Package ‘boodist’

August 10, 2023

Title Some Distributions from the 'Boost' Library and More

Version 1.0.0

Description Make some distributions from the 'C++' library 'Boost' available in 'R'. In addition, the normal-inverse Gaussian distribution and the generalized inverse Gaussian distribution are provided. The distributions are represented by 'R6' classes. The method to sample from the generalized inverse Gaussian distribution is the one given in ``Random variate generation for the generalized inverse Gaussian distribution'' Luc Devroye (2012) <doi:10.1007/s11222-012-9367-z>.

License GPL-3

URL `https://github.com/stla/boodist`

BugReports `https://github.com/stla/boodist/issues`

Imports R6, Rcpp, RcppNumerical, stats

LinkingTo BH, Rcpp, RcppEigen, RcppNumerical

Suggests plotly

Encoding UTF-8

RoxygenNote 7.2.3

SystemRequirements C++17

NeedsCompilation yes

Author Stéphane Laurent [aut, cre]

Maintainer Stéphane Laurent `<laurent_step@outlook.fr>`

Repository CRAN

Date/Publication 2023-08-10 10:00:02 UTC

R topics documented:

`findChi2df` *Find degrees of freedom*

Description

Find the degrees of freedom parameter of a non-central Chi-squared distribution given a quantile, its corresponding probability, and the non-centrality parameter.

Usage

```
findChi2df(ncp, q, p)
```

Arguments

`ncp`	non-centrality parameter, a non-negative number
`q`	a quantile
`p`	probability corresponding to the quantile `q`

Value

The degrees of freedom parameter of the non-central Chi-squared distribution with non-centrality parameter `ncp` and with cumulative probability `p` at the quantile `q`.

Examples

```
library(boodist)
nu <- findChi2df(ncp = 10, q = 3, p = 0.1)
pchisq(3, df = nu, ncp = 10) # should be 0.1
```

`findChi2ncp` *Find non-centrality parameter*

Description

Find the non-centrality parameter of a Chi-squared distribution given a quantile, its corresponding probability, and the degrees of freedom.

Usage

```
findChi2ncp(df, q, p)
```

Arguments

`df`	degrees of freedom, a positive number
`q`	a quantile
`p`	probability corresponding to the quantile `q`

Value

The non-centrality parameter of the Chi-squared distribution with degrees of freedom parameter `df` and with cumulative probability `p` at the quantile `q`.

Examples

```
library(boodist)
ncp <- findChi2ncp(df = 1, q = 3, p = 0.1)
pchisq(3, df = 1, ncp = ncp) # should be 0.1
```

`GeneralizedInverseGaussian`
Generalized inverse Gaussian distribution

Description

A R6 class to represent a generalized inverse Gaussian distribution.

Details

See Wikipedia.

Active bindings

`theta` Get or set the value of `theta`.

`eta` Get or set the value of `eta`.

`lambda` Get or set the value of `lambda`.

Methods

Public methods:

- `GeneralizedInverseGaussian$new()`
- `GeneralizedInverseGaussian$d()`
- `GeneralizedInverseGaussian$p()`
- `GeneralizedInverseGaussian$q()`
- `GeneralizedInverseGaussian$r()`
- `GeneralizedInverseGaussian$mean()`
- `GeneralizedInverseGaussian$mode()`
- `GeneralizedInverseGaussian$sd()`
- `GeneralizedInverseGaussian$variance()`
- `GeneralizedInverseGaussian$clone()`

Method `new()`**:** New generalized inverse Gaussian distribution.

Usage:

```
GeneralizedInverseGaussian$new(theta, eta, lambda)
```

Arguments:

`theta` concentration parameter, `>0`

`eta` scale parameter, `>0`

`lambda` parameter (denoted by `p` on Wikipedia)

Returns: A `GeneralizedInverseGaussian` object.

Method `d()`**:** Density function of the generalized inverse Gaussian distribution.

Usage:

```
GeneralizedInverseGaussian$d(x, log = FALSE)
```

Arguments:

`x` vector of positive numbers

`log` Boolean, whether to return the log-density

Returns: The density or the log-density evaluated at `x`.

Method `p()`**:** Cumulative distribution function of the generalized inverse Gaussian distribution.

