Package ‘distr6’

February 22, 2022

Title The Complete R6 Probability Distributions Interface

Version 1.6.7


License MIT + file LICENSE

URL https://alan-turing-institute.github.io/distr6/,
https://github.com/alan-turing-institute/distr6/

BugReports https://github.com/alan-turing-institute/distr6/issues

Imports checkmate, data.table, param6 (>= 0.2.4), R6, Rcpp, set6 (>= 0.2.3), stats

Suggests actuar, cubature, extraDistr, GoFKernel, knitr, plotly, pracma, R62S3, rmarkdown, testthat

LinkingTo Rcpp

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.1.1

SystemRequirements C++11

Collate 'helpers.R' 'distr6Globals.R' 'Distribution.R'
'DistributionDecorator.R'
'DistributionDecorator_CoreStatistics.R'
R topics documented:

'skewType.R' 'sugar.R' 'zzz.R'

NeedsCompilation yes

Author Raphael Sonabend [aut, cre] (<https://orcid.org/0000-0001-9225-4654>),
Franz Kiraly [aut],
Peter Ruckdeschel [ctb] (Author of distr),
Matthias Kohl [ctb] (Author of distr),
Nurul Ain Toha [ctb],
Shen Chen [ctb],
Jordan Deenichin [ctb],
Chengyang Gao [ctb],
Chloe Zhaoyuan Gu [ctb],
Yunjie He [ctb],
Xiaowen Huang [ctb],
Shuhan Liu [ctb],
Runlong Yu [ctb],
Chijing Zeng [ctb],
Qian Zhou [ctb]

Maintainer Raphael Sonabend <raphaelsonabend@gmail.com>

Repository CRAN

Date/Publication 2022-02-22 03:20:02 UTC

R topics documented:

distr6-package ......................................................... 6
Arcsine ................................................................. 7
as.Distribution ......................................................... 11
as.MixtureDistribution ............................................ 12
as.ProductDistribution ........................................... 12
as.VectorDistribution ............................................. 13
Bernoulli ................................................................. 13
Beta .................................................................. 18
BetaNoncentral ....................................................... 22
Binomial ................................................................. 24
c.Distribution .......................................................... 28
Categorical ............................................................. 29
Cauchy ................................................................. 34
ChiSquared ............................................................. 39
ChiSquaredNoncentral ............................................... 43
Convolution ............................................................ 47
CoreStatistics ........................................................ 48
Cosine ................................................................. 52
decorate ................................................................. 54
Degenerate ............................................................. 55
Dirichlet ................................................................. 59
DiscreteUniform ...................................................... 62
distr6News ............................................................... 67
R topics documented:

- Distribution .................................................. 67
- DistributionDecorator ..................................... 77
- DistributionWrapper ....................................... 78
- distrSimulate ............................................... 80
- dstr .......................................................... 81
- Empirical ..................................................... 82
- EmpiricalMV .................................................. 87
- Epanechnikov ............................................... 90
- Erlang .......................................................... 92
- exkurtoisType .............................................. 96
- ExoticStatistics ............................................. 97
- Exponential .................................................. 101
- FDistribution ............................................... 106
- FDistributionNoncentral .................................. 110
- Frechet ........................................................ 113
- FunctionImputation ....................................... 117
- Gamma .......................................................... 118
- generalPNorm ............................................... 123
- Geometric ....................................................... 124
- Gompertz ......................................................... 129
- Gumbel .......................................................... 131
- huberize ........................................................ 136
- HuberizedDistribution .................................. 136
- Hypergeometric .............................................. 138
- InverseGamma ............................................... 142
- Kernel .......................................................... 146
- Laplace .......................................................... 148
- length.VectorDistribution .................................. 153
- lines.Distribution ......................................... 153
- listDecorators ............................................... 154
- listDistributions .......................................... 155
- listKernels ..................................................... 156
- listWrappers .................................................. 156
- Logistic .......................................................... 157
- LogisticKernel ............................................. 161
- Loglogistic ................................................... 165
- Loglogistic ................................................... 167
- Lognormal ...................................................... 171
- makeUniqueDistributions .................................. 176
- MixtureDistribution ...................................... 177
- mixturiseVector ............................................ 182
- Multinomial ................................................... 183
- MultivariateNormal ...................................... 188
- NegativeBinomial ......................................... 192
- Normal .......................................................... 197
- NormalKernel ............................................... 201
- Pareto ........................................................... 203
- plot.Distribution .......................................... 207
plot.VectorDistribution ......................................... 209
Poisson ............................................................ 210
ProductDistribution ............................................... 214
qqplot ............................................................... 220
Quartic ............................................................... 221
Rayleigh ............................................................. 223
rep.Distribution ................................................... 227
SDistribution ......................................................... 227
ShiftedLoglogistic .................................................. 228
Sigmoid ............................................................... 232
Silverman ............................................................ 234
simulateEmpiricalDistribution ................................. 236
skewType ............................................................ 236
StudentT ............................................................. 237
StudentTNoncentral ............................................... 241
testContinuous ...................................................... 244
testDiscrete .......................................................... 245
testDistribution .................................................... 245
testDistributionList ............................................... 246
testLeptokurtic ....................................................... 247
testMatrixvariate ................................................... 248
testMesokurtic ....................................................... 249
testMixture .......................................................... 250
testMultivariate ..................................................... 250
testNegativeSkew ................................................. 251
testNoSkew .......................................................... 252
testParameterSet .................................................... 253
testParameterSetList ............................................... 254
testPlatykurtic ....................................................... 255
testPositiveSkew .................................................... 256
testSymmetric ....................................................... 257
testUnivariate ....................................................... 257
Triangular ........................................................... 258
TriangularKernel .................................................... 264
Tricube .............................................................. 265
Triweight ............................................................ 267
truncate .............................................................. 269
TruncatedDistribution ............................................ 269
Uniform ............................................................. 271
UniformKernel ....................................................... 276
VectorDistribution ................................................ 277
Wald ................................................................. 286
Weibull ............................................................... 290
WeightedDiscrete ................................................. 294
 [.VectorDistribution ............................................. 299

Index ................................................................. 301
distr6: Object Oriented Distributions in R

Description

distr6 is an object oriented (OO) interface, primarily used for interacting with probability distributions in R. Additionally distr6 includes functionality for composite distributions, a symbolic representation for mathematical sets and intervals, basic methods for common kernels and numeric methods for distribution analysis. distr6 is the official R6 upgrade to the distr family of packages.

Details

The main features of distr6 are:

- Currently implements 45 probability distributions (and 11 Kernels) including all distributions in the R stats package. Each distribution has (where possible) closed form analytic expressions for basic statistical methods.
- Decorators that add further functionality to probability distributions including numeric results for useful modelling functions such as p-norms and k-moments.
- Wrappers for composite distributions including convolutions, truncation, mixture distributions and product distributions.

To learn more about distr6, start with the distr6 vignette:

vignette("distr6","distr6")

And for more advanced usage see the complete tutorials at https://alan-turing-institute.github.io/distr6/index.html #nolint

Author(s)

Maintainer: Raphael Sonabend <raphaelsonabend@gmail.com> (ORCID)

Authors:

- Franz Kiraly <f.kiraly@ucl.ac.uk>

Other contributors:

- Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de> (Author of distr) [contributor]
- Matthias Kohl <Matthias.Kohl@stamats.de> (Author of distr) [contributor]
- Nurul Ain Toha <nurul.toha.15@ucl.ac.uk> [contributor]
- Shen Chen <seanchen9832@icloud.com> [contributor]
- Jordan Deenichin <d.deenichin@gmail.com> [contributor]
- Chengyang Gao <garoc371@gmail.com> [contributor]
- Chloe Zhaoyuan Gu <guzhaoyuan@outlook.com> [contributor]
Arcsine Distribution Class

Description
Mathematical and statistical functions for the Arcsine distribution, which is commonly used in the study of random walks and as a special case of the Beta distribution.

Details
The Arcsine distribution parameterised with lower, $a$, and upper, $b$, limits is defined by the pdf,

$$f(x) = 1/(\pi \sqrt{(x-a)(b-x)})$$

for $-\infty < a \leq b < \infty$.

Value
Returns an R6 object inheriting from class SDistribution.

Distribution support
The distribution is supported on $[a, b]$.

Default Parameterisation
Arc(lower = 0, upper = 1)

Omitted Methods
N/A
Also known as
N/A

Super classes
distr6::Distribution -> distr6::SDistribution -> Arcsine

Public fields
name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.

Active bindings
properties Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:
• Arcsine$new()
• Arcsine$mean()
• Arcsine$mode()
• Arcsine$variance()
• Arcsine$skewness()
• Arcsine$kurtosis()
• Arcsine$entropy()
• Arcsine$pgf()
• Arcsine$clone()

Method new(): Creates a new instance of this R6 class.
Usage:
Arcsine$new(lower = NULL, upper = NULL, decorators = NULL)

Arguments:
lower (numeric(1))
Lower limit of the Distribution, defined on the Reals.
upper (numeric(1))
Upper limit of the Distribution, defined on the Reals.
decorators (character())
Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) * x \]

with an integration analogue for continuous distributions.
Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Arcsine$mode(which = "all")
Arguments:
which (character(1) | numeric(1))
Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
Arcsine$variance(...) 
Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[ sk_X = E_X \left( \frac{x - \mu}{\sigma} \right)^3 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:
Arcsine$skewness(...) 
Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left( \frac{x - \mu}{\sigma} \right)^4 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Arcsine$kurtosis(excess = TRUE, ...) 
Arguments:
excess (logical(1))
    If TRUE (default) excess kurtosis returned.
    ... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

\[- \sum (f_X) \log(f_X)\]

where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions.

Usage:
Arcsine$entropy(base = 2, ...)

Arguments:
base (integer(1))
    Base of the entropy logarithm, default = 2 (Shannon entropy)
    ... Unused.

Method pgf(): The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(z^X)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:
Arcsine$pgf(z, ...)

Arguments:
z (integer(1))
    \( z \) integer to evaluate probability generating function at.
    ... Unused.

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

Usage:
Arcsine$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References
Michael P. McLaughlin.

See Also
Other continuous distributions: BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared,
Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma,
Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal,
Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT,
Triangular, Uniform, Wald, Weibull
Other univariate distributions: Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

### Description

Coerces matrices to a VectorDistribution containing WeightedDiscrete distributions. Number of distributions are the number of rows in the matrix, number of x points are number of columns in the matrix.

### Usage

```r
as.Distribution(obj, fun, decorators = NULL)
```

## S3 method for class 'matrix'

```r
as.Distribution(obj, fun, decorators = NULL)
```

### Arguments

- `obj` *matrix*. Column names correspond to x in WeightedDiscrete, so this method only works if all distributions (rows in the matrix) have the same points to be evaluated on. Elements correspond to either the pdf or cdf of the distribution (see below).
- `fun` Either "pdf" or "cdf", passed to WeightedDiscrete and tells the constructor if the elements in obj correspond to the pdf or cdf of the distribution.
- `decorators` Passed to VectorDistribution.

### Value

A VectorDistribution

### Examples

```r
pdf <- runif(200)
mat <- matrix(pdf, 20, 10)
mat <- t(apply(mat, 1, function(x) x / sum(x)))
colnames(mat) <- 1:10
as.Distribution(mat, fun = "pdf")
```
as.MixtureDistribution

Coercion to Mixture Distribution

Description

Helper functions to quickly convert compatible objects to a MixtureDistribution.

Usage

as.MixtureDistribution(object, weights = "uniform")

Arguments

object ProductDistribution or VectorDistribution
weights (character(1)|numeric())
Weights to use in the resulting mixture. If all distributions are weighted equally then "uniform" provides a much faster implementation, otherwise a vector of length equal to the number of wrapped distributions, this is automatically scaled internally.

as.ProductDistribution

Coercion to Product Distribution

Description

Helper functions to quickly convert compatible objects to a ProductDistribution.

Usage

as.ProductDistribution(object)

Arguments

object MixtureDistribution or VectorDistribution
as.VectorDistribution  

Description
Helper functions to quickly convert compatible objects to a VectorDistribution.

Usage
as.VectorDistribution(object)

Arguments

object  MixtureDistribution or ProductDistribution

Bernoulli  

Bernoulli Distribution Class

Description
Mathematical and statistical functions for the Bernoulli distribution, which is commonly used to model a two-outcome scenario.

Details
The Bernoulli distribution parameterised with probability of success, p, is defined by the pmf,

\[ f(x) = p, \quad \text{if} \quad x = 1 \]

\[ f(x) = 1 - p, \quad \text{if} \quad x = 0 \]

for probability p.

Value
Returns an R6 object inheriting from class SDistribution.

Distribution support
The distribution is supported on \{0, 1\}.

Default Parameterisation
Bern(prob = 0.5)

Omitted Methods
N/A
Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Bernoulli

Public fields

- **name**  Full name of distribution.
- **short_name**  Short name of distribution for printing.
- **description**  Brief description of the distribution.
- **packages**  Packages required to be installed in order to construct the distribution.

Active bindings

- **properties**  Returns distribution properties, including skewness type and symmetry.

Methods

**Public methods:**

- Bernoulli$new()
- Bernoulli$mean()
- Bernoulli$mode()
- Bernoulli$median()
- Bernoulli$variance()
- Bernoulli$skewness()
- Bernoulli$kurtosis()
- Bernoulli$entropy()
- Bernoulli$mgf()
- Bernoulli$cf()
- Bernoulli$pgf()
- Bernoulli$clone()

**Method new():**  Creates a new instance of this R6 class.

*Usage:*

Bernoulli$new(prob = NULL, qprob = NULL, decorators = NULL)

*Arguments:*

- **prob**  (numeric(1))
  
  Probability of success.

- **qprob**  (numeric(1))
  
  Probability of failure. If provided then prob is ignored. qprob = 1 -prob.

- **decorators**  (character())
  
  Decorators to add to the distribution during construction.
Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) \cdot x \]

with an integration analogue for continuous distributions.

Usage:
Bernoulli$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Bernoulli$mode(which = "all")

Arguments:
which (character(1) | numeric(1)
Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method median(): Returns the median of the distribution. If an analytical expression is available returns distribution median, otherwise if symmetric returns self$mean, otherwise returns self$quantile(0.5).

