.weights = c(10, 1, 10))
get.weibull.par(q = q, fit.weights = c(1, 100, 1))
get.weibull.par(q = q, fit.weights = c(1, 10, 1))
graphics::par(old.par)

## example with only two quantiles
q <- stats::qweibull(p = c(0.025, 0.975), shape = 2, scale = 1)
old.par <- graphics::par(mfrow = c(1, 3))
get.weibull.par(p = c(0.025, 0.975), q = q)
get.weibull.par(p = c(0.025, 0.975), q = q, fit.weights = c(100, 1))
```

```
get.weibull.par(p = c(0.025, 0.975), q = q, fit.weights = c(10, 1))
graphics::par(old.par)
```

---

plotDiagnostics.perc *Graphical tools for choosing distribution by given quantiles*

Description

Diagnostic plot for choosing a most appropriate continuous probability for known quantiles

Usage

```
plotDiagnostics.perc(fit.results, tolPlot = 0.1)
```

Arguments

fit.results	a list containing fitting results as an output of the function rriskFitdist.perc.
tolPlot	numerical value, if the sums of the differences between the distribution percentiles and the given percentiles are smaller than this value, the distribution will be plotted.

Details

This function plots distribution whose percentiles go through the given percentiles q. The argument tolPlot controls this match.

Value

Only graphical output.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Kristin Tolksdorf <kristin.tolksdorf@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

Examples

```
p <- c(0.025, 0.5, 0.975)
q <- c(9.68, 29.20, 50.98)
fit.results1 <- rriskFitdist.perc(p = p, q = q, show.output = FALSE, tolConv = 0.5)
old.par <- graphics::par(mfrow = c(1, 2))
plotDiagnostics.perc(fit.results1)
plotDiagnostics.perc(fit.results1, tolPlot = 5)
graphics::par(old.par)

p <- c(0.2, 0.7)
```

```
q <- c(2.6, 19.1)
fit.results2 <- rriskFitdist.perc(p = p, q = q, show.output = FALSE)
plotDiagnostics.perc(fit.results2)

p <- c(0.3, 0.8, 0.9)
q <- c(10, 20, 40)
fit.results3 <- rriskFitdist.perc(p = p, q = q, show.output = FALSE)
plotDiagnostics.perc(fit.results3)

p <- c(0.3, 0.8, 0.9)
q <- c(10, 30, 40)
fit.results4 <- rriskFitdist.perc(p = p, q = q, show.output = FALSE)
plotDiagnostics.perc(fit.results4)

## Example with fitted beta pert distribution
p <- c(0.025, 0.5, 0.6, 0.975)
q <- mc2d::qpert(p = p, min = 0, mode = 3, max = 10, shape = 5)
fit.results5 <- rriskFitdist.perc(p = p, q = q, show.output = FALSE)
plotDiagnostics.perc(fit.results5)
```

`rriskFitdist.cont` *Fitting univariate distributions by maximum likelihood or by matching moments*

Description

Fits a univariate distribution by maximum likelihood or by matching moments.

Usage

```
rriskFitdist.cont(data, distr, method = c("mle", "mme"), start,
   chisqbreaks, meancount, ...)
```

Arguments

`data`	A numerical vector, data to be fitted.
`distr`	A character string `name` naming a distribution for which the corresponding density function `dname`, the corresponding distribution function `pname` and the corresponding quantile function must be defined, or directly the density function.
`method`	A character string coding for the fitting method: "mle" for the maximum likelihood estimation and "mme" for the matching moment estimation.
`start`	A named list giving the initial values of parameters of the named distribution. This argument will not be taken into account if `method = "mme"`, and may be omitted for some distributions for which reasonable starting values are computed if `method = "mle"`.

chisqbreaks	A numerical vector defining the breaks of the cell used to compute the chi-square statistic. If omitted, these breaks are automatically computed from the data in order to reach roughly the same number of observations per cell, roughly equal to the argument meancount, or slightly more if there are some ties.
meancount	The mean number of observations per cell expected for the definition of the breaks of the cells used to compute the chi-squared statistic.
...	further arguments to be passed to generic function, or to the function rriskMLEdist, in order to control the optimization method.

Details

This function is an alias of the function fitdist from the package **fitdistrplus** (Version 0.1-2). The original function was extended to fitting additional distributions. The following continuous distributions can be fitted by this function: normal, lognormal, logistic, exponential, F, Student's t, Beta, Cauchy, Weibull, Gamma.

For more details see the assistance page of the function fitdist from the package **fitdistrplus**.

This function is not intended to be called directly but is internally called in useFitdist.

Value

rriskFitdist.cont returns a list containing 19 items with following fitting results:

estimate	numeric, a single value or a vector containing estimated parameters.
method	the character string representing the used fitting method ("mle" or "mme").
sd	the estimated standard errors or NULL in case of the "mme" method.
cor	the estimated correlation matrix or NULL in case of the "mme" method.
loglik	the log-likelihood or NULL in case of the "mme" method.
aic	the Akaike information criterion or NULL in case of the "mme" method.
bic	the BIC or SBC (Schwarz Bayesian criterion) or NULL in case of the "mme" method.
n	the length of the data set.
data	the data set.
distname	the name of the estimated distribution.
chisq	the Chi-squared statistic or NULL, if not computed.
chisqbreaks	breaks used to define cells in the chi-square statistic.
chisqpvalue	p-value of the chi-square statistic or NULL, if not computed.
chisqdf	degree of freedom of the chi-square distribution or NULL, if not computed.
chisqtable	a table with observed and theoretical counts used for the Chi-squared calculations.
ad	the Anderson-Darling statistic or NULL, if not computed.
adtest	the decision of the Anderson-Darling test or NULL, if not computed.
ks	the Kolmogorov-Smirnov statistic or NULL, if not computed.
kstest	the decision of the Kolmogorov-Smirnov test or NULL, if not computed.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Kristin Tolksdorf <kristin.tolksdorf@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)
Marie-Laure Delignette-Muller (coauthor of the package **fitdistrplus**)
Christophe Dutang (coauthor of the package **fitdistrplus**)

Examples