Usage:

```
GeneralizedInverseGaussian$p(q)
```

Arguments:

`q` numeric vector of quantiles (`>= 0`)

Returns: The cumulative probabilities corresponding to `q`, with two attributes (see the **Note**).

Method `q()`**:** Quantile function of the generalized inverse Gaussian distribution.

Usage:

```
GeneralizedInverseGaussian$q(p, bounds = NULL)
```

Arguments:

`p` numeric vector of probabilities

`bounds` bounds enclosing the quantiles to be found (see the **Note**), or `NULL` for automatic bounds

Returns: The quantiles corresponding to `p`.

Method `r()`**:** Sampling from the generalized inverse Gaussian distribution.

Usage:

```
GeneralizedInverseGaussian$r(n)
```

Arguments:

`n` number of simulations

Returns: A numeric vector of length `n`.

Method `mean()`**:** Mean of the generalized inverse Gaussian distribution.

Usage:

```
GeneralizedInverseGaussian$mean()
```

Returns: The mean of the generalized inverse Gaussian distribution.

Method `mode()`**:** Mode of the generalized inverse Gaussian distribution.

Usage:

```
GeneralizedInverseGaussian$mode()
```

Returns: The mode of the generalized inverse Gaussian distribution.

Method `sd()`**:** Standard deviation of the generalized inverse Gaussian distribution.

Usage:

```
GeneralizedInverseGaussian$sd()
```

Returns: The standard deviation of the generalized inverse Gaussian distribution.

Method `variance()`**:** Variance of the generalized inverse Gaussian distribution.

Usage:

```
GeneralizedInverseGaussian$variance()
```

Returns: The variance of the generalized inverse Gaussian distribution.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GeneralizedInverseGaussian$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Note

The cumulative distribution function is evaluated by integrating the density function (in C++). Its returned value has two attributes: a numeric vector `"error_estimate"` and an integer vector `"error_code"`. The error code is 0 if no problem is detected. If an error code is not 0, a warning is thrown. The quantile function is evaluated by root-finding and then the user must provide some bounds enclosing the values of the quantiles or choose the automatic bounds. A maximum number of iterations is fixed in the root-finding algorithm. If it is reached, a warning is thrown.

Examples

```
if(require("plotly")) {
library(boodist)

x_ <- seq(0, 3, length.out = 100L)
lambda_ <- seq(-1, 1, length.out = 100L)
dsty <- vapply(lambda_, function(lambda) {
  GeneralizedInverseGaussian$new(theta = 1, eta = 1, lambda)$d(x_)
}, numeric(length(x_)))
#
txt <- matrix(NA_character_, nrow = length(x_), ncol = length(lambda_))
for(i in 1L:nrow(txt)) {
  for(j in 1L:ncol(txt)) {
    txt[i, j] <- paste0(
      "x: ", formatC(x_[i]),
      "<br> lambda: ", formatC(lambda_[j]),
      "<br> density: ", formatC(dsty[i, j])
    )
  }
}
#
plot_ly(
  x = ~lambda_, y = ~x_, z = ~dsty, type = "surface",
  text = txt, hoverinfo = "text", showscale = FALSE
) %>% layout(
  title = "Generalized inverse Gaussian distribution",
  margin = list(t = 40, r= 5, b = 5, l = 5),
  scene = list(
    xaxis = list(
      title = "lambda"
    ),
    yaxis = list(
      title = "x"
    ),
    zaxis = list(
      title = "density"
    )
  )
)
}
```

Gumbel *Gumbel distribution*

Description

A R6 class to represent a Gumbel distribution.

Details

See Wikipedia.

Active bindings

`a` Get or set the value of `a`.

`b` Get or set the value of `b`.

Methods

Public methods:

- `Gumbel$new()`
- `Gumbel$d()`
- `Gumbel$p()`
- `Gumbel$q()`
- `Gumbel$r()`
- `Gumbel$mean()`
- `Gumbel$median()`
- `Gumbel$mode()`
- `Gumbel$sd()`
- `Gumbel$variance()`
- `Gumbel$skewness()`
- `Gumbel$kurtosis()`
- `Gumbel$kurtosisExcess()`
- `Gumbel$clone()`

Method `new()`**:** New Gumbel distribution.