Usage:
Bernoulli$median()

Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
Bernoulli$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[ sk_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3 \]

where \( E_X \) is the expectation of distribution X, \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:
Bernoulli$skewness(...)

Arguments:
Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left( \frac{x - \mu}{\sigma} \right)^4 \]

where \( E_X \) is the expectation of distribution X, \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Bernoulli$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

\[ -\sum (f_X) \log(f_X) \]

where \( f_X \) is the pdf of distribution X, with an integration analogue for continuous distributions.

Usage:
Bernoulli$entropy(base = 2, ...)

Arguments:
base (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method mgf(): The moment generating function is defined by

\[ mgf_X(t) = E_X [exp(xt)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
Bernoulli$mgf(t, ...)

Arguments:
t (integer(1))
  t integer to evaluate function at.
... Unused.

Method cf(): The characteristic function is defined by

\[ cf_X(t) = E_X [exp(xti)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
Bernoulli$cf(t, ...)
Arguments:

\( t \) (integer(1))
    \( t \) integer to evaluate function at.

... Unused.

Method \texttt{pgf}(): The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(z^t)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:

\texttt{Bernoulli$pgf(z, \ldots)}

Arguments:

\( z \) (integer(1))
    \( z \) integer to evaluate probability generating function at.

... Unused.

Method \texttt{clone}(): The objects of this class are cloneable with this method.

Usage:

\texttt{Bernoulli$clone(deep = FALSE)}

Arguments:

\( deep \) Whether to make a deep clone.

References

Michael P. McLaughlin.

See Also

Other discrete distributions: Binomial, Categorical, Degenerate, DiscreteUniform, EmpiricalMV,
Empirical, Geometric, Hypergeometric, Logarithmic, Multinomial, NegativeBinomial, WeightedDiscrete

Other univariate distributions: Arcsine, BetaNoncentral, Beta, Binomial, Categorical, Cauchy,
ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential,
FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel,
Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal,
NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral,
StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete
Beta Distribution Class

Description
Mathematical and statistical functions for the Beta distribution, which is commonly used as the prior in Bayesian modelling.

Details
The Beta distribution parameterised with two shape parameters, \( \alpha, \beta \), is defined by the pdf,

\[
f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}
\]

for \( \alpha, \beta > 0 \), where \( B \) is the Beta function.

Value
Returns an R6 object inheriting from class SDistribution.

Distribution support
The distribution is supported on \([0, 1]\).

Default Parameterisation
Beta(shape1 = 1, shape2 = 1)

Omitted Methods
N/A

Also known as
N/A

Super classes
distr6::Distribution -> distr6::SDistribution -> Beta

Public fields
name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.
Active bindings

properties Returns distribution properties, including skewness type and symmetry.

Methods

**Public methods:**

- `Beta$new()`
- `Beta$mean()`
- `Beta$mode()`
- `Beta$variance()`
- `Beta$skewness()`
- `Beta$kurtosis()`
- `Beta$entropy()`
- `Beta$pgf()`
- `Beta$clone()`

**Method `new()`:** Creates a new instance of this R6 class.

*Usage:*

```r
Beta$new(shape1 = NULL, shape2 = NULL, decorators = NULL)
```

*Arguments:*

- `shape1` (numeric(1))
  
  First shape parameter, shape1 > 0.

- `shape2` (numeric(1))
  
  Second shape parameter, shape2 > 0.

- `decorators` (character())
  
  Decorators to add to the distribution during construction.

**Method `mean()`:** The arithmetic mean of a (discrete) probability distribution X is the expectation

\[
E_X(X) = \sum p_X(x) \cdot x
\]

with an integration analogue for continuous distributions.

*Usage:*

```r
Beta$mean(...)  
```

*Arguments:*

- `...`: Unused.

**Method `mode()`:** The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

*Usage:*

```r
Beta$mode(which = "all")
```

*Arguments:*

- `which` (character(1) | numeric(1))
  
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.
Method \texttt{variance()}: The variance of a distribution is defined by the formula

$$
\text{var}_X = E[X^2] - E[X]^2
$$

where $E_X$ is the expectation of distribution $X$. If the distribution is multivariate the covariance matrix is returned.

\textit{Usage}:
\texttt{Beta$\text{variance}(...)}$

\textit{Arguments}:
... Unused.

Method \texttt{skewness()}: The skewness of a distribution is defined by the third standardised moment,

$$
\text{sk}_X = E_X\left[\frac{x - \mu}{\sigma}^3\right]
$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution.

\textit{Usage}:
\texttt{Beta$\text{skewness}(...)}$

\textit{Arguments}:
... Unused.

Method \texttt{kurtosis()}: The kurtosis of a distribution is defined by the fourth standardised moment,

$$
\text{k}_X = E_X\left[\frac{x - \mu}{\sigma}^4\right]
$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

\textit{Usage}:
\texttt{Beta$\text{kurtosis}(\text{excess = TRUE, \ldots})$}

\textit{Arguments}:
\texttt{excess (logical(1))}
  If \texttt{TRUE} (default) excess kurtosis returned.
... Unused.

Method \texttt{entropy()}: The entropy of a (discrete) distribution is defined by

$$
- \sum (f_X)\log(f_X)
$$

where $f_X$ is the pdf of distribution $X$, with an integration analogue for continuous distributions.

\textit{Usage}:
\texttt{Beta$\text{entropy}(\text{base = 2, \ldots})$}

\textit{Arguments}:
\texttt{base (integer(1))}
  Base of the entropy logarithm, default = 2 (Shannon entropy)
Method `pgf()`: The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(z^x)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:

Beta$pgf(z, ...)

Arguments:

- `z` (integer(1))
  - \( z \) integer to evaluate probability generating function at.

... Unused.

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

Usage:

Beta$clone(deep = FALSE)

Arguments:

- `deep` Whether to make a deep clone.

References


See Also

Other continuous distributions: Arcsine, BetaNoncentral, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete
Description
Mathematical and statistical functions for the Noncentral Beta distribution, which is commonly used as the prior in Bayesian modelling.

Details
The Noncentral Beta distribution parameterised with two shape parameters, $\alpha, \beta$, and location, $\lambda$, is defined by the pdf,

$$f(x) = \exp(-\lambda/2) \sum_{r=0}^{\infty} \left( \frac{\lambda/2}{r!} \right) \frac{(x^{\alpha} r^{-1} (1-x)^{\beta-1})}{B(\alpha + r, \beta)}$$

for $\alpha, \beta > 0, \lambda \geq 0$, where $B$ is the Beta function.

Value
Returns an R6 object inheriting from class SDistribution.

Distribution support
The distribution is supported on $[0, 1]$.

Default Parameterisation
BetaNC(shape1 = 1, shape2 = 1, location = 0)

Omitted Methods
N/A

Also known as
N/A

Super classes
distr6::Distribution -> distr6::SDistribution -> BetaNoncentral

Public fields
name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.
Active bindings

properties Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:

• BetaNoncentral$new()
• BetaNoncentral$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
BetaNoncentral$new(
  shape1 = NULL,
  shape2 = NULL,
  location = NULL,
  decorators = NULL
)

Arguments:
shape1 (numeric(1))
  First shape parameter, shape1 > 0.
shape2 (numeric(1))
  Second shape parameter, shape2 > 0.
location (numeric(1))
  Location parameter, defined on the non-negative Reals.
decorators (character())
  Decorators to add to the distribution during construction.

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

Usage:
BetaNoncentral$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

Author(s)

Jordan Deenichin

References

Binomial Distribution Class

Description

Mathematical and statistical functions for the Binomial distribution, which is commonly used to model the number of successes out of a number of independent trials.

Details

The Binomial distribution parameterised with number of trials, n, and probability of success, p, is defined by the pmf,

\[ f(x) = \binom{n}{x} p^x (1 - p)^{n-x} \]

for \( n = 0, 1, 2, \ldots \) and probability \( p \), where \( \binom{a}{b} \) is the combination (or binomial coefficient) function.

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on 0, 1, ..., \( n \).

Default Parameterisation

\text{Binom(size = 10, prob = 0.5)}

Omitted Methods

N/A
Binomial

Also known as
N/A

Super classes
distr6::Distribution -> distr6::SDistribution -> Binomial

Public fields
name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.

Active bindings
properties Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:
• Binomial$new()
• Binomial$mean()
• Binomial$mode()
• Binomial$variance()
• Binomial$skewness()
• Binomial$kurtosis()
• Binomial$entropy()
• Binomial$mgf()
• Binomial$cf()
• Binomial$pgf()
• Binomial$clone()

Method new(): Creates a new instance of this R6 class.
Usage:
Binomial$new(size = NULL, prob = NULL, qprob = NULL, decorators = NULL)

Arguments:
size (integer(1))
  Number of trials, defined on the positive Naturals.
prob (numeric(1))
  Probability of success.
qprob (numeric(1))
  Probability of failure. If provided then prob is ignored. qprob = 1 -prob.
decorators (character())
  Decorators to add to the distribution during construction.
\textbf{Method} \texttt{mean()}: The arithmetic mean of a (discrete) probability distribution $X$ is the expectation

$$E_X(X) = \sum p_X(x) * x$$

with an integration analogue for continuous distributions.

\textit{Usage:}
\begin{verbatim}
Binomial$mean(...)
\end{verbatim}

\textit{Arguments:}
... Unused.

\textbf{Method} \texttt{mode()}: The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

\textit{Usage:}
\begin{verbatim}
Binomial$mode(which = "all")
\end{verbatim}

\textit{Arguments:}
\begin{verbatim}
which (character(1) | numeric(1))
\end{verbatim}

Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

\textbf{Method} \texttt{variance()}: The variance of a distribution is defined by the formula

$$var_X = E[X^2] - E[X]^2$$

where $E_X$ is the expectation of distribution $X$. If the distribution is multivariate the covariance matrix is returned.

\textit{Usage:}
\begin{verbatim}
Binomial$variance(...)
\end{verbatim}

\textit{Arguments:}
... Unused.

\textbf{Method} \texttt{skewness()}: The skewness of a distribution is defined by the third standardised moment,

$$sk_X = E_X\left[\frac{x - \mu^3}{\sigma}\right]$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution.

\textit{Usage:}
\begin{verbatim}
Binomial$skewness(...)
\end{verbatim}

\textit{Arguments:}
... Unused.

\textbf{Method} \texttt{kurtosis()}: The kurtosis of a distribution is defined by the fourth standardised moment,

$$k_X = E_X\left[\frac{x - \mu^4}{\sigma}\right]$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.
Binomial

Usage:
Binomial$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

\[- \sum (f_X) log(f_X)\]

where $f_X$ is the pdf of distribution $X$, with an integration analogue for continuous distributions.

Usage:
Binomial$entropy(base = 2, ...)

Arguments:
base (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method mgf(): The moment generating function is defined by

\[ mgf_X(t) = E_X[exp(t x)] \]

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:
Binomial$mgf(t, ...)

Arguments:
t (integer(1))
  t integer to evaluate function at.
... Unused.

Method cf(): The characteristic function is defined by

\[ cf_X(t) = E_X[exp(t i x)] \]

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:
Binomial$cf(t, ...)

Arguments:
t (integer(1))
  t integer to evaluate function at.
... Unused.

Method pgf(): The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(z x)] \]

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$. 
Usage:
Binomial$pgf(z, ...)

Arguments:
z (integer(1))
  z integer to evaluate probability generating function at.
... Unused.

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

Usage:
Binomial$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References
Michael P. McLaughlin.

See Also
Other discrete distributions: Bernoulli, Categorical, Degenerate, DiscreteUniform, EmpiricalMV,
Empirical, Geometric, Hypergeometric, Logarithmic, Multinomial, NegativeBinomial, WeightedDiscrete
Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Categorical, Cauchy,
ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential,
FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel,
Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal,
NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral,
StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

---

c.Distribution  

Combine Distributions into a VectorDistribution

Description
Helper function for quickly combining distributions into a VectorDistribution.

Usage
## S3 method for class 'Distribution'
c(..., name = NULL, short_name = NULL, decorators = NULL)

Arguments
...
distributions to be concatenated.
name, short_name, decorators
See VectorDistribution
Categorical

Value

A VectorDistribution

See Also

VectorDistribution

Examples

# Construct and combine
c(Binomial$new(), Normal$new())

# More complicated distributions
b <- truncate(Binomial$new(), 2, 6)
n <- huberize(Normal$new(), -1, 1)
c(b, n)

# Concatenate VectorDistributions
v1 <- VectorDistribution$new(list(Binomial$new(), Normal$new()))
v2 <- VectorDistribution$new(
  distribution = "Gamma",
  params = data.table::data.table(shape = 1:2, rate = 1:2)
)
c(v1, v2)
**Distribution support**

The distribution is supported on \( x_1, ..., x_k \).

**Default Parameterisation**

\[
\text{Cat(elements = 1, probs = 1)}
\]

**Omitted Methods**

N/A

**Also known as**

N/A

**Super classes**

\[
distr6::Distribution \rightarrow distr6::SDistribution \rightarrow \text{Categorical}
\]

**Public fields**

- name: Full name of distribution.
- short_name: Short name of distribution for printing.
- description: Brief description of the distribution.

**Active bindings**

- properties: Returns distribution properties, including skewness type and symmetry.

**Methods**

**Public methods:**

- `Categorical$new()`
- `Categorical$mean()`
- `Categorical$mode()`
- `Categorical$variance()`
- `Categorical$skewness()`
- `Categorical$kurtosis()`
- `Categorical$entropy()`
- `Categorical$mgf()`
- `Categorical$cf()`
- `Categorical$pgf()`
- `Categorical$clone()`

**Method** `new()`: Creates a new instance of this R6 class.

*Usage:*

Categorical$new(elements = NULL, probs = NULL, decorators = NULL)
Arguments:
elements list()
   Categories in the distribution, see examples.
probs numeric()
   Probabilities of respective categories occurring.
decorators (character())
   Decorators to add to the distribution during construction.