```
set.seed(1)
x <- stats::rnorm(5000, mean = 10, sd = 5)
rriskFitdist.cont(x, "norm")
rriskFitdist.cont(x, "t")
```

`rriskFitdist.perc` *Fitting an amount of distribution families by given quantiles*

Description

This function fits the amount of distribution families to given quantiles and returns diagnostics that allow user to choose a most appropriate probability.

Usage

```
rriskFitdist.perc(p = c(0.025, 0.5, 0.975), q = c(9.68, 29.20, 50.98),
   show.output = TRUE, tolConv = 0.001, fit.weights = rep(1, length(p)))
```

Arguments

`p`	numerical vector giving probabilities.
`q`	numerical vector giving percentiles.
`show.output`	logical, if `TRUE` the `optim` result will be printed (default value is `TRUE`).
`tolConv`	positive numerical value, the absolute convergence tolerance for reaching zero by fitting distributions `get.norm.par` will be shown.
`fit.weights`	numerical vector of the same length as a probabilities vector `p` containing positive values for weighting quantiles. By default all quantiles will be weighted by 1.

Details

Both inputs `p` and `q` should be of the same length. The items of the probability vector `p` should lie between 0 and 1.

Value

Returns a list containing the data frame with the input vectors p and q and the results matrix giving fitted distributions, estimated parameters and a vector of theoretical percentiles calculated based on the estimated parameters. If the consistency check of input parameters fails the function returns NA.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Kristin Tolksdorf <kristin.tolksdorf@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

Examples

```
fit.results1 <- rriskFitdist.perc(show.output = FALSE)
fit.results1

fit.results2 <- rriskFitdist.perc(show.output = FALSE, tolConv = 0.6)
fit.results2

p <- c(0.2, 0.7)
q <- c(2.6, 19.1)
fit.results3 <- rriskFitdist.perc(p = p, q = q, show.output = FALSE)
fit.results3

p <- c(0.3, 0.8, 0.9)
q <- c(10, 20, 40)
fit.results4 <- rriskFitdist.perc(p = p, q = q, show.output = FALSE)
fit.results4

## Example with fitted pert distribution
p <- c(0.025, 0.5, 0.6, 0.975)
q <- mc2d::qpert(p = p, min = 0, mode = 3, max = 10, shape = 5)
fit.results5 <- rriskFitdist.perc(p = p, q = q, show.output = FALSE)
fit.results5
```

---

rriskMLEdist *Maximum likelihood fitting of univariate distributions*

Description

Fits a univariate distribution by maximum likelihood.

Usage

```
rriskMLEdist(data, distr, start = NULL, optim.method = "default",
   lower = -Inf, upper = Inf, custom.optim = NULL, ...)
```

Arguments

data	A numerical vector for non censored data or a dataframe of two columns respectively named left and right, describing each observed value as an interval for censored data. In that case the left column contains either NA for left censored observations, the left bound of the interval for interval censored observations, or the observed value for non-censored observations. The right column contains either NA for right censored observations, the right bound of the interval for interval censored observations, or the observed value for non-censored observations.
distr	A character string "name" naming a distribution (or directly the density function) for which the corresponding density function dname and the corresponding distribution pname must be classically defined. The possible values are: "norm", "exp", "lnorm", "logis", "gamma", "weibull", "beta", "chisq", "t", "f", "cauchy", "gompertz".
start	A named list giving the initial values of parameters of the named distribution. This argument may be omitted for some distributions for which reasonable starting values are computed (see details).
optim.method	"default" (see details) or optimization method to pass to optim.
lower	Left bounds on the parameters for the "L-BFGS-B" method (see optim).
upper	Right bounds on the parameters for the "L-BFGS-B" method (see optim).
custom.optim	a function carrying the MLE optimization (see details).
...	further arguments passed to the optim or custom.optim function.

Details

This function is an alias of the function mledist from the package **fitdistrplus** (Version 0.1-2). The original function was extended to fitting additional distributions. The following continuous distributions can be fitted by this function: normal, exponential, lognormal, logistic, gamma, Weibull, Beta, chi-square, Student's t, F, Cauchy, Gompertz and triangular. And the following discrete distributions can be fitted: (wird ergaenzt).

For more details see the assistance page of the function mledist from the package **fitdistrplus**.

This function is not intended to be called directly but is internally called in rriskFitdist.cont.

Value

rriskMLEdist returns a list with fitting results containing following informations

estimate	numeric, a single value or a vector containing estimated parameters.
convergence	an integer code for the convergence of optim. The value 0 indicates a successful convergence.
loglik	the log-likelihood
hessian	a symmetric matrix computed by optim as an estimate of the Hessian at the solution found or computed in the user-supplied optimization function. It is used in rriskFitdist.cont to estimate standard errors.
optim.function	the name of the optimization function used for maximum likelihood.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Kristin Tolksdorf <kristin.tolksdorf@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting),
Marie-Laure Delignette-Muller (coauthor of the package **fitdistrplus**),
Christophe Dutang (coauthor of the package **fitdistrplus**)

Examples