Usage:

```
Gumbel$new(a, b)
```

Arguments:

`a` location parameter

`b` scale parameter, `>0`

Returns: A `Gumbel` object.

Method `d()`**:** Density function of the Gumbel distribution.

Usage:

```
Gumbel$d(x, log = FALSE)
```

Arguments:

`x` numeric vector

`log` Boolean, whether to return the logarithm of the density

Returns: The density or the log-density evaluated at `x`.

Method `p()`**:** Cumulative distribution function of the Gumbel distribution.

Usage:

```
Gumbel$p(q, lower = TRUE)
```

Arguments:

`q` numeric vector of quantiles

`lower` Boolean, whether to deal with the lower tail

Returns: The cumulative probabilities corresponding to `q`.

Method `q()`**:** Quantile function of the Gumbel distribution.

Usage:

```
Gumbel$q(p, lower = TRUE)
```

Arguments:

`p` numeric vector of probabilities

`lower` Boolean, whether to deal with the lower tail

Returns: The quantiles corresponding to `p`.

Method `r()`**:** Sampling from the Gumbel distribution.

Usage:

```
Gumbel$r(n)
```

Arguments:

`n` number of simulations

Returns: A numeric vector of length `n`.

Method `mean()`**:** Mean of the Gumbel distribution.

Usage:

```
Gumbel$mean()
```

Returns: The mean of the Gumbel distribution.

Method `median()`**:** Median of the Gumbel distribution.

Usage:

```
Gumbel$median()
```

Returns: The median of the Gumbel distribution.

Method `mode()`**:** Mode of the Gumbel distribution.

Usage:

```
Gumbel$mode()
```

Returns: The mode of the Gumbel distribution.

Method `sd()`**:** Standard deviation of the Gumbel distribution.

Usage:

```
Gumbel$sd()
```

Returns: The standard deviation of the Gumbel distribution.

Method `variance()`**:** Variance of the Gumbel distribution.

Usage:

```
Gumbel$variance()
```

Returns: The variance of the Gumbel distribution.

Method `skewness()`**:** Skewness of the Gumbel distribution.

Usage:

```
Gumbel$skewness()
```

Returns: The skewness of the Gumbel distribution.

Method `kurtosis()`**:** Kurtosis of the Gumbel distribution.

Usage:

```
Gumbel$kurtosis()
```

Returns: The kurtosis of the Gumbel distribution.

Method `kurtosisExcess()`**:** Kurtosis excess of the Gumbel distribution.

Usage:

```
Gumbel$kurtosisExcess()
```

Returns: The kurtosis excess of the Gumbel distribution.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Gumbel$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Hyperexponential` *Hyperexponential distribution*

Description

A R6 class to represent a hyperexponential distribution.

Details

See Wikipedia.

Active bindings

`probs` Get or set the value of `probs`.

`rates` Get or set the value of `rates`.

Methods

Public methods:

- `Hyperexponential$new()`
- `Hyperexponential$d()`
- `Hyperexponential$p()`
- `Hyperexponential$q()`
- `Hyperexponential$r()`
- `Hyperexponential$mean()`
- `Hyperexponential$mode()`
- `Hyperexponential$sd()`
- `Hyperexponential$variance()`
- `Hyperexponential$skewness()`
- `Hyperexponential$kurtosis()`
- `Hyperexponential$kurtosisExcess()`
- `Hyperexponential$clone()`

Method `new()`: New hyperexponential distribution.

Usage:

```
Hyperexponential$new(probs, rates)
```

Arguments:

`probs`	probabilities (weights), a vector of positive numbers
`rates`	rate parameters, vector of positive numbers of the same length as the `probs` vector

Returns: A `Hyperexponential` object.

Method `d()`: Density function of the hyperexponential distribution.

Usage:

```
Hyperexponential$d(x)
```

Arguments:

`x`	vector of positive numbers

Returns: The density evaluated at `x`.

Method `p()`: Cumulative distribution function of the hyperexponential distribution.

Usage:

```
Hyperexponential$p(q, lower = TRUE)
```

Arguments:

`q`	numeric vector of quantiles
`lower`	Boolean, whether to deal with the lower tail

Returns: The cumulative probabilities corresponding to `q`.