Examples:
# Note probabilities are automatically normalised (if not vectorised)
x <- Categorical$new(elements = list("Bapple", "Banana", 2), probs = c(0.2, 0.4, 1))

# Length of elements and probabilities cannot be changed after construction
x$setParameterValue(probs = c(0.1, 0.2, 0.7))

# d/p/q/r
x$pdf(c("Bapple", "Carrot", 1, 2))
x$cdf("Banana") # Assumes ordered in construction
x$quantile(0.42) # Assumes ordered in construction
x$rand(10)

# Statistics
x$mode()

summary(x)

Method mean(): The arithmetic mean of a (discrete) probability distribution $X$ is the expectation

$$E_X(X) = \sum p_X(x) * x$$

with an integration analogue for continuous distributions.

Usage:
Categorical$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Categorical$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
   Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula

$$\text{var}_X = E[X^2] - E[X]^2$$
where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
Categorical$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[
sk_X = E_X \left[ \frac{x - \mu^3}{\sigma} \right]
\]

where \( E_X \) is the expectation of distribution X, \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:
Categorical$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[
k_X = E_X \left[ \frac{x - \mu^4}{\sigma} \right]
\]

where \( E_X \) is the expectation of distribution X, \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Categorical$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

\[-\sum (f_X) log(f_X)\]

where \( f_X \) is the pdf of distribution X, with an integration analogue for continuous distributions.

Usage:
Categorical$entropy(base = 2, ...)

Arguments:
base (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.
Method `mgf()`: The moment generating function is defined by

\[
mgf_X(t) = E_X[exp(xt)]
\]

where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

Usage:
Categorical$mgf(t, ...)

Arguments:
t (integer(1))
  t integer to evaluate function at.
... Unused.

Method `cf()`: The characteristic function is defined by

\[
cf_X(t) = E_X[exp(xti)]
\]

where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

Usage:
Categorical$cf(t, ...)

Arguments:
t (integer(1))
  t integer to evaluate function at.
... Unused.

Method `pgf()`: The probability generating function is defined by

\[
pgf_X(z) = E_X[exp(zi)]
\]

where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

Usage:
Categorical$pgf(z, ...)

Arguments:
z (integer(1))
  z integer to evaluate probability generating function at.
... Unused.

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

Usage:
Categorical$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References

See Also

Other discrete distributions: Bernoulli, Binomial, Degenerate, DiscreteUniform, EmpiricalMV, Empirical, Geometric, Hypergeometric, Logarithmic, Multinomial, NegativeBinomial, WeightedDiscrete

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

Examples

```r
# Note probabilities are automatically normalised (if not vectorised)
x <- Categorical$new(elements = list("Bapple", "Banana", 2), probs = c(0.2, 0.4, 1))

# Length of elements and probabilities cannot be changed after construction
x$setParameterValue(probs = c(0.1, 0.2, 0.7))

# d/p/q/r
x$pdf(c("Bapple", "Carrot", 1, 2))
x$cdf("Banana") # Assumes ordered in construction
x$quantile(0.42) # Assumes ordered in construction
x$rand(10)

# Statistics
x$mode()

summary(x)
```

---

Cauchy

**Cauchy Distribution Class**

Description

Mathematical and statistical functions for the Cauchy distribution, which is commonly used in physics and finance.

Details

The Cauchy distribution parameterised with location, \( \alpha \), and scale, \( \beta \), is defined by the pdf,

\[
f(x) = \frac{1}{\pi \beta (1 + ((x - \alpha) / \beta)^2)}
\]

for \( \alpha \in \mathbb{R} \) and \( \beta > 0 \).
Cauchy

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on the Reals.

Default Parameterisation

Cauchy(location = 0, scale = 1)

Omitted Methods

N/A

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Cauchy

Public fields

name  Full name of distribution.
short_name  Short name of distribution for printing.
description  Brief description of the distribution.
packages  Packages required to be installed in order to construct the distribution.

Methods

Public methods:

- Cauchy$new()
- Cauchy$mean()
- Cauchy$mode()
- Cauchy$variance()
- Cauchy$skewness()
- Cauchy$kurtosis()
- Cauchy$entropy()
- Cauchy$mgf()
- Cauchy$cf()
- Cauchy$pgf()
- Cauchy$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Cauchy$new(location = NULL, scale = NULL, decorators = NULL)

Arguments:
location (numeric(1))
  Location parameter defined on the Reals.
scale (numeric(1))
  Scale parameter defined on the positive Reals.
decorators (character())
  Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution \( X \) is the expectation

\[
E_X(X) = \sum p_X(x) \cdot x
\]

with an integration analogue for continuous distributions.

Usage:
Cauchy$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Cauchy$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula

\[
var_X = E[X^2] - E[X]^2
\]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
Cauchy$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[
sk_X = E_X \left[ \frac{x - \mu^3}{\sigma^3} \right]
\]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.
Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = \frac{E_X \left[ \left( \frac{x - \mu}{\sigma} \right)^4 \right]}{\sigma^4} \]

where \( E_X \) is the expectation of distribution X, \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Cauchy$kurtosis(excess = TRUE, ...)  
Arguments:  
excess (logical(1))  
If TRUE (default) excess kurtosis returned.  
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

\[ -\sum (f_X) \log(f_X) \]

where \( f_X \) is the pdf of distribution X, with an integration analogue for continuous distributions.

Usage:
Cauchy$entropy(base = 2, ...)  
Arguments:  
base (integer(1))  
Base of the entropy logarithm, default = 2 (Shannon entropy)  
... Unused.

Method mgf(): The moment generating function is defined by

\[ mgf_X(t) = E_X[exp(xt)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
Cauchy$mgf(t, ...)  
Arguments:  
t (integer(1))  
t integer to evaluate function at.  
... Unused.

Method cf(): The characteristic function is defined by

\[ cf_X(t) = E_X[exp(xti)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X.
Usage:
Cauchy$cf(t, \ldots)

Arguments:
t (integer(1))
  t integer to evaluate function at.
\ldots Unused.

Method pgf(): The probability generating function is defined by

\[ pgf_X(z) = E[X \exp(z^X)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
Cauchy$pgf(z, \ldots)

Arguments:
z (integer(1))
  z integer to evaluate probability generating function at.
\ldots Unused.

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

Usage:
Cauchy$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

Author(s)
Chijing Zeng

References

See Also
Other continuous distributions: Arcsine, BetaNoncentral, Beta, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull
Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithm, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete
ChiSquared  

Chi-Squared Distribution Class

Description

Mathematical and statistical functions for the Chi-Squared distribution, which is commonly used to model the sum of independent squared Normal distributions and for confidence intervals.

Details

The Chi-Squared distribution parameterised with degrees of freedom, \( \nu \), is defined by the pdf,

\[
 f(x) = \frac{(x^{\nu/2-1}\exp(-x/2))}{(2^{\nu/2}\Gamma(\nu/2))}
\]

for \( \nu > 0 \).

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on the Positive Reals.

Default Parameterisation

\( \text{ChiSq}(df = 1) \)

Omitted Methods

N/A

Also known as

N/A

Super classes

\[ \text{distr6::Distribution} \to \text{distr6::SDistribution} \to \text{ChiSquared} \]

Public fields

- name: Full name of distribution.
- short_name: Short name of distribution for printing.
- description: Brief description of the distribution.
- packages: Packages required to be installed in order to construct the distribution.
Active bindings properties Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:
- ChiSquared$new()
- ChiSquared$mean()
- ChiSquared$mode()
- ChiSquared$variance()
- ChiSquared$skewness()
- ChiSquared$kurtosis()
- ChiSquared$entropy()
- ChiSquared$mgf()
- ChiSquared$cf()
- ChiSquared$pgf()
- ChiSquared$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
ChiSquared$new(df = NULL, decorators = NULL)

Arguments:
df (integer(1))
- Degrees of freedom of the distribution defined on the positive Reals.
decorators (character())
- Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

$$ E_X(X) = \sum p_X(x) \cdot x $$

with an integration analogue for continuous distributions.

Usage:
ChiSquared$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
ChiSquared$mode(which = "all")

Arguments:
which (character(1) | numeric(1)
- Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.
Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
ChiSquared$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[ s_{kX} = E_X \left[ \frac{x - \mu}{\sigma}^3 \right] \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:
ChiSquared$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left[ \frac{x - \mu}{\sigma}^4 \right] \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
ChiSquared$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
   If TRUE (default) excess kurtosis returned.
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

\[-\sum (f_X) \log(f_X)\]

where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions.

Usage:
ChiSquared$entropy(base = 2, ...)

Arguments:
base (integer(1))
   Base of the entropy logarithm, default = 2 (Shannon entropy)
Method mgf(): The moment generating function is defined by

$$mgf_X(t) = E_X[exp(xt)]$$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:
ChiSquared$mgf(t, ...)

Arguments:
t (integer(1))
    t integer to evaluate function at.

... Unused.

Method cf(): The characteristic function is defined by

$$cf_X(t) = E_X[exp(xti)]$$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:
ChiSquared$cf(t, ...)

Arguments:
t (integer(1))
    t integer to evaluate function at.

... Unused.

Method pgf(): The probability generating function is defined by

$$pgf_X(z) = E_X[exp(xz^i)]$$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:
ChiSquared$pgf(z, ...)

Arguments:
z (integer(1))
    z integer to evaluate probability generating function at.

... Unused.

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

Usage:
ChiSquared$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References
ChiSquaredNoncentral

See Also
Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull
Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

---

ChiSquaredNoncentral  
Noncentral Chi-Squared Distribution Class

Description
Mathematical and statistical functions for the Noncentral Chi-Squared distribution, which is commonly used to model the sum of independent squared Normal distributions and for confidence intervals.

Details
The Noncentral Chi-Squared distribution parameterised with degrees of freedom, \( \nu \), and location, \( \lambda \), is defined by the pdf,

\[
f(x) = \exp(-\lambda/2) \sum_{r=0}^{\infty} \left( (\lambda/2)^r / r! \right) \left( x^{(\nu+2r)/2} - 1 \exp(-x/2) \right) / \left( 2^{(\nu+2r)/2} \Gamma((\nu + 2r)/2) \right)
\]

for \( \nu \geq 0, \lambda \geq 0 \).

Value
Returns an R6 object inheriting from class SDistribution.

Distribution support
The distribution is supported on the Positive Reals.

Default Parameterisation
ChiSqNC(df = 1, location = 0)

Omitted Methods
N/A
Also known as
N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> ChiSquaredNoncentral

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.

Active bindings

properties Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:
• ChiSquaredNoncentral$new()
• ChiSquaredNoncentral$mean()
• ChiSquaredNoncentral$variance()
• ChiSquaredNoncentral$skewness()
• ChiSquaredNoncentral$kurtosis()
• ChiSquaredNoncentral$mgf()
• ChiSquaredNoncentral$cf()
• ChiSquaredNoncentral$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
ChiSquaredNoncentral$new(df = NULL, location = NULL, decorators = NULL)

Arguments:
df (integer(1))
  Degrees of freedom of the distribution defined on the positive Reals.
location (numeric(1))
  Location parameter, defined on the non-negative Reals.
decorators (character())
  Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) \ast x \]

with an integration analogue for continuous distributions.
Usage:
ChiSquaredNoncentral$mean(...)

Arguments:
... Unused.

Method variance(): The variance of a distribution is defined by the formula

$$\text{var}_X = E[X^2] - E[X]^2$$

where $E_X$ is the expectation of distribution $X$. If the distribution is multivariate the covariance matrix is returned.

Usage:
ChiSquaredNoncentral$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

$$sk_X = E_X\left[\frac{x - \mu}{\sigma}^3\right]$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution.

Usage:
ChiSquaredNoncentral$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

$$k_X = E_X\left[\frac{x - \mu}{\sigma}^4\right]$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
ChiSquaredNoncentral$kurtosis(excess = \text{TRUE}, ...)

Arguments:
excess (logical(1))

If \text{TRUE} (default) excess kurtosis returned.
... Unused.

Method mgf(): The moment generating function is defined by

$$mgf_X(t) = E_X[exp(xt)]$$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:
ChiSquaredNoncentral$mgf(t, ...)

Arguments:
t (integer(1))
    t integer to evaluate function at.
... Unused.

Method cf(): The characteristic function is defined by

\[ cf_X(t) = E_X[exp(xti)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
ChiSquaredNoncentral$cf(t, ...)

Arguments:
t (integer(1))
    t integer to evaluate function at.
... Unused.

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

Usage:
ChiSquaredNoncentral$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

Author(s)
Jordan Deenichin

References

See Also
Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete
**Description**

Calculates the convolution of two distribution via numerical calculations.

**Usage**

```r
## S3 method for class 'Distribution'
+ x + y

## S3 method for class 'Distribution'
- x - y
```

**Arguments**

- `x, y` Distribution

**Details**

The convolution of two probability distributions $X, Y$ is the sum

$$ Z = X + Y $$

which has a pmf,

$$ P(Z = z) = \sum_x P(X = x)P(Y = z - x) $$

with an integration analogue for continuous distributions.

Currently distr6 supports the addition of discrete and continuous probability distributions, but only subtraction of continuous distributions.

**Value**

Returns an R6 object of class Convolution.

**Super classes**

`distr6::Distribution` -> `distr6::DistributionWrapper` -> Convolution

**Methods**

**Public methods:**

- `Convolution$new()`
- `Convolution$clone()`

**Method `new()`:** Creates a new instance of this R6 class.
Usage:
Convolution$new(dist1, dist2, add = TRUE)

Arguments:
dist1 (Distribution)
  First Distribution in convolution, i.e. dist1 ± dist2.
dist2 (Distribution)
  Second Distribution in convolution, i.e. dist1 ± dist2.
add (logical(1))
  If TRUE (default) then adds the distributions together, otherwise subtracts.

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

Usage:
Convolution$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also

Other wrappers: DistributionWrapper, HuberizedDistribution, MixtureDistribution, ProductDistribution, TruncatedDistribution, VectorDistribution

Examples

binom <- Bernoulli$new() + Bernoulli$new()
binom$pdf(2)
Binomial$new(size = 2)$pdf(2)
norm <- Normal$new(mean = 3) - Normal$new(mean = 2)
norm$pdf(1)
Normal$new(mean = 1, var = 2)$pdf(1)
Super class

distr6::DistributionDecorator -> CoreStatistics

Methods

Public methods:

- CoreStatistics$mgf()
- CoreStatistics$cf()
- CoreStatistics$pgf()
- CoreStatistics$entropy()
- CoreStatistics$skewness()
- CoreStatistics$kurtosis()
- CoreStatistics$variance()
- CoreStatistics$kthmoment()
- CoreStatistics$genExp()
- CoreStatistics$mode()
- CoreStatistics$mean()
- CoreStatistics$clone()

Method mgf(): Numerically estimates the moment-generating function.