```
## a basic fit of some distribution with maximum likelihood estimation
set.seed(1)
x2 <- rchisq(500, 4)
rriskMLEdist(x2, "norm")
rriskMLEdist(x2, "exp")
rriskMLEdist(x2, "lnorm")
rriskMLEdist(x2, "logis")
rriskMLEdist(x2, "gamma")
rriskMLEdist(x2, "weibull")
#rriskMLEdist(x2, "beta")
rriskMLEdist(x2, "chisq")
rriskMLEdist(x2, "t")
rriskMLEdist(x2, "f")
rriskMLEdist(x2, "cauchy")
rriskMLEdist(x2, "gompertz")

## produces an error:
# rriskMLEdist(x2, "triang")
```

---

rriskMMEdist *Fitting univariate distributions by matching moments*

Description

Fits a univariate distribution by matching moments.

Usage

```
rriskMMEdist(data, distr)
```

Arguments

data	a numerical vector.
distr	A character string "name" naming a distribution or directly the density function dname. The estimated values of the distribution parameters are provided only for the following distributions : "norm", "lnorm", "exp", "pois", "gamma", "logis", "nbinom" , "geom", "beta" and "unif".

Details

This function is an alias of the function `mmedist` from the package **fitdistrplus** (Version 0.1-2). The original function was extended to fitting additional distributions. Parameter of the following distribution families can be estimated in this function: normal, lognormal, exponential, Poisson, gamma, logistic, negative binomial, geometric, Beta and continuous univariate.

For more details see the assistance page of the function `mmedist` from the package **fitdistrplus**.

This function is not intended to be called directly but is internally called in `rriskFitdist.cont`.

Value

`rriskMMEdist` returns the named parameter or a named vector of parameters.

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Kristin Tolksdorf <kristin.tolksdorf@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting),
Marie-Laure Delignette-Muller (coauthor of the package **fitdistrplus**),
Christophe Dutang (coauthor of the package **fitdistrplus**)

Examples

```
## Continuous distributions
set.seed(1)
x1 <- stats::rnorm(500, mean = 2, sd = 0.7)
rriskMMEdist(x1, "norm")
rriskMMEdist(x1, "exp")
rriskMMEdist(x1, "gamma")
rriskMMEdist(x1, "logis")
rriskMMEdist(x1, "unif")

## produces an error:
# rriskMMEdist(x1, "lnorm")
# rriskMMEdist(x1, "beta")

## Discrete distributions
set.seed(2)
x2 <- rpois(500, lambda = 3)
rriskMMEdist(x2, "pois")
rriskMMEdist(x2, "nbinom")
rriskMMEdist(x2, "geom")
```

useFitdist *Fitting amount continuous distributions to given univariate data.*

Description

Fitting amount continuous distributions to given univariate data

Usage

```
useFitdist(data2fit, show.output = TRUE, distributions)
```

Arguments

data2fit	numerical vector, data to be fitted.
show.output	logical value, if TRUE the output will be printed.
distributions	simple character or character vector giving the names of distribution families, that should be fitted to the data. The possible values are: norm, cauchy, logis, beta, exp, chisq, unif, gamma, lnorm, weibull, f, t, gompertz, triang.

Details

This function is not intended to be called directly but is internally called in fit.cont.

Value

Returns matrix with fitting results. More information...

Author(s)

Matthias Greiner <matthias.greiner@bfr.bund.de> (BfR),
Kristin Tolksdorf <kristin.tolksdorf@bfr.bund.de> (BfR),
Katharina Schueller <schueller@stat-up.de> (STAT-UP Statistical Consulting),
Natalia Belgorodski <belgorodski@stat-up.de> (STAT-UP Statistical Consulting)

Examples

```
x1 <- rgamma(374, 4,0.08)
res1 <- useFitdist(data2fit = x1)
res1

x2 <- rbeta(300, shape1 = 1, shape2 = 2)
res2 <- useFitdist(data2fit = x2)
res2
```

Index