Method `q()`: Quantile function of the hyperexponential distribution.

Usage:

`Hyperexponential$q(p, lower = TRUE)`

Arguments:

`p` numeric vector of probabilities

`lower` Boolean, whether to deal with the lower tail

Returns: The quantiles corresponding to `p`.

Method `r()`**:** Sampling from the hyperexponential distribution.

Usage:

`Hyperexponential$r(n)`

Arguments:

`n` number of simulations

Returns: A numeric vector of length `n`.

Method `mean()`**:** Mean of the hyperexponential distribution.

Usage:

`Hyperexponential$mean()`

Returns: The mean of the hyperexponential distribution.

Method `mode()`**:** Mode of the hyperexponential distribution.

Usage:

`Hyperexponential$mode()`

Returns: The mode of the hyperexponential distribution.

Method `sd()`**:** Standard deviation of the hyperexponential distribution.

Usage:

`Hyperexponential$sd()`

Returns: The standard deviation of the hyperexponential distribution.

Method `variance()`**:** Variance of the hyperexponential distribution.

Usage:

`Hyperexponential$variance()`

Returns: The variance of the hyperexponential distribution.

Method `skewness()`**:** Skewness of the hyperexponential distribution.

Usage:

`Hyperexponential$skewness()`

Returns: The skewness of the hyperexponential distribution.

Method `kurtosis()`**:** Kurtosis of the hyperexponential distribution.

Usage:

`Hyperexponential$kurtosis()`

Returns: The kurtosis of the hyperexponential distribution.

Method `kurtosisExcess()`**:** Kurtosis excess of the hyperexponential distribution.

Usage:

```
Hyperexponential$kurtosisExcess()
```

Returns: The kurtosis excess of the hyperexponential distribution.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Hyperexponential$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`InverseGamma` *Inverse Gamma distribution*

Description

A R6 class to represent an inverse Gamma distribution.

Details

See Wikipedia.

Active bindings

`alpha` Get or set the value of `alpha`.

`beta` Get or set the value of `beta`.

Methods

Public methods:

- `InverseGamma$new()`
- `InverseGamma$d()`
- `InverseGamma$p()`
- `InverseGamma$q()`
- `InverseGamma$r()`
- `InverseGamma$mean()`
- `InverseGamma$median()`
- `InverseGamma$mode()`
- `InverseGamma$sd()`
- `InverseGamma$variance()`
- `InverseGamma$skewness()`
- `InverseGamma$kurtosis()`
- `InverseGamma$kurtosisExcess()`

- `InverseGamma$clone()`

Method `new()`**:** New inverse Gamma distribution.

Usage:

```
InverseGamma$new(alpha, beta)
```

Arguments:

`alpha` shape parameter, `>0`

`beta` scale parameter, `>0`

Returns: An `inverseGamma` object.

Method `d()`**:** Density function of the inverse Gamma distribution.

Usage:

```
InverseGamma$d(x, log = FALSE)
```

Arguments:

`x` vector of positive numbers

`log` Boolean, whether to return the logarithm of the density

Returns: The density or the log-density evaluated at `x`.

Method `p()`**:** Cumulative distribution function of the inverse Gamma distribution.

Usage:

```
InverseGamma$p(q, lower = TRUE)
```

Arguments:

`q` numeric vector of quantiles

`lower` Boolean, whether to deal with the lower tail

Returns: The cumulative probabilities corresponding to `q`.

Method `q()`**:** Quantile function of the inverse Gamma distribution.

Usage:

```
InverseGamma$q(p, lower = TRUE)
```

Arguments:

`p` numeric vector of probabilities

`lower` Boolean, whether to deal with the lower tail

Returns: The quantiles corresponding to `p`.

Method `r()`**:** Sampling from the inverse Gamma distribution.

Usage:

```
InverseGamma$r(n)
```

Arguments:

`n` number of simulations

Returns: A numeric vector of length `n`.

Method `mean()`**:** Mean of the inverse Gamma distribution.

Usage:

```
InverseGamma$mean()
```

Returns: The mean of the inverse Gamma distribution.

Method `median()`**:** Median of the inverse Gamma distribution.