Usage:
CoreStatistics$mgf(t, ...)

Arguments:

- t (integer(1))
  t integer to evaluate function at.
- ... ANY
  Passed to $genExp.

Method cf(): Numerically estimates the characteristic function.

Usage:
CoreStatistics$cf(t, ...)

Arguments:

- t (integer(1))
  t integer to evaluate function at.
- ... ANY
  Passed to $genExp.

Method pgf(): Numerically estimates the probability-generating function.

Usage:
CoreStatistics$pgf(z, ...)

Arguments:

- z (integer(1))
  z integer to evaluate probability generating function at.
... ANY
   Passed to $genExp.

**Method** entropy(): Numerically estimates the entropy function.

*Usage:*
CoreStatistics$entropy(base = 2, 

*Arguments:*
base (integer(1))
   Base of the entropy logarithm, default = 2 (Shannon entropy)

... ANY
   Passed to $genExp.

**Method** skewness(): Numerically estimates the distribution skewness.

*Usage:*
CoreStatistics$skewness(...)

*Arguments:*
... ANY
   Passed to $genExp.

**Method** kurtosis(): Numerically estimates the distribution kurtosis.

*Usage:*
CoreStatistics$kurtosis(excess = TRUE, 

*Arguments:*
excess (logical(1))
   If TRUE (default) excess kurtosis returned.

... ANY
   Passed to $genExp.

**Method** variance(): Numerically estimates the distribution variance.

*Usage:*
CoreStatistics$variance(...)

*Arguments:*
... ANY
   Passed to $genExp.

**Method** kthmoment(): The kth central moment of a distribution is defined by

\[
CM(k)_X = E_X[(x - \mu)^k]
\]

the kth standardised moment of a distribution is defined by

\[
SM(k)_X = \frac{CM(k)}{\sigma^k}
\]

the kth raw moment of a distribution is defined by

\[
RM(k)_X = E_X[x^k]
\]

where \(E_X\) is the expectation of distribution \(X\), \(\mu\) is the mean of the distribution and \(\sigma\) is the standard deviation of the distribution.
Usage:
CoreStatistics$kthmoment(k, type = c("central", "standard", "raw"), ...)

Arguments:
k integer(1)
   The k-th moment to evaluate the distribution at.
type character(1)
   Type of moment to evaluate.
... ANY
   Passed to $genExp.

Method genExp(): Numerically estimates $E[f(X)]$ for some function $f$.
Usage:
CoreStatistics$genExp(trafo = NULL, cubature = FALSE, ...)

Arguments:
trafo function()
   Transformation function to define the expectation, default is distribution mean.
cubature logical(1)
   If TRUE uses cubature::cubintegrate for approximation, otherwise integrate.
... ANY
   Passed to cubature::cubintegrate.

Method mode(): Numerically estimates the distribution mode.
Usage:
CoreStatistics$mode(which = "all")

Arguments:
which (character(1) | numeric(1)
   Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method mean(): Numerically estimates the distribution mean.
Usage:
CoreStatistics$mean(...) 

Arguments:
... ANY
   Passed to $genExp.

Method clone(): The objects of this class are cloneable with this method.
Usage:
CoreStatistics$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also

Other decorators: ExoticStatistics, FunctionImputation
Cosine

Examples

decorate(Exponential$new(), "CoreStatistics")
Exponential$new(decorators = "CoreStatistics")
CoreStatistics$new()$decorate(Exponential$new())

---

Cosine  Cosine Kernel

Description

Mathematical and statistical functions for the Cosine kernel defined by the pdf,

\[
f(x) = \left(\frac{\pi}{4}\right) \cos(x\pi/2)
\]

over the support \(x \in (-1, 1)\).

Super classes

distr6::Distribution -> distr6::Kernel -> Cosine

Public fields

name  Full name of distribution.
short_name  Short name of distribution for printing.
description  Brief description of the distribution.

Methods

Public methods:

- Cosine$pdfSquared2Norm()
- Cosine$cdfSquared2Norm()
- Cosine$variance()
- Cosine$clone()

Method pdfSquared2Norm(): The squared 2-norm of the pdf is defined by

\[
\int_a^b (f_X(u))^2 \, du
\]

where \(X\) is the Distribution, \(f_X\) is its pdf and \(a, b\) are the distribution support limits.

Usage:

Cosine$pdfSquared2Norm(x = 0, upper = Inf)

Arguments:

x (√ numeric(1))
    Amount to shift the result.
Method cdfSquared2Norm(): The squared 2-norm of the cdf is defined by
\[ \int_a^b (F_X(u))^2 \, du \]
where \( X \) is the Distribution, \( F_X \) is its pdf and \( a, b \) are the distribution support limits.

Usage:
```
Cosine$cdfSquared2Norm(x = 0, upper = 0)
```

Arguments:
- \( x \) (numeric(1))
  Amount to shift the result.
- \( upper \) (numeric(1))
  Upper limit of the integral.

Method variance(): The variance of a distribution is defined by the formula
\[ var_X = E[X^2] - E[X]^2 \]
where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
```
Cosine$variance(...)  
```

Arguments:
- \( ... \) Unused.

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

Usage:
```
Cosine$clone(deep = FALSE)
```

Arguments:
- \( deep \) Whether to make a deep clone.

See Also

Other kernels: Epanechnikov, LogisticKernel, NormalKernel, Quartic, Sigmoid, Silverman, TriangularKernel, Tricube, Triweight, UniformKernel
**decorate**

---

**Decorate Distributions**

**Description**

Functionality to decorate R6 Distributions (and child classes) with extra methods.

**Usage**

`decorate(distribution, decorators, ...)`

**Arguments**

- `distribution` ([Distribution])  
  Distribution to decorate.
- `decorators` (character())  
  Vector of `DistributionDecorator` names to decorate the Distribution with.
- `...` ANY  
  Extra arguments passed down to specific decorators.

**Details**

Decorating is the process of adding methods to classes that are not part of the core interface (Gamma et al. 1994). Use `listDecorators` to see which decorators are currently available. The primary use-cases are to add numeric results when analytic ones are missing, to add complex modelling functions and to impute missing d/p/q/r functions.

**Value**

Returns a `Distribution` with additional methods from the chosen `DistributionDecorator`.

**References**

Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. 1994. “Design Patterns: Elements of Reusable Object-Oriented Software.” Addison-Wesley.

**See Also**

`listDecorators()` for available decorators and `DistributionDecorator` for the parent class.

**Examples**

```r
B <- Binomial$new()
decorate(B, "CoreStatistics")

E <- Exponential$new()
decorate(E, c("CoreStatistics", "ExoticStatistics"))
```
Degenerate Distribution Class

Description
Mathematical and statistical functions for the Degenerate distribution, which is commonly used to model deterministic events or as a representation of the delta, or Heaviside, function.

Details
The Degenerate distribution parameterised with mean, \( \mu \) is defined by the pmf,

\[
f(x) = 1, \quad \text{if } x = \mu \\
f(x) = 0, \quad \text{if } x \neq \mu
\]

for \( \mu \in \mathbb{R} \).

Value
Returns an R6 object inheriting from class SDistribution.

Distribution support
The distribution is supported on \( \mu \).

Default Parameterisation
Degen(mean = 0)

Omitted Methods
N/A

Also known as
Also known as the Dirac distribution.

Super classes
distr6::Distribution -> distr6::SDistribution -> Degenerate

Public fields
name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
Active bindings

Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:

- `Degenerate$new()`
- `Degenerate$mean()`
- `Degenerate$mode()`
- `Degenerate$variance()`
- `Degenerate$skewness()`
- `Degenerate$kurtosis()`
- `Degenerate$entropy()`
- `Degenerate$mgf()`
- `Degenerate$cf()`
- `Degenerate$clone()`

Method `new()`: Creates a new instance of this R6 class.

Usage:

```r
Degenerate$new(mean = NULL, decorators = NULL)
```

Arguments:

- `mean` numeric(1)
  
  Mean of the distribution, defined on the Reals.

- `decorators` (character())
  
  Decorators to add to the distribution during construction.

Method `mean()`: The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) \times x \]

with an integration analogue for continuous distributions.

Usage:

```r
Degenerate$mean(...)```

Arguments:

- `...` Unused.

Method `mode()`: The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:

```r
Degenerate$mode(which = "all")```

Arguments:

- `which` (character(1) | numeric(1))
  
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.
Method variance(): The variance of a distribution is defined by the formula

$$\text{var}_X = E[X^2] - E[X]^2$$

where $E_X$ is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
Degenerate$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

$$sk_X = E_X\left[\frac{x - \mu}{\sigma}\right]^3$$

where $E_X$ is the expectation of distribution X, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution.

Usage:
Degenerate$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

$$k_X = E_X\left[\frac{x - \mu}{\sigma}\right]^4$$

where $E_X$ is the expectation of distribution X, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Degenerate$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

$$-\sum (f_X) \log(f_X)$$

where $f_X$ is the pdf of distribution X, with an integration analogue for continuous distributions.

Usage:
Degenerate$entropy(base = 2, ...)

Arguments:
base (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)
Method \texttt{mgf()}: The moment generating function is defined by

\[ mgf_X(t) = E_X[exp(xt)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

\textit{Usage}:
\texttt{Degenerate$mgf(t, \ldots)}

\textit{Arguments}:
\( t \) (integer(1))
\hspace{1cm} \( t \) integer to evaluate function at.

\dots Unused.

Method \texttt{cf()}: The characteristic function is defined by

\[ cf_X(t) = E_X[exp(xti)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

\textit{Usage}:
\texttt{Degenerate$cf(t, \ldots)}

\textit{Arguments}:
\( t \) (integer(1))
\hspace{1cm} \( t \) integer to evaluate function at.

\dots Unused.

Method \texttt{clone()}: The objects of this class are cloneable with this method.

\textit{Usage}:
\texttt{Degenerate$clone(deep = FALSE)}

\textit{Arguments}:
\( deep \) Whether to make a deep clone.

References


See Also

Other discrete distributions: Bernoulli, Binomial, Categorical, DiscreteUniform, EmpiricalMV, Empirical, Geometric, Hypergeometric, Logarithmic, Multinomial, NegativeBinomial, WeightedDiscrete

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete
Dirichlet Distribution Class

Description

Mathematical and statistical functions for the Dirichlet distribution, which is commonly used as a prior in Bayesian modelling and is multivariate generalisation of the Beta distribution.

Details

The Dirichlet distribution parameterised with concentration parameters, $\alpha_1, \ldots, \alpha_k$, is defined by the pdf,

$$f(x_1, \ldots, x_k) = \frac{\prod \Gamma(\alpha_i)}{\Gamma(\sum \alpha_i)} \prod x_i^{\alpha_i - 1}$$

for $\alpha = \alpha_1, \ldots, \alpha_k; \alpha > 0$, where $\Gamma$ is the gamma function.

Sampling is performed via sampling independent Gamma distributions and normalising the samples (Devroye, 1986).

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on $x_i \in (0, 1), \sum x_i = 1$.

Default Parameterisation

`Diri(params = c(1, 1))`

Omitted Methods

`cdf` and `quantile` are omitted as no closed form analytic expression could be found, decorate with FunctionImputation for a numerical imputation.

Also known as

N/A

Super classes

`distr6::Distribution -> distr6::SDistribution -> Dirichlet`

Public fields

- name Full name of distribution.
- short_name Short name of distribution for printing.
- description Brief description of the distribution.
- packages Packages required to be installed in order to construct the distribution.
Active bindings

**properties**  Returns distribution properties, including skewness type and symmetry.

Methods

**Public methods:**

- `Dirichlet$new()`
- `Dirichlet$mean()`
- `Dirichlet$mode()`
- `Dirichlet$variance()`
- `Dirichlet$entropy()`
- `Dirichlet$pgf()`
- `Dirichlet$setParameterValue()`
- `Dirichlet$clone()`

**Method** `new()`: Creates a new instance of this R6 class.

*Usage:*
`Dirichlet$new(params = NULL, decorators = NULL)`

*Arguments:*

- `params` numeric()
  - Vector of concentration parameters of the distribution defined on the positive Reals.
- `decorators` (character())
  - Decorators to add to the distribution during construction.

**Method** `mean()`: The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) * x \]

with an integration analogue for continuous distributions.

*Usage:*
`Dirichlet$mean(...)`

*Arguments:*

- `...` Unused.

**Method** `mode()`: The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

*Usage:*
`Dirichlet$mode(which = "all")`

*Arguments:*

- `which` (character(1) | numeric(1))
  - Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.
Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
Dirichlet$variance(...)

Arguments:
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

\[ -\sum (f_X \log(f_X)) \]

where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions.

Usage:
Dirichlet$entropy(base = 2, ...)

Arguments:
base (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method pgf(): The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(z^X)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:
Dirichlet$pgf(z, ...)

Arguments:
z (integer(1))
  z integer to evaluate probability generating function at.
... Unused.

Method setParameterValue(): Sets the value(s) of the given parameter(s).

Usage:
Dirichlet$setParameterValue(
    ...,
    lst = list(...),
    error = "warn",
    resolveConflicts = FALSE
)

Arguments:
... ANY
  Named arguments of parameters to set values for. See examples.
DiscreteUniform

lst (list(1))
Alternative argument for passing parameters. List names should be parameter names and
list values are the new values to set.

error (character(1))
If "warn" then returns a warning on error, otherwise breaks if "stop".

resolveConflicts (logical(1))
If FALSE (default) throws error if conflicting parameterisations are provided, otherwise au-
tomatically resolves them by removing all conflicting parameters.

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

Usage:
Dirichlet$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References
Michael P. McLaughlin.

96305-7.

See Also
Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentra-
ChiSquared, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma,
Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal,
Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT,
Triangular, Uniform, Wald, Weibull

Other multivariate distributions: EmpiricalMV, Multinomial, MultivariateNormal

Examples

d <- Dirichlet$new(params = c(2, 5, 6))
d$pdf(0.1, 0.4, 0.5)
d$pdf(c(0.3, 0.2), c(0.6, 0.9), c(0.9, 0.1))

DiscreteUniform

Discrete Uniform Distribution Class

Description
Mathematical and statistical functions for the Discrete Uniform distribution, which is commonly
used as a discrete variant of the more popular Uniform distribution, used to model events with an
equal probability of occurring (e.g. role of a die).
Details
The Discrete Uniform distribution parameterised with lower, \( a \), and upper, \( b \), limits is defined by the pmf,

\[
f(x) = \frac{1}{b - a + 1}
\]

for \( a, b \in \mathbb{Z}; \ b \geq a \).

Value
Returns an R6 object inheriting from class SDistribution.

Distribution support
The distribution is supported on \( \{a, a + 1, \ldots, b\} \).

Default Parameterisation
DUnif(lower = 0, upper = 1)

Omitted Methods
N/A

Also known as
N/A

Super classes
distr6::Distribution -> distr6::SDistribution -> DiscreteUniform

Public fields
name  Full name of distribution.
short_name  Short name of distribution for printing.
description  Brief description of the distribution.
packages  Packages required to be installed in order to construct the distribution.

Active bindings
properties  Returns distribution properties, including skewness type and symmetry.

Methods
Public methods:
- DiscreteUniform$new()
- DiscreteUniform$mean()
- DiscreteUniform$mode()
- DiscreteUniform$variance()
Method `new()`: Creates a new instance of this R6 class.

Usage:
DiscreteUniform$new(lower = NULL, upper = NULL, decorators = NULL)

Arguments:
- `lower` (integer(1))
  Lower limit of the Distribution, defined on the Naturals.
- `upper` (integer(1))
  Upper limit of the Distribution, defined on the Naturals.
- `decorators` (character())
  Decorators to add to the distribution during construction.

Method `mean()`: The arithmetic mean of a (discrete) probability distribution $X$ is the expectation

$$E_X(X) = \sum p_X(x) \cdot x$$

with an integration analogue for continuous distributions.

Usage:
DiscreteUniform$mean(...)

Arguments:
... Unused.

Method `mode()`: The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
DiscreteUniform$mode(which = "all")

Arguments:
- `which` (character(1) | numeric(1))
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method `variance()`: The variance of a distribution is defined by the formula

$$\text{var}_X = E[X^2] - E[X]^2$$

where $E_X$ is the expectation of distribution $X$. If the distribution is multivariate the covariance matrix is returned.

Usage:
Method `skewness()`: The skewness of a distribution is defined by the third standardised moment,

\[ sk_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:
DiscreteUniform$skewness(...)
Arguments:
... Unused.

Method `kurtosis()`: The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^4 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
DiscreteUniform$kurtosis(excess = TRUE, ...)
Arguments:
excess (logical(1))
If TRUE (default) excess kurtosis returned.
... Unused.

Method `entropy()`: The entropy of a (discrete) distribution is defined by

\[ - \sum (f_X) \log(f_X) \]

where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions.

Usage:
DiscreteUniform$entropy(base = 2, ...)
Arguments:
base (integer(1))
Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method `mgf()`: The moment generating function is defined by

\[ mgf_X(t) = E_X [exp(xt)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).
Usage:
DiscreteUniform$mgf(t, ...)

Arguments:
t (integer(1))
  t integer to evaluate function at.
... Unused.

Method cf(): The characteristic function is defined by
\[ cf_X(t) = E_X[exp(xti)] \]
where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
DiscreteUniform$cf(t, ...)

Arguments:
t (integer(1))
  t integer to evaluate function at.
... Unused.

Method pgf(): The probability generating function is defined by
\[ pgf_X(z) = E_X[exp(z^t)] \]
where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
DiscreteUniform$pgf(z, ...)

Arguments:
z (integer(1))
  z integer to evaluate probability generating function at.
... Unused.

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

Usage:
DiscreteUniform$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References
See Also

Other discrete distributions: Bernoulli, Binomial, Categorical, Degenerate, EmpiricalMV, Empirical, Geometric, Hypergeometric, Logarithmic, Multinomial, NegativeBinomial, WeightedDiscrete

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

distr6News

Description

Displays the contents of the NEWS.md file for viewing distr6 release information.

Usage

distr6News()

Value

NEWS.md in viewer.

Examples

```r
## Not run:
distr6News()
## End(Not run)
```

Distribution

Generalised Distribution Object

Description

A generalised distribution object for defining custom probability distributions as well as serving as the parent class to specific, familiar distributions.

Value

Returns R6 object of class Distribution.
Public fields

name  Full name of distribution.
short_name  Short name of distribution for printing.
description  Brief description of the distribution.

Active bindings

decorators  Returns decorators currently used to decorate the distribution.
traits  Returns distribution traits.
valueSupport  Deprecated, use $traits$valueSupport.
variateForm  Deprecated, use $traits$variateForm.
type  Deprecated, use $traits$type.
properties  Returns distribution properties, including skewness type and symmetry.
support  Deprecated, use $properties$type.
symmetry  Deprecated, use $properties$symmetry.
sup  Returns supremum (upper bound) of the distribution support.
inf  Returns infimum (lower bound) of the distribution support.
dmax  Returns maximum of the distribution support.
dmin  Returns minimum of the distribution support.
kurtosisType  Deprecated, use $properties$kurtosis.
skewnessType  Deprecated, use $properties$skewness.

Methods

Public methods:

- Distribution$new()
- Distribution$strprint()
- Distribution$sprint()
- Distribution$summary()
- Distribution$parameters()
- Distribution$getParameterValue()
- Distribution$setParameterValue()
- Distribution$pdf()
- Distribution$cdf()
- Distribution$quantile()
- Distribution$rand()
- Distribution$prec()
- Distribution$stdev()
- Distribution$median()
- Distribution$iqr()
- Distribution$confidence()
- `Distribution$correlation()`
- `Distribution$liesInSupport()`
- `Distribution$liesInType()`
- `Distribution$workingSupport()`
- `Distribution$clone()`

**Method** `new()`: Creates a new instance of this R6 class.

**Usage:**

```r
Distribution$new(
  name = NULL,
  short_name = NULL,
  type,
  support = NULL,
  symmetric = FALSE,
  pdf = NULL,
  cdf = NULL,
  quantile = NULL,
  rand = NULL,
  parameters = NULL,
  decorators = NULL,
  valueSupport = NULL,
  variateForm = NULL,
  description = NULL,
  .suppressChecks = FALSE
)
```

**Arguments:**

- `name` character(1)
  - Full name of distribution.
- `short_name` character(1)
  - Short name of distribution for printing.
- `type` ([set6::Set])
  - Distribution type.
- `support` ([set6::Set])
  - Distribution support.
- `symmetric` logical(1)
  - Symmetry of the distribution.
- `pdf` function(1)
  - Probability density function of the distribution. At least one of `pdf` and `cdf` must be provided.
- `cdf` function(1)
  - Cumulative distribution function of the distribution. At least one of `pdf` and `cdf` must be provided.
- `quantile` function(1)
  - Quantile (inverse-cdf) function of the distribution.
- `rand` function(1)
  - Simulation function for drawing random samples from the distribution.
parameters ([param6::ParameterSet])
   Parameter set for defining the parameters in the distribution, which should be set before
   construction.

decorators (character())
   Decorators to add to the distribution during construction.

valueSupport (character(1))
   The support type of the distribution, one of "discrete", "continuous", "mixture". If NULL,
   determined automatically.

variateForm (character(1))
   The variate type of the distribution, one of "univariate", "multivariate", "matrixvariate". If
   NULL, determined automatically.

description (character(1))
   Optional short description of the distribution.

 suppressChecks (logical(1))
   Used internally.

**Method** strprint():  Printable string representation of the Distribution. Primarily used
   internally.

Usage:
Distribution$strprint(n = 2)

Arguments:
  n (integer(1))
     Number of parameters to display when printing.

**Method** print():  Prints the Distribution.

Usage:
Distribution$print(n = 2, ...)

Arguments:
  n (integer(1))
     Passed to $strprint.
  ... ANY
     Unused. Added for consistency.

**Method** summary():  Prints a summary of the Distribution.

Usage:
Distribution$summary(full = TRUE, ...)

Arguments:
  full (logical(1))
     If TRUE (default) prints a long summary of the distribution, otherwise prints a shorter sum-
     mary.
  ... ANY
     Unused. Added for consistency.

**Method** parameters():  Returns the full parameter details for the supplied parameter.

Usage:
Distribution$parameters(id = NULL)

Arguments:
id Deprecated.

Method `getParameterValue()`: Returns the value of the supplied parameter.

Usage:
Distribution$getParameterValue(id, error = "warn")

Arguments:
id character()
    id of parameter value to return.
error (character(1))
    If "warn" then returns a warning on error, otherwise breaks if "stop".

Method `setParameterValue()`: Sets the value(s) of the given parameter(s).

Usage:
Distribution$setParameterValue(
    ...,
    lst = list(...),
    error = "warn",
    resolveConflicts = FALSE
)

Arguments:
... ANY
    Named arguments of parameters to set values for. See examples.
lst (list(1))
    Alternative argument for passing parameters. List names should be parameter names and
    list values are the new values to set.
error (character(1))
    If "warn" then returns a warning on error, otherwise breaks if "stop".
resolveConflicts (logical(1))
    If FALSE (default) throws error if conflicting parameterisations are provided, otherwise au-
    tomatically resolves them by removing all conflicting parameters.

Examples:
b = Binomial$new()
b$setParameterValue(size = 4, prob = 0.4)
b$setParameterValue(lst = list(size = 4, prob = 0.4))

Method `pdf()`: For discrete distributions the probability mass function (pmf) is returned, defined
as
\[ p_X(x) = P(X = x) \]
for continuous distributions the probability density function (pdf), \( f_X \), is returned
\[ f_X(x) = P(x < X \leq x + dx) \]
for some infinitesimally small \( dx \).
If available a pdf will be returned using an analytic expression. Otherwise, if the distribution has
not been decorated with `FunctionImputation`, `NULL` is returned.
Usage:
Distribution$pdf(..., log = FALSE, simplify = TRUE, data = NULL)

Arguments:
... (numeric())
  Points to evaluate the function at Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.
log (logical(1))
  If TRUE returns the logarithm of the probabilities. Default is FALSE.
simplify logical(1)
  If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.
data array
  Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

Examples:
b <- Binomial$new()
b$pdf(1:10)
b$pdf(1:10, log = TRUE)
b$pdf(data = matrix(1:10))

mvn <- MultivariateNormal$new()
mvn$pdf(1, 2)
mvn$pdf(1:2, 3:4)
mvn$pdf(data = matrix(1:4, nrow = 2), simplify = FALSE)

Method cdf(): The (lower tail) cumulative distribution function, $F_X$, is defined as

$$F_X(x) = P(X \leq x)$$

If lower.tail is FALSE then $1 - F_X(x)$ is returned, also known as the survival function.
If available a cdf will be returned using an analytic expression. Otherwise, if the distribution has not been decorated with FunctionImputation, NULL is returned.

Usage:
Distribution$cdf(
  ..., 
  lower.tail = TRUE, 
  log.p = FALSE, 
  simplify = TRUE, 
  data = NULL
)

Arguments:
... (numeric())
  Points to evaluate the function at Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.
lower.tail (logical(1))
  If TRUE (default), probabilities are \( X \leq x \), otherwise, \( P(X > x) \).

log.p (logical(1))
  If TRUE returns the logarithm of the probabilities. Default is FALSE.

simplify logical(1)
  If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.

data array
  Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

Examples:

```r
b <- Binomial$new()
b$cdf(1:10)
b$cdf(1:10, log.p = TRUE, lower.tail = FALSE)
b$cdf(data = matrix(1:10))
```

**Method quantile():** The quantile function, \( q_X \), is the inverse cdf, i.e.

\[
q_X(p) = F_X^{-1}(p) = \inf\{x \in R : F_X(x) \geq p\}
\]

#nolint
If `lower.tail` is FALSE then \( q_X(1 - p) \) is returned.
If available a quantile will be returned using an analytic expression. Otherwise, if the distribution has not been decorated with FunctionImputation, NULL is returned.

**Usage:**

```r
Distribution$quantile(
  ..., 
  lower.tail = TRUE, 
  log.p = FALSE, 
  simplify = TRUE, 
  data = NULL 
)
```

**Arguments:**

```r
... (numeric())
  Points to evaluate the function at Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.
```

lower.tail (logical(1))
  If TRUE (default), probabilities are \( X \leq x \), otherwise, \( P(X > x) \).

log.p (logical(1))
  If TRUE returns the logarithm of the probabilities. Default is FALSE.

simplify logical(1)
  If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.

data array
  Alternative method to specify points to evaluate. If univariate then rows correspond with
number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

Examples:
\begin{verbatim}
b <- Binomial$new()
b$quantile(0.42)
b$quantile(log(0.42), log.p = TRUE, lower.tail = TRUE)
b$quantile(data = matrix(c(0.1,0.2)))
\end{verbatim}

Method rand(): The rand function draws \( n \) simulations from the distribution. If available simulations will be returned using an analytic expression. Otherwise, if the distribution has not been decorated with FunctionImputation, NULL is returned.

Usage:
Distribution$rand(n, simplify = TRUE)

Arguments:
\begin{itemize}
  \item \textbf{n} (numeric(1))
    \begin{itemize}
      \item Number of points to simulate from the distribution. If length greater than 1, then \( n \leftarrow length(n) \), \textbf{simplify} logical(1)
      \item If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.
    \end{itemize}
\end{itemize}

Examples:
\begin{verbatim}
b <- Binomial$new()
b$rand(10)
mvn <- MultivariateNormal$new()
mvn$rand(5)
\end{verbatim}

Method prec(): Returns the precision of the distribution as \( 1 / \text{self}\$variance() \).