Usage:

```
InverseGamma$median()
```

Returns: The median of the inverse Gamma distribution.

Method `mode()`**:** Mode of the inverse Gamma distribution.

Usage:

```
InverseGamma$mode()
```

Returns: The mode of the inverse Gamma distribution.

Method `sd()`**:** Standard deviation of the inverse Gamma distribution.

Usage:

```
InverseGamma$sd()
```

Returns: The standard deviation of the inverse Gamma distribution.

Method `variance()`**:** Variance of the inverse Gamma distribution.

Usage:

```
InverseGamma$variance()
```

Returns: The variance of the inverse Gamma distribution.

Method `skewness()`**:** Skewness of the inverse Gamma distribution.

Usage:

```
InverseGamma$skewness()
```

Returns: The skewness of the inverse Gamma distribution.

Method `kurtosis()`**:** Kurtosis of the inverse Gamma distribution.

Usage:

```
InverseGamma$kurtosis()
```

Returns: The kurtosis of the inverse Gamma distribution.

Method `kurtosisExcess()`**:** Kurtosis excess of the inverse Gamma distribution.

Usage:

```
InverseGamma$kurtosisExcess()
```

Returns: The kurtosis excess of the inverse Gamma distribution.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
InverseGamma$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
if(require("plotly")) {
x_ <- seq(0, 2, length.out = 100L)
alpha_ <- seq(0.5, 2.5, length.out = 100L)
dsty <- vapply(alpha_, function(alpha) {
  InverseGamma$new(alpha, beta = 1)$d(x_)
}, numeric(length(x_)))
#
txt <- matrix(NA_character_, nrow = length(x_), ncol = length(alpha_))
for(i in 1L:nrow(txt)) {
  for(j in 1L:ncol(txt)) {
    txt[i, j] <- paste0(
      "x: ", formatC(x_[i]),
      "<br> alpha: ", formatC(alpha_[j]),
      "<br> density: ", formatC(dsty[i, j])
    )
  }
}
#
plot_ly(
  x = ~alpha_, y = ~x_, z = ~dsty, type = "surface",
  text = txt, hoverinfo = "text", showscale = FALSE
) %>% layout(
  title = "Inverse Gamma distribution",
  margin = list(t = 40, r= 5, b = 5, l = 5),
  scene = list(
    xaxis = list(
      title = "alpha"
    ),
    yaxis = list(
      title = "x"
    ),
    zaxis = list(
      title = "density"
    )
  )
)
}
```

`InverseGaussian` *Inverse Gaussian distribution*

Description

A R6 class to represent an inverse Gaussian distribution.

Details

See Wikipedia.

Active bindings

`mu` Get or set the value of `mu`.

`lambda` Get or set the value of `lambda`.

Methods

Public methods:

- `InverseGaussian$new()`
- `InverseGaussian$d()`
- `InverseGaussian$p()`
- `InverseGaussian$q()`
- `InverseGaussian$r()`
- `InverseGaussian$mean()`
- `InverseGaussian$median()`
- `InverseGaussian$mode()`
- `InverseGaussian$sd()`
- `InverseGaussian$variance()`
- `InverseGaussian$skewness()`
- `InverseGaussian$kurtosis()`
- `InverseGaussian$kurtosisExcess()`
- `InverseGaussian$clone()`

Method `new()`: New inverse Gaussian distribution.

Usage:

```
InverseGaussian$new(mu, lambda)
```

Arguments:

`mu` parameter, the mean, `>0`

`lambda` shape parameter, `>0`

Returns: An `inverseGaussian` object.

Method `d()`: Density function of the inverse Gaussian distribution.

Usage:

```
InverseGaussian$d(x, log = FALSE)
```

Arguments:

`x` vector of positive numbers

`log` Boolean, whether to return the logarithm of the density

Returns: The density or the log-density evaluated at `x`.

Method `p()`: Cumulative distribution function of the inverse Gaussian distribution.

Usage:

```
InverseGaussian$p(q, lower = TRUE)
```

Arguments:

`q` numeric vector of quantiles
`lower` Boolean, whether to deal with the lower tail

Returns: The cumulative probabilities corresponding to `q`.

Method `q()`**:** Quantile function of the inverse Gaussian distribution.

Usage:

```
InverseGaussian$q(p, lower = TRUE)
```

Arguments:

`p` numeric vector of probabilities
`lower` Boolean, whether to deal with the lower tail

Returns: The quantiles corresponding to `p`.

Method `r()`**:** Sampling from the inverse Gaussian distribution.

Usage:

```
InverseGaussian$r(n)
```

Arguments:

`n` number of simulations

Returns: A numeric vector of length `n`.

Method `mean()`**:** Mean of the inverse Gaussian distribution.

Usage:

```
InverseGaussian$mean()
```

Returns: The mean of the inverse Gaussian distribution.

Method `median()`**:** Median of the inverse Gaussian distribution.

Usage:

```
InverseGaussian$median()
```

Returns: The median of the inverse Gaussian distribution.

Method `mode()`**:** Mode of the inverse Gaussian distribution.

Usage:

```
InverseGaussian$mode()
```

Returns: The mode of the inverse Gaussian distribution.

Method `sd()`**:** Standard deviation of the inverse Gaussian distribution.

Usage:

```
InverseGaussian$sd()
```

Returns: The standard deviation of the inverse Gaussian distribution.

Method `variance()`**:** Variance of the inverse Gaussian distribution.

Usage:

```
InverseGaussian$variance()
```

Returns: The variance of the inverse Gaussian distribution.

Method `skewness()`**:** Skewness of the inverse Gaussian distribution.

Usage:

```
InverseGaussian$skewness()
```

Returns: The skewness of the inverse Gaussian distribution.

Method `kurtosis()`**:** Kurtosis of the inverse Gaussian distribution.

Usage:

```
InverseGaussian$kurtosis()
```

Returns: The kurtosis of the inverse Gaussian distribution.

Method `kurtosisExcess()`**:** Kurtosis excess of the inverse Gaussian distribution.

Usage:

```
InverseGaussian$kurtosisExcess()
```

Returns: The kurtosis excess of the inverse Gaussian distribution.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
InverseGaussian$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`NormalInverseGaussian` *Normal-inverse Gaussian distribution*

Description

A R6 class to represent a normal-inverse Gaussian distribution.

Details

See Wikipedia.

Active bindings

`mu` Get or set the value of `mu`.

`alpha` Get or set the value of `alpha`.

`beta` Get or set the value of `beta`.

`delta` Get or set the value of `delta`.

Methods

Public methods:

- `NormalInverseGaussian$new()`
- `NormalInverseGaussian$d()`
- `NormalInverseGaussian$p()`
- `NormalInverseGaussian$q()`
- `NormalInverseGaussian$r()`
- `NormalInverseGaussian$mean()`
- `NormalInverseGaussian$sd()`
- `NormalInverseGaussian$variance()`
- `NormalInverseGaussian$skewness()`
- `NormalInverseGaussian$kurtosis()`
- `NormalInverseGaussian$kurtosisExcess()`
- `NormalInverseGaussian$clone()`

Method `new()`: New normal-inverse Gaussian distribution.

Usage:

```
NormalInverseGaussian$new(mu, alpha, beta, delta)
```

Arguments:

`mu` location parameter

`alpha` tail heaviness parameter, `>0`

`beta` asymmetry parameter

`delta` scale parameter, `>0`

Returns: A `NormalInverseGaussian` object.

Method `d()`: Density function of the normal-inverse Gaussian distribution.

Usage:

```
NormalInverseGaussian$d(x, log = FALSE)
```

Arguments:

`x` numeric vector

`log` Boolean, whether to return the logarithm of the density

Returns: The density or the log-density evaluated at `x`.

Method `p()`: Cumulative distribution function of the normal-inverse Gaussian distribution.

Usage:

```
NormalInverseGaussian$p(q)
```

Arguments:

`q` numeric vector of quantiles

Returns: The cumulative probabilities corresponding to `q`, with two attributes (see the **Note**).

Method `q()`: Quantile function of the normal-inverse Gaussian distribution.

Usage:

```
NormalInverseGaussian$q(p, bounds = NULL)
```

Arguments:

`p` numeric vector of probabilities

`bounds` bounds enclosing the quantiles to be found (see the **Note**), or `NULL` for automatic bounds

Returns: The quantiles corresponding to `p`.

Method `r()`**:** Sampling from the normal-inverse Gaussian distribution.

Usage:

```
NormalInverseGaussian$r(n)
```

Arguments:

`n` number of simulations

Returns: A numeric vector of length `n`.

Method `mean()`**:** Mean of the normal-inverse Gaussian distribution.

Usage:

```
NormalInverseGaussian$mean()
```

Returns: The mean of the normal-inverse Gaussian distribution.

Method `sd()`**:** Standard deviation of the normal-inverse Gaussian distribution.

Usage:

```
NormalInverseGaussian$sd()
```

Returns: The standard deviation of the normal-inverse Gaussian distribution.

Method `variance()`**:** Variance of the normal-inverse Gaussian distribution.

Usage:

```
NormalInverseGaussian$variance()
```

Returns: The variance of the normal-inverse Gaussian distribution.

Method `skewness()`**:** Skewness of the normal-inverse Gaussian distribution.

Usage:

```
NormalInverseGaussian$skewness()
```

Returns: The skewness of the normal-inverse Gaussian distribution.

Method `kurtosis()`**:** Kurtosis of the normal-inverse Gaussian distribution.

Usage:

```
NormalInverseGaussian$kurtosis()
```

Returns: The kurtosis of the normal-inverse Gaussian distribution.

Method `kurtosisExcess()`**:** Kurtosis excess of the normal-inverse Gaussian distribution.

Usage:

```
NormalInverseGaussian$kurtosisExcess()
```

Returns: The kurtosis excess of the normal-inverse Gaussian distribution.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NormalInverseGaussian$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Note

The cumulative distribution function is evaluated by integrating the density function (in C++). Its returned value has two attributes: a numeric vector `"error_estimate"` and an integer vector `"error_code"`. The error code is 0 if no problem is detected. If an error code is not 0, a warning is thrown. The quantile function is evaluated by root-finding and then the user must provide some bounds enclosing the values of the quantiles or choose the automatic bounds. A maximum number of iterations is fixed in the root-finding algorithm. If it is reached, a warning is thrown.

`SkewNormal` *Skew normal distribution*

Description

A R6 class to represent a skew normal distribution.

Details

See Wikipedia.

Active bindings

`xi` Get or set the value of `xi`.

`omega` Get or set the value of `omega`.

`alpha` Get or set the value of `alpha`.

Methods

Public methods:

- `SkewNormal$new()`
- `SkewNormal$d()`
- `SkewNormal$p()`
- `SkewNormal$q()`
- `SkewNormal$r()`
- `SkewNormal$mean()`
- `SkewNormal$mode()`
- `SkewNormal$sd()`

- `SkewNormal$variance()`
- `SkewNormal$skewness()`
- `SkewNormal$kurtosis()`
- `SkewNormal$kurtosisExcess()`
- `SkewNormal$clone()`

Method `new()`**:** New skew normal distribution.

Usage:

```
SkewNormal$new(xi, omega, alpha)
```

Arguments:

`xi` location parameter

`omega` scale parameter, `>0`

`alpha` shape parameter

Returns: A `SkewNormal` object.

Method `d()`**:** Density function of the skew normal distribution.

Usage:

```
SkewNormal$d(x)
```

Arguments:

`x` numeric vector

Returns: The density evaluated at `x`.

Method `p()`**:** Cumulative distribution function of the skew normal distribution.

Usage:

```
SkewNormal$p(q, lower = TRUE)
```

Arguments:

`q` numeric vector of quantiles

`lower` Boolean, whether to deal with the lower tail

Returns: The cumulative probabilities corresponding to `q`.

Method `q()`**:** Quantile function of the skew normal distribution.

Usage:

```
SkewNormal$q(p, lower = TRUE)
```

Arguments:

`p` numeric vector of probabilities

`lower` Boolean, whether to deal with the lower tail

Returns: The quantiles corresponding to `p`.

Method `r()`**:** Sampling from the skew normal distribution.

Usage:

```
SkewNormal$r(n)
```

Arguments:

`n` number of simulations

Returns: A numeric vector of length `n`.

Method `mean()`**:** Mean of the skew normal distribution.

Usage:

```
SkewNormal$mean()
```

Returns: The mean of the skew normal distribution.

Method `mode()`**:** Mode of the skew normal distribution.

Usage:

```
SkewNormal$mode()
```

Returns: The mode of the skew normal distribution.

Method `sd()`**:** Standard deviation of the skew normal distribution.

Usage:

```
SkewNormal$sd()
```

Returns: The standard deviation of the skew normal distribution.

Method `variance()`**:** Variance of the skew normal distribution.

Usage:

```
SkewNormal$variance()
```

Returns: The variance of the skew normal distribution.

Method `skewness()`**:** Skewness of the skew normal distribution.

Usage:

```
SkewNormal$skewness()
```

Returns: The skewness of the skew normal distribution.

Method `kurtosis()`**:** Kurtosis of the skew normal distribution.

Usage:

```
SkewNormal$kurtosis()
```

Returns: The kurtosis of the skew normal distribution.

Method `kurtosisExcess()`**:** Kurtosis excess of the skew normal distribution.

Usage:

```
SkewNormal$kurtosisExcess()
```

Returns: The kurtosis excess of the skew normal distribution.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SkewNormal$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Student` *Non-central Student distribution*

Description

A R6 class to represent a non-central Student distribution.

Active bindings

`nu` Get or set the value of `nu`.

`delta` Get or set the value of `delta`.

Methods

Public methods:

- `Student$new()`
- `Student$d()`
- `Student$p()`
- `Student$q()`
- `Student$r()`
- `Student$mean()`
- `Student$median()`
- `Student$mode()`
- `Student$sd()`
- `Student$variance()`
- `Student$skewness()`
- `Student$kurtosis()`
- `Student$kurtosisExcess()`
- `Student$clone()`

Method `new()`: New Student distribution.

Usage:

```
Student$new(nu, delta)
```

Arguments:

`nu` degrees of freedom parameter, `>0`

`delta` non-centrality parameter

Returns: A `Student` object.

Method `d()`: Density function of the Student distribution.

Usage:

```
Student$d(x)
```

Arguments:

`x` numeric vector

Returns: The density evaluated at `x`.

Method `p()`**:** Cumulative distribution function of the Student distribution.

Usage:

```
Student$p(q, lower = TRUE)
```

Arguments:

`q` numeric vector of quantiles

`lower` Boolean, whether to deal with the lower tail

Returns: The cumulative probabilities corresponding to `q`.

Method `q()`**:** Quantile function of the Student distribution.

Usage:

```
Student$q(p, lower = TRUE)
```

Arguments:

`p` numeric vector of probabilities

`lower` Boolean, whether to deal with the lower tail

Returns: The quantiles corresponding to `p`.

Method `r()`**:** Sampling from the Student distribution.

Usage:

```
Student$r(n)
```

Arguments:

`n` number of simulations

Returns: A numeric vector of length `n`.

Method `mean()`**:** Mean of the Student distribution.

Usage:

```
Student$mean()
```

Returns: The mean of the Student distribution.

Method `median()`**:** Median of the Student distribution.

Usage:

```
Student$median()
```

Returns: The median of the Student distribution.

Method `mode()`**:** Mode of the Student distribution.

Usage:

```
Student$mode()
```

Returns: The mode of the Student distribution.

Method `sd()`**:** Standard deviation of the Student distribution.

Usage:

`Student$sd()`

Returns: The standard deviation of the Student distribution.

Method `variance()`**:** Variance of the Student distribution.

Usage:

`Student$variance()`

Returns: The variance of the Student distribution.

Method `skewness()`**:** Skewness of the Student distribution.

Usage:

`Student$skewness()`

Returns: The skewness of the Student distribution.

Method `kurtosis()`**:** Kurtosis of the Student distribution.

Usage:

`Student$kurtosis()`

Returns: The kurtosis of the Student distribution.

Method `kurtosisExcess()`**:** Kurtosis excess of the Student distribution.

Usage:

`Student$kurtosisExcess()`

Returns: The kurtosis excess of the Student distribution.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

`Student$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

Note

The non-centrality parameter of the Student distribution in the **stats** package is limited to `abs(ncp) <= 37.62`. The present implementation allows a larger range.

Index