Usage:
Distribution$prec()

Method stdev(): Returns the standard deviation of the distribution as \( \sqrt{\text{self}\$variance()} \).

Usage:
Distribution$stdev()

Method median(): Returns the median of the distribution. If an analytical expression is available returns distribution median, otherwise if symmetric returns \text{self}\$mean, otherwise returns \text{self}\$quantile(0.5).

Usage:
Distribution$median(na.rm = NULL, ...)

Arguments:
\begin{itemize}
  \item \textbf{na.rm} (logical(1))
    \begin{itemize}
      \item Ignored, added for consistency.
    \end{itemize}
  \item \textbf{...} ANY
    \begin{itemize}
      \item Ignored, added for consistency.
    \end{itemize}
\end{itemize}
Method `iqr()`: Inter-quartile range of the distribution. Estimated as `self$quantile(0.75) - self$quantile(0.25).

Usage:
Distribution$`iqr()`

Method `confidence()`: 1 or 2-sided confidence interval around distribution.

Usage:
Distribution$`confidence(alpha = 0.95, sides = "both", median = FALSE)`

Arguments:

- `alpha` (numeric(1))
  - Level of confidence, default is 95%
- `sides` (character(1))
  - One of 'lower', 'upper' or 'both'
- `median` (logical(1))
  - If TRUE also returns median

Method `correlation()`: If univariate returns 1, otherwise returns the distribution correlation.

Usage:
Distribution$`correlation()`

Method `liesInSupport()`: Tests if the given values lie in the support of the distribution. Uses `[set6::Set]$contains.

Usage:
Distribution$`liesInSupport(x, all = TRUE, bound = FALSE)`

Arguments:

- `x` ANY
  - Values to test.
- `all` logical(1)
  - If TRUE (default) returns TRUE if all x are in the distribution, otherwise returns a vector of logicals corresponding to each element in x.
- `bound` logical(1)
  - If TRUE then tests if x lie between the upper and lower bounds of the distribution, otherwise tests if x lie between the maximum and minimum of the distribution.

Method `liesInType()`: Tests if the given values lie in the type of the distribution. Uses `[set6::Set]$contains.

Usage:
Distribution$`liesInType(x, all = TRUE, bound = FALSE)`

Arguments:

- `x` ANY
  - Values to test.
- `all` logical(1)
  - If TRUE (default) returns TRUE if all x are in the distribution, otherwise returns a vector of logicals corresponding to each element in x.
bound logical(1)

If TRUE then tests if x lie between the upper and lower bounds of the distribution, otherwise tests if x lie between the maximum and minimum of the distribution.

**Method workingSupport():** Returns an estimate for the computational support of the distribution. If an analytical cdf is available, then this is computed as the smallest interval in which the cdf lower bound is 0 and the upper bound is 1, bounds are incremented in $10^i$ intervals. If no analytical cdf is available, then this is computed as the smallest interval in which the lower and upper bounds of the pdf are 0, this is much less precise and is more prone to error. Used primarily by decorators.

**Usage:**

Distribution$workingSupport()

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

**Usage:**

Distribution$clone(deep = FALSE)

**Arguments:**

deep Whether to make a deep clone.

**Examples**

```r
## Method `Distribution$setParameterValue`

b = Binomial$new()
b$setParameterValue(size = 4, prob = 0.4)
b$setParameterValue(lst = list(size = 4, prob = 0.4))

## Method `Distribution$pdf`

b <- Binomial$new()
b$pdf(1:10)
b$pdf(1:10, log = TRUE)
b$pdf(data = matrix(1:10))

mvn <- MultivariateNormal$new()
mvn$pdf(1, 2)
mvn$pdf(1:2, 3:4)
mvn$pdf(data = matrix(1:4, nrow = 2), simplify = FALSE)

## Method `Distribution$cdf`

b <- Binomial$new()
b$cdf(1:10)
```
### DistributionDecorator  

**Abstract DistributionDecorator Class**

**Description**

Abstract class that cannot be constructed directly.

**Details**

Decorating is the process of adding methods to classes that are not part of the core interface (Gamma et al. 1994). Use listDecorators to see which decorators are currently available. The primary use-cases are to add numeric results when analytic ones are missing, to add complex modelling functions and to impute missing d/p/q/r functions.

Use decorate or $decorate to decorate distributions.

**Value**

Returns error. Abstract classes cannot be constructed directly.  
An R6 object.

**Public fields**

- packages  
  Packages required to be installed in order to construct the distribution.

**Active bindings**

- methods  
  Returns the names of the available methods in this decorator.
Methods

Public methods:

• DistributionDecorator$new()
• DistributionDecorator$decorate()
• DistributionDecorator$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
DistributionDecorator$new()

Method decorate(): Decorates the given distribution with the methods available in this decorator.

Usage:
DistributionDecorator$decorate(distribution, ...)

Arguments:
distribution Distribution
  Distribution to decorate.
... ANY
  Extra arguments passed down to specific decorators.

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

Usage:
DistributionDecorator$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

References

Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. 1994. “Design Patterns: Elements of Reusable Object-Oriented Software.” Addison-Wesley.

---

DistributionWrapper Abstract DistributionWrapper Class

Description

Abstract class that cannot be constructed directly.

Details

Wrappers in distr6 use the composite pattern (Gamma et al. 1994), so that a wrapped distribution has the same methods and fields as an unwrapped one. After wrapping, the parameters of a distribution are prefixed with the distribution name to ensure uniqueness of parameter IDs. Use listWrappers function to see constructable wrappers.
DistributionWrapper

Value

Returns error. Abstract classes cannot be constructed directly.

Super class

distr6::Distribution -> DistributionWrapper

Methods

Public methods:

- DistributionWrapper$new()
- DistributionWrapper$wrappedModels()
- DistributionWrapper$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
DistributionWrapper$new(
  distlist = NULL,
  name,
  short_name,
  description,
  support,
  type,
  valueSupport,
  variateForm,
  parameters = NULL,
  outerID = NULL
)

Arguments:

distlist (list())
  List of Distributions.
name (character(1))
  Wrapped distribution name.
short_name (character(1))
  Wrapped distribution ID.
description (character())
  Wrapped distribution description.
support ([set6::Set])
  Wrapped distribution support.
type ([set6::Set])
  Wrapped distribution type.
valueSupport (character(1))
  Wrapped distribution value support.
variateForm (character(1))
  Wrapped distribution variate form.
parameters (param6::ParameterSet)
  Optional parameters to add to the internal collection, ignored if distlist is given.
outerID (param6::ParameterSet)
  Parameters added by the wrapper.

**Method** `wrappedModels()`: Returns model(s) wrapped by this wrapper.

*Usage:*
DistributionWrapper$wrappedModels(model = NULL)

*Arguments:*
  - model (character(1))
    - id of wrapped `Distributions` to return. If `NULL` (default), a list of all wrapped `Distributions` is returned; if only one `Distribution` is matched then this is returned, otherwise a list of `Distributions`.

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

*Usage:*
DistributionWrapper$clone(deep = FALSE)

*Arguments:*
  - deep Whether to make a deep clone.

**References**
Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. 1994. “Design Patterns: Elements of Reusable Object-Oriented Software.” Addison-Wesley.

**See Also**
Other wrappers: `Convolution`, `HuberizedDistribution`, `MixtureDistribution`, `ProductDistribution`, `TruncatedDistribution`, `VectorDistribution`

---

**distrSimulate**

*Simulate from a Distribution*

**Description**
Helper function to quickly simulate from a distribution with given parameters.

**Usage**

distrSimulate(
  n = 100,
  distribution = "Normal",
  pars = list(),
  simplify = TRUE,
  seed,
  ...
)
Arguments

- **n**
  - number of points to simulate.

- **distribution**
  - distribution to simulate from, corresponds to ClassName of distr6 distribution, abbreviations allowed.

- **pars**
  - parameters to pass to distribution. If omitted then distribution defaults used.

- **simplify**
  - if TRUE (default) only the simulations are returned, otherwise the constructed distribution is also returned.

- **seed**
  - passed to set.seed

- **...**
  - additional optional arguments for set.seed

Value

If simplify then vector of n simulations, otherwise list of simulations and distribution.

---

**dstr**

*Helper Functionality for Constructing Distributions*

Description

Helper functions for constructing an *SDistribution* (with dstr) or *VectorDistribution* (with dstrs).

Usage

```r
dstr(d, ..., pars = NULL)
dstrs(d, pars = NULL, ...)
```

Arguments

- **d** (character(1))
  - Distribution. Can be the ShortName or ClassName from listDistributions().

- **...** (ANY)
  - Passed to the distribution constructor, should be parameters or decorators.

- **pars** (list())
  - List of parameters of same length as d corresponding to distribution parameters.

Examples

```r
# Construct standard Normal and distribution
dstr("Norm") # ShortName
dstr("Normal") # ClassName

# Construct Binomial(5, 0.1)
dstr("Binomial", size = 5, prob = 0.1)
```
Empirical Distribution Class

Description

Mathematical and statistical functions for the Empirical distribution, which is commonly used in sampling such as MCMC.

Details

The Empirical distribution is defined by the pmf,

\[ p(x) = \sum I(x = x_i)/k \]

for \( x_i \in \mathbb{R}, i = 1, \ldots, k \).

Sampling from this distribution is performed with the sample function with the elements given as the support set and uniform probabilities. Sampling is performed with replacement, which is consistent with other distributions but non-standard for Empirical distributions. Use simulateEmpiricalDistribution to sample without replacement.

The cdf and quantile assumes that the elements are supplied in an indexed order (otherwise the results are meaningless).

Value

Returns an R6 object inheriting from class SDistribution.
Distribution support

The distribution is supported on \(x_1, \ldots, x_k\).

Default Parameterisation

\(\text{Emp}(\text{samples} = 1)\)

Omitted Methods

N/A

Also known as

N/A

Super classes

\texttt{distr6::Distribution} -> \texttt{distr6::SDistribution} -> \texttt{Empirical}

Public fields

- name Full name of distribution.
- short\_name Short name of distribution for printing.
- description Brief description of the distribution.

Methods

Public methods:

- \texttt{Empirical\$new()}
- \texttt{Empirical\$mean()}
- \texttt{Empirical\$mode()}
- \texttt{Empirical\$variance()}
- \texttt{Empirical\$skewness()}
- \texttt{Empirical\$kurtosis()}
- \texttt{Empirical\$entropy()}
- \texttt{Empirical\$mgf()}
- \texttt{Empirical\$cf()}
- \texttt{Empirical\$pgf()}
- \texttt{Empirical\$setParameterValue()}
- \texttt{Empirical\$clone()}

Method \texttt{new()}: Creates a new instance of this \texttt{R6} class.

Usage:

\texttt{Empirical\$new(samples = \text{NULL}, \text{decorators} = \text{NULL})}

Arguments:
samples (numeric())
   Vector of observed samples, see examples.
decorators (character())
   Decorators to add to the distribution during construction.

Examples:
Empirical$new(runif(1000))

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) \times x \]

with an integration analogue for continuous distributions.

Usage:
Empirical$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Empirical$mode(which = "all")

Arguments:
which (character(1)|numeric(1))
   Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula

\[ var_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
Empirical$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[ sk_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3 \]

where \( E_X \) is the expectation of distribution X, \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:
Empirical$skewness(...)
Arguments:  
... Unused.

Method `kurtosis()`: The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left( \frac{x - \mu}{\sigma} \right)^4 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
`Empirical$kurtosis(excess = TRUE, ...)`

Arguments:
`excess` (logical(1))
  If TRUE (default) excess kurtosis returned.
... Unused.

Method `entropy()`: The entropy of a (discrete) distribution is defined by

\[ -\sum(f_X) \log(f_X) \]

where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions.

Usage:
`Empirical$entropy(base = 2, ...)`

Arguments:
`base` (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method `mgf()`: The moment generating function is defined by

\[ mgf_X(t) = E_X[exp(t)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:
`Empirical$mgf(t, ...)`

Arguments:
`t` (integer(1))
  t integer to evaluate function at.
... Unused.

Method `cf()`: The characteristic function is defined by

\[ cf_X(t) = E_X[exp(ti)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:
Empirical$cf(t, ...)  
Arguments:  
t (integer(1))  
  t integer to evaluate function at.  
... Unused.

Method pgf(): The probability generating function is defined by  

\[ pgf_X(z) = E_X[exp(z^x)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:  
Empirical$pgf(z, ...)  
Arguments:  
z (integer(1))  
  z integer to evaluate probability generating function at.  
... Unused.

Method setParameterValue(): Sets the value(s) of the given parameter(s).

Usage:  
Empirical$setParameterValue(  
  ...,  
  lst = NULL,  
  error = "warn",  
  resolveConflicts = FALSE  
)

Arguments:  
... ANY  
  Named arguments of parameters to set values for. See examples.  
lst (list(1))  
  Alternative argument for passing parameters. List names should be parameter names and list values are the new values to set.  
error (character(1))  
  If "warn" then returns a warning on error, otherwise breaks if "stop".  
resolveConflicts (logical(1))  
  If FALSE (default) throws error if conflicting parameterisations are provided, otherwise automatically resolves them by removing all conflicting parameters.

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

Usage:  
Empirical$clone(deep = FALSE)

Arguments:  
deep  
  Whether to make a deep clone.
EmpiricalMV

References


See Also

Other discrete distributions: Bernoulli, Binomial, Categorical, Degenerate, DiscreteUniform, EmpiricalMV, Geometric, Hypergeometric, Logarithmic, Multinomial, NegativeBinomial, WeightedDiscrete

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

Examples

```r
## Method `Empirical$new`

Empirical$new(runif(1000))
```

---

**EmpiricalMV**  
*EmpiricalMV Distribution Class*

**Description**

Mathematical and statistical functions for the EmpiricalMV distribution, which is commonly used in sampling such as MCMC.

**Details**

The EmpiricalMV distribution is defined by the pmf,

\[
p(x) = \sum I(x = x_i)/k
\]

for \( x_i \in R, i = 1, ..., k \).

Sampling from this distribution is performed with the `sample` function with the elements given as the support set and uniform probabilities. Sampling is performed with replacement, which is consistent with other distributions but non-standard for Empirical distributions. Use `simulateEmpiricalDistribution` to sample without replacement.

The cdf assumes that the elements are supplied in an indexed order (otherwise the results are meaningless).
Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on $x_1, ..., x_k$.

Default Parameterisation

EmpMV(data = data.frame(1, 1))

Omitted Methods

N/A

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> EmpiricalMV

Public fields

name  Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.

Methods

Public methods:

• EmpiricalMV$new()
• EmpiricalMV$mean()
• EmpiricalMV$variance()
• EmpiricalMV$setParameterValue()
• EmpiricalMV$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
EmpiricalMV$new(data = NULL, decorators = NULL)

Arguments:

data [matrix]
Matrix-like object where each column is a vector of observed samples corresponding to each variable.
decorators (character())
Decorators to add to the distribution during construction.
Examples:
EmpiricalMV$new(MultivariateNormal$new()$rand(100))

Method mean(): The arithmetic mean of a (discrete) probability distribution \( X \) is the expectation
\[
E_X(X) = \sum p_X(x) \times x
\]
with an integration analogue for continuous distributions.

Usage:
EmpiricalMV$mean(...)

Arguments:
... Unused.

Method variance(): The variance of a distribution is defined by the formula
\[
var_X = E[X^2] - E[X]^2
\]
where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
EmpiricalMV$variance(...)

Arguments:
... Unused.

Method setParameterValue(): Sets the value(s) of the given parameter(s).

Usage:
EmpiricalMV$setParameterValue(
  ..., 
  lst = NULL, 
  error = "warn", 
  resolveConflicts = FALSE
)

Arguments:
... ANY
  Named arguments of parameters to set values for. See examples.
lst (list(1))
  Alternative argument for passing parameters. List names should be parameter names and
  list values are the new values to set.
error (character(1))
  If "warn" then returns a warning on error, otherwise breaks if "stop".
resolveConflicts (logical(1))
  If FALSE (default) throws error if conflicting parameterisations are provided, otherwise au-
  tomatically resolves them by removing all conflicting parameters.

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

Usage:
EmpiricalMV$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.
References


See Also

Other discrete distributions: Bernoulli, Binomial, Categorical, Degenerate, DiscreteUniform, Empirical, Geometric, Hypergeometric, Logarithmic, Multinomial, NegativeBinomial, WeightedDiscrete

Other multivariate distributions: Dirichlet, Multinomial, MultivariateNormal

Examples

```r
## ------------------------------------------------
## Method `EmpiricalMV$new`
## ------------------------------------------------
EmpiricalMV$new(MultivariateNormal$new()$rand(100))
```

### Epanechnikov Epanechnikov Kernel

**Description**

Mathematical and statistical functions for the Epanechnikov kernel defined by the pdf,

\[ f(x) = \frac{3}{4} (1 - x^2) \]

over the support \( x \in (-1, 1) \).

**Details**

The quantile function is omitted as no closed form analytic expressions could be found, decorate with FunctionImputation for numeric results.

**Super classes**

\[
\text{distr6::Distribution} \rightarrow \text{distr6::Kernel} \rightarrow \text{Epanechnikov}
\]

**Public fields**

- name: Full name of distribution.
- short_name: Short name of distribution for printing.
- description: Brief description of the distribution.
Methods

Public methods:
• Epanechnikov$pdfSquared2Norm()
• Epanechnikov$cdfSquared2Norm()
• Epanechnikov$variance()
• Epanechnikov$clone()

Method pdfSquared2Norm(): The squared 2-norm of the pdf is defined by
\[
\int_{a}^{b} (f_X(u))^2 \, du
\]
where \( X \) is the Distribution, \( f_X \) is its pdf and \( a, b \) are the distribution support limits.

Usage:
Epanechnikov$pdfSquared2Norm(x = 0, upper = Inf)

Arguments:
x (numeric(1))
  Amount to shift the result.
upper (numeric(1))
  Upper limit of the integral.

Method cdfSquared2Norm(): The squared 2-norm of the cdf is defined by
\[
\int_{a}^{b} (F_X(u))^2 \, du
\]
where \( X \) is the Distribution, \( F_X \) is its pdf and \( a, b \) are the distribution support limits.

Usage:
Epanechnikov$cdfSquared2Norm(x = 0, upper = 0)

Arguments:
x (numeric(1))
  Amount to shift the result.
upper (numeric(1))
  Upper limit of the integral.

Method variance(): The variance of a distribution is defined by the formula
\[
var_X = E[X^2] - E[X]^2
\]
where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
Epanechnikov$variance(...)
**Method** clone(): The objects of this class are cloneable with this method.

**Usage:**
Epanechnikov$clone(deep = FALSE)

**Arguments:**
deep Whether to make a deep clone.

**See Also**
Other kernels: Cosine, LogisticKernel, NormalKernel, Quartic, Sigmoid, Silverman, TriangularKernel, Tricube, Triweight, UniformKernel
Super classes

\[ \text{distr6::Distribution} \rightarrow \text{distr6::SDistribution} \rightarrow \text{Erlang} \]

Public fields

- name: Full name of distribution.
- short_name: Short name of distribution for printing.
- description: Brief description of the distribution.
- packages: Packages required to be installed in order to construct the distribution.

Methods

Public methods:

- Erlang$new()
- Erlang$mean()
- Erlang$mode()
- Erlang$variance()
- Erlang$skewness()
- Erlang$kurtosis()
- Erlang$entropy()
- Erlang$mgf()
- Erlang$cf()
- Erlang$pgf()
- Erlang$clone()

Method new(): Creates a new instance of this R6 class.

Usage:

\[
\text{Erlang}\text{\$new(shape = NULL, rate = NULL, scale = NULL, decorators = NULL)}
\]

Arguments:

- shape (integer(1))
  - Shape parameter, defined on the positive Naturals.
- rate (numeric(1))
  - Rate parameter of the distribution, defined on the positive Reals.
- scale (numeric(1))
  - Scale parameter of the distribution, defined on the positive Reals. \(\text{scale} = 1/\text{rate}\). If provided \text{rate} is ignored.
- decorators (character())
  - Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution \(X\) is the expectation

\[
E_X(X) = \sum p_X(x) \ast x
\]

with an integration analogue for continuous distributions.
Usage:
Erlang$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Erlang$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
Erlang$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[ sk_X = E_X \left[ \frac{x - \mu}{\sigma}^3 \right] \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:
Erlang$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left[ \frac{x - \mu}{\sigma}^4 \right] \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Erlang$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
    If TRUE (default) excess kurtosis returned.
    ... Unused.

**Method entropy()**: The entropy of a (discrete) distribution is defined by

\[-\sum (f_X) \log(f_X)\]

where \(f_X\) is the pdf of distribution \(X\), with an integration analogue for continuous distributions.

*Usage:*

Erlang$entropy(base = 2, ...)

*Arguments:*

base (integer(1))
    Base of the entropy logarithm, default = 2 (Shannon entropy)
    ... Unused.

**Method mgf()**: The moment generating function is defined by

\[mgf_X(t) = E_X[exp(xt)]\]

where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

*Usage:*

Erlang$mgf(t, ...)

*Arguments:*

t (integer(1))
    t integer to evaluate function at.
    ... Unused.

**Method cf()**: The characteristic function is defined by

\[cf_X(t) = E_X[exp(xti)]\]

where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

*Usage:*

Erlang$cf(t, ...)

*Arguments:*

t (integer(1))
    t integer to evaluate function at.
    ... Unused.

**Method pgf()**: The probability generating function is defined by

\[pgf_X(z) = E_X[exp(zx)]\]

where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

*Usage:*

Erlang$pgf(z, ...)
Description

Gets the type of (excess) kurtosis

Usage

exkurtosisType(kurtosis)

Arguments

kurtosis numeric.
Details

Kurtosis is a measure of the tailedness of a distribution. Distributions can be compared to the Normal distribution by whether their kurtosis is higher, lower or the same as that of the Normal distribution.

A distribution with a negative excess kurtosis is called 'platykurtic', a distribution with a positive excess kurtosis is called 'leptokurtic' and a distribution with an excess kurtosis equal to zero is called 'mesokurtic'.

Value

Returns one of 'platykurtic', 'mesokurtic' or 'leptokurtic'.

Examples

```r
exkurtosisType(-1)
exkurtosisType(0)
exkurtosisType(1)
```

---

ExoticStatistics  

Exotic Statistical Methods Decorator

Description

This decorator adds methods for more complex statistical methods including p-norms, survival and hazard functions and anti-derivatives. If possible analytical expressions are exploited, otherwise numerical ones are used with a message.

Details

Decorator objects add functionality to the given Distribution object by copying methods in the decorator environment to the chosen Distribution environment.

All methods implemented in decorators try to exploit analytical results where possible, otherwise numerical results are used with a message.

Super class

distr6::DistributionDecorator -> ExoticStatistics

Methods

Public methods:

- ExoticStatistics$cdfAntiDeriv()
- ExoticStatistics$survivalAntiDeriv()
- ExoticStatistics$survival()
- ExoticStatistics$hazard()
- ExoticStatistics$cumHazard()
• ExoticStatistics$cdfPNorm()
• ExoticStatistics$pdfPNorm()
• ExoticStatistics$survivalPNorm()
• ExoticStatistics$clone()

**Method cdfAntiDeriv():** The cdf anti-derivative is defined by

$$ acdf(a, b) = \int_a^b F_X(x) dx $$

where X is the distribution, $F_X$ is the cdf of the distribution X and $a, b$ are the lower and upper limits of integration.

**Usage:**
ExoticStatistics$cdfAntiDeriv(lower = NULL, upper = NULL)

**Arguments:**
lower (numeric(1))
   Lower bounds of integral.
upper (numeric(1))
   Upper bounds of integral.

**Method survivalAntiDeriv():** The survival anti-derivative is defined by

$$ as(a, b) = \int_a^b S_X(x) dx $$

where X is the distribution, $S_X$ is the survival function of the distribution X and $a, b$ are the lower and upper limits of integration.

**Usage:**
ExoticStatistics$survivalAntiDeriv(lower = NULL, upper = NULL)

**Arguments:**
lower (numeric(1))
   Lower bounds of integral.
upper (numeric(1))
   Upper bounds of integral.

**Method survival():** The survival function is defined by

$$ S_X(x) = P(X \geq x) = 1 - F_X(x) = \int_x^\infty f_X(x) dx $$

where X is the distribution, $S_X$ is the survival function, $F_X$ is the cdf and $f_X$ is the pdf.

**Usage:**
ExoticStatistics$survival(..., log = FALSE, simplify = TRUE, data = NULL)

**Arguments:**
... (numeric())
   Points to evaluate the function at Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.
log (logical(1))
  If TRUE returns the logarithm of the probabilities. Default is FALSE.

simplify logical(1)
  If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.

data array
  Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

Method hazard(): The hazard function is defined by

$$h_X(x) = \frac{f_X}{S_X}$$

where X is the distribution, $S_X$ is the survival function and $f_X$ is the pdf.

Usage:
ExoticStatistics$hazard(..., log = FALSE, simplify = TRUE, data = NULL)

Arguments:
... (numeric())
  Points to evaluate the function at Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.

log (logical(1))
  If TRUE returns the logarithm of the probabilities. Default is FALSE.

simplify logical(1)
  If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.

data array
  Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

Method cumHazard(): The cumulative hazard function is defined analytically by

$$H_X(x) = -\log(S_X)$$

where X is the distribution and $S_X$ is the survival function.

Usage:
ExoticStatistics$cumHazard(..., log = FALSE, simplify = TRUE, data = NULL)

Arguments:
... (numeric())
  Points to evaluate the function at Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.

log (logical(1))
  If TRUE returns the logarithm of the probabilities. Default is FALSE.
simplify logical(1)
If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.

data array
Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

Method cdfPNorm(): The p-norm of the cdf is defined by

$$\left( \int_a^b |F_X|^p d\mu \right)^{1/p}$$

where X is the distribution, $F_X$ is the cdf and $a, b$ are the lower and upper limits of integration.
Returns NULL if distribution is not continuous.

Usage:

ExoticStatistics$cdfPNorm(p = 2, lower = NULL, upper = NULL)

Arguments:
p (integer(1)) Norm to evaluate.
lower (numeric(1))
  Lower bounds of integral.
upper (numeric(1))
  Upper bounds of integral.

Method pdfPNorm(): The p-norm of the pdf is defined by

$$\left( \int_a^b |f_X|^p d\mu \right)^{1/p}$$

where X is the distribution, $f_X$ is the pdf and $a, b$ are the lower and upper limits of integration.
Returns NULL if distribution is not continuous.

Usage:

ExoticStatistics$pdfPNorm(p = 2, lower = NULL, upper = NULL)

Arguments:
p (integer(1)) Norm to evaluate.
lower (numeric(1))
  Lower bounds of integral.
upper (numeric(1))
  Upper bounds of integral.

Method survivalPNorm(): The p-norm of the survival function is defined by

$$\left( \int_a^b |S_X|^p d\mu \right)^{1/p}$$

where X is the distribution, $S_X$ is the survival function and $a, b$ are the lower and upper limits of integration.
Returns NULL if distribution is not continuous.
Exponential

Usage:
ExoticStatistics$survivalPNorm(p = 2, lower = NULL, upper = NULL)

Arguments:
p (integer(1)) Norm to evaluate.
lower (numeric(1)) Lower bounds of integral.
upper (numeric(1)) Upper bounds of integral.

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

Usage:
ExoticStatistics$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also
Other decorators: CoreStatistics, FunctionImputation

Examples
decorate(Exponential$new(), "ExoticStatistics")
Exponential$new(decorators = "ExoticStatistics")
ExoticStatistics$new()$decorate(Exponential$new())

---

Exponential Distribution Class

Description
Mathematical and statistical functions for the Exponential distribution, which is commonly used to model inter-arrival times in a Poisson process and has the memoryless property.

Details
The Exponential distribution parameterised with rate, λ, is defined by the pdf,

\[ f(x) = \lambda exp(-x\lambda) \]

for λ > 0.

Value
Returns an R6 object inheriting from class SDistribution.
Distribution support

The distribution is supported on the Positive Reals.

Default Parameterisation

Exp(rate = 1)

Omitted Methods

N/A

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Exponential

Public fields

- name  Full name of distribution.
- short_name  Short name of distribution for printing.
- description  Brief description of the distribution.
- packages  Packages required to be installed in order to construct the distribution.

Methods

Public methods:

- Exponential$new()
- Exponential$mean()
- Exponential$mode()
- Exponential$median()
- Exponential$variance()
- Exponential$skewness()
- Exponential$kurtosis()
- Exponential$entropy()
- Exponential$mgf()
- Exponential$cf()
- Exponential$pgf()
- Exponential$clone()

Method new(): Creates a new instance of this R6 class.

Usage:

Exponential$new(rate = NULL, scale = NULL, decorators = NULL)

Arguments:
rate (numeric(1))
Rate parameter of the distribution, defined on the positive Reals.
scale numeric(1))
Scale parameter of the distribution, defined on the positive Reals. scale = 1/rate. If
provided rate is ignored.
decorators (character())
Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ EX(X) = \sum p_X(x) * x \]

with an integration analogue for continuous distributions.

Usage:
Exponential$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local
maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Exponential$mode(which = "all")

Arguments:
which (character(1) | numeric(1)
Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies
which mode to return.

Method median(): Returns the median of the distribution. If an analytical expression is avail-
able returns distribution median, otherwise if symmetric returns self$mean, otherwise returns
self$quantile(0.5).

Usage:
Exponential$median()

Method variance(): The variance of a distribution is defined by the formula

\[ var_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance
matrix is returned.

Usage:
Exponential$variance(...)

Arguments:
... Unused.
Method skewness(): The skewness of a distribution is defined by the third standardised moment,

$$sk_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution.

Usage:
Exponential$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

$$k_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^4$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Exponential$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

$$- \sum (f_X) \log(f_X)$$

where $f_X$ is the pdf of distribution $X$, with an integration analogue for continuous distributions.

Usage:
Exponential$entropy(base = 2, ...)

Arguments:
base (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method mgf(): The moment generating function is defined by

$$mgf_X(t) = E_X[exp(xt)]$$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:
Exponential$mgf(t, ...)
Method \(\text{cf}()\): The characteristic function is defined by
\[
\text{cf}_X(t) = E_X[\exp(xt)]
\]
where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

Usage:
\(\text{Exponential}\$\text{cf}(t, \ldots)\)

Arguments:
- \(t\) (integer(1))
  - \(t\) integer to evaluate function at.
- \ldots\ Unused.

Method \(\text{pgf}()\): The probability generating function is defined by
\[
\text{pgf}_X(z) = E_X[\exp(z^t)]
\]
where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

Usage:
\(\text{Exponential}\$\text{pgf}(z, \ldots)\)

Arguments:
- \(z\) (integer(1))
  - \(z\) integer to evaluate probability generating function at.
- \ldots\ Unused.

Method \(\text{clone}()\): The objects of this class are cloneable with this method.

Usage:
\(\text{Exponential}\$\text{clone}(\text{deep} = \text{FALSE})\)

Arguments:
- \(\text{deep}\) Whether to make a deep clone.

References
Michael P. McLaughlin.

See Also
Other continuous distributions: \texttt{Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull}
Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

---

**FDistribution**

**’F’ Distribution Class**

**Description**

Mathematical and statistical functions for the ’F’ distribution, which is commonly used in ANOVA testing and is the ratio of scaled Chi-Squared distributions.

**Details**

The ’F’ distribution parameterised with two degrees of freedom parameters, \( \mu, \nu \), is defined by the pdf,

\[
f(x) = \frac{\Gamma((\mu + \nu)/2)/\Gamma(\nu/2))}{\Gamma(\mu/2)}(\mu/\nu)^{\mu/2}x^{\mu/2-1}((1 + (\mu/\nu)x)^{-\mu+\nu}/2}
\]

for \( \mu, \nu > 0 \).

**Value**

Returns an R6 object inheriting from class SDistribution.

**Distribution support**

The distribution is supported on the Positive Reals.

**Default Parameterisation**

\( F(\text{df1} = 1, \text{df2} = 1) \)

**Omitted Methods**

N/A

**Also known as**

N/A

**Super classes**

\texttt{distr6::Distribution -> distr6::SDistribution -> FDistribution}
Public fields

- name  Full name of distribution.
- short_name  Short name of distribution for printing.
- description  Brief description of the distribution.
- packages  Packages required to be installed in order to construct the distribution.

Active bindings

- properties  Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:

- FDistribution$new()
- FDistribution$mean()
- FDistribution$mode()
- FDistribution$variance()
- FDistribution$skewness()
- FDistribution$kurtosis()
- FDistribution$entropy()
- FDistribution$mgf()
- FDistribution$pgf()
- FDistribution$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
FDistribution$new(df1 = NULL, df2 = NULL, decorators = NULL)

Arguments:
- df1 (numeric(1))
  First degree of freedom of the distribution defined on the positive Reals.
- df2 (numeric(1))
  Second degree of freedom of the distribution defined on the positive Reals.
- decorators (character())
  Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution $X$ is the expectation

\[ E_X(X) = \sum p_X(x) \times x \]

with an integration analogue for continuous distributions.

Usage:
FDistribution$mean(...)  

Arguments:
... Unused.
Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
FDistribution$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula

\[ var_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
FDistribution$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[ sk_X = E_X \left[ \frac{x - \mu}{\sigma}^3 \right] \]

where \( E_X \) is the expectation of distribution X, \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:
FDistribution$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left[ \frac{x - \mu}{\sigma}^4 \right] \]

where \( E_X \) is the expectation of distribution X, \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
FDistribution$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
If TRUE (default) excess kurtosis returned.
... Unused.
Method **entropy()**: The entropy of a (discrete) distribution is defined by

\[ -\sum (f_X \log(f_X)) \]

where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions.

Usage:

- `FDistribution$entropy(base = 2, ...)`

Arguments:

- **base** (integer(1))
  
  Base of the entropy logarithm, default = 2 (Shannon entropy)

... Unused.

Method **mgf()**: The moment generating function is defined by

\[ mgf_X(t) = E_X[exp(xt)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:

- `FDistribution$mgf(t, ...)`

Arguments:

- **t** (integer(1))
  
  \( t \) integer to evaluate function at.

... Unused.

Method **pgf()**: The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(z^x)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:

- `FDistribution$pgf(z, ...)`

Arguments:

- **z** (integer(1))
  
  \( z \) integer to evaluate probability generating function at.

... Unused.

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

Usage:

- `FDistribution$clone(deep = FALSE)`

Arguments:

- **deep** Whether to make a deep clone.

References

FDistributionNoncentral

Noncentral F Distribution Class

Description
Mathematical and statistical functions for the Noncentral F distribution, which is commonly used in ANOVA testing and is the ratio of scaled Chi-Squared distributions.

Details
The Noncentral F distribution parameterised with two degrees of freedom parameters, $\mu, \nu$, and location, $\lambda$, # nolint is defined by the pdf,

$$f(x) = \sum_{r=0}^{\infty} \left( \frac{\exp(-\lambda/2)(\lambda/2)^r}{(B(\nu/2, \mu/2+r)r!)} \right) \left( \frac{\mu}{\nu} \right)^{\mu/2+r} \frac{(\nu/(\nu+x\mu))^{(\mu+\nu)/2+r}}{x^{\mu/2-1+r}}$$

for $\mu, \nu > 0, \lambda \geq 0$.

Value
Returns an R6 object inheriting from class SDistribution.

Distribution support
The distribution is supported on the Positive Reals.

Default Parameterisation
FNC(df1 = 1, df2 = 1, location = 0)

Omitted Methods
N/A
**FDistributionNoncentral**

**Also known as**

N/A

**Super classes**

```
distr6::Distribution -> distr6::SDistribution -> FDistributionNoncentral
```

**Public fields**

- name  Full name of distribution.
- short_name  Short name of distribution for printing.
- description  Brief description of the distribution.
- packages  Packages required to be installed in order to construct the distribution.

**Active bindings**

- properties  Returns distribution properties, including skewness type and symmetry.

**Methods**

**Public methods:**

- `FDistributionNoncentral$new()`
- `FDistributionNoncentral$mean()`
- `FDistributionNoncentral$variance()`
- `FDistributionNoncentral$clone()`

**Method new():** Creates a new instance of this R6 class.

**Usage:**

```r
FDistributionNoncentral$new(
  df1 = NULL,
  df2 = NULL,
  location = NULL,
  decorators = NULL
)
```

**Arguments:**

- `df1` (numeric(1))
  First degree of freedom of the distribution defined on the positive Reals.
- `df2` (numeric(1))
  Second degree of freedom of the distribution defined on the positive Reals.
- `location` (numeric(1))
  Location parameter, defined on the Reals.
- `decorators` (character())
  Decorators to add to the distribution during construction.
Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) * x \]

with an integration analogue for continuous distributions.

Usage:
FDistributionNoncentral$mean(...)

Arguments:
... Unused.

Method variance(): The variance of a distribution is defined by the formula

\[ var_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
FDistributionNoncentral$variance(...)

Arguments:
... Unused.

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

Usage:
FDistributionNoncentral$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

Author(s)
Jordan Deenichin

References

See Also
Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete
Frechet Distribution Class

Description

Mathematical and statistical functions for the Frechet distribution, which is commonly used as a special case of the Generalised Extreme Value distribution.

Details

The Frechet distribution parameterised with shape, \( \alpha \), scale, \( \beta \), and minimum, \( \gamma \), is defined by the pdf,

\[
f(x) = \left(\frac{\alpha}{\beta}\right)\left((x - \gamma)/\beta\right)^{-1-\alpha}\exp\left(-(x - \gamma)/\beta\right)^{-\alpha}
\]

for \( \alpha, \beta \in \mathbb{R}^+ \) and \( \gamma \in \mathbb{R} \).

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on \( x > \gamma \).

Default Parameterisation

\( \text{Frechet}(\text{shape} = 1, \text{scale} = 1, \text{minimum} = 0) \)

Omitted Methods

N/A

Also known as

Also known as the Inverse Weibull distribution.

Super classes

\( \text{distr6::Distribution} \rightarrow \text{distr6::SDistribution} \rightarrow \text{Frechet} \)

Public fields

- name Full name of distribution.
- short_name Short name of distribution for printing.
- description Brief description of the distribution.
- packages Packages required to be installed in order to construct the distribution.
Active bindings

properties Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:

• Frechet$new()
• Frechet$mean()
• Frechet$mode()
• Frechet$median()
• Frechet$variance()
• Frechet$skewness()
• Frechet$kurtosis()
• Frechet$entropy()
• Frechet$pgf()
• Frechet$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Frechet$new(shape = NULL, scale = NULL, minimum = NULL, decorators = NULL)

Arguments:
shape (numeric(1))
Shape parameter, defined on the positive Reals.
scale (numeric(1))
Scale parameter, defined on the positive Reals.
minimum (numeric(1))
Minimum of the distribution, defined on the Reals.
decorators (character())
Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) \times x \]

with an integration analogue for continuous distributions.

Usage:
Frechet$mean(...)

Arguments:
...
Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Frechet$mode(which = "all")
Arguments:
which (character(1) | numeric(1))
    Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method median(): Returns the median of the distribution. If an analytical expression is available returns distribution median, otherwise if symmetric returns self$mean, otherwise returns self$quantile(0.5).
Usage:
    Frechet$median()

Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.
Usage:
    Frechet$variance(...)  
Arguments:
    ... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[ \text{sk}_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.
Usage:
    Frechet$skewness(...)  
Arguments:
    ... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^4 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.
Usage:
    Frechet$kurtosis(excess = TRUE, ...)  
Arguments:
    excess (logical(1))  
        If TRUE (default) excess kurtosis returned.  
Arguments:
    ... Unused.
Method `entropy()`: The entropy of a (discrete) distribution is defined by

\[- \sum (f_X) \log(f_X)\]

where \(f_X\) is the pdf of distribution \(X\), with an integration analogue for continuous distributions.

Usage:
Frechet$entropy(base = 2, ...)  
Arguments:
base (integer(1))  
  Base of the entropy logarithm, default = 2 (Shannon entropy)  
  ... Unused.

Method `pgf()`: The probability generating function is defined by

\[pgf_X(z) = E_X[exp(z^x)]\]

where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

Usage:
Frechet$pgf(z, ...)  
Arguments:
z (integer(1))  
  z integer to evaluate probability generating function at.  
  ... Unused.

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

Usage:
Frechet$clone(deep = FALSE)  
Arguments:
deep Whether to make a deep clone.

References

See Also
Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete
**Description**

This decorator imputes missing pdf/cdf/quantile/rand methods from R6 Distributions by using strategies dependent on which methods are already present in the distribution. Unlike other decorators, private methods are added to the Distribution, not public methods. Therefore the underlying public [Distribution]$pdf, [Distribution]$cdf, [Distribution]$quantile, and [Distribution]$rand functions stay the same.

**Details**

Decorator objects add functionality to the given Distribution object by copying methods in the decorator environment to the chosen Distribution environment. All methods implemented in decorators try to exploit analytical results where possible, otherwise numerical results are used with a message.

**Super class**

`distr6::DistributionDecorator` -> FunctionImputation

**Public fields**

- **packages**: Packages required to be installed in order to construct the distribution.

**Active bindings**

- **methods**: Returns the names of the available methods in this decorator.

**Methods**

**Public methods:**

- `FunctionImputation$decorate()`
- `FunctionImputation$clone()`

**Method** `decorate()`: Decorates the given distribution with the methods available in this decorator.

*Usage:*

`FunctionImputation$decorate(distribution, n = 1000)`

*Arguments:*

- `distribution` Distribution
  - Distribution to decorate.
- `n` (integer(1))
  - Grid size for imputing functions, cannot be changed after decorating. Generally larger `n` means better accuracy but slower computation, and smaller `n` means worse accuracy and faster computation.
Method clone(): The objects of this class are cloneable with this method.

Usage:
FunctionImputation$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

See Also
Other decorators: CoreStatistics, ExoticStatistics

Examples
if (requireNamespace("GoFKernel", quietly = TRUE) &&
requireNamespace("pracma", quietly = TRUE)) {
  pdf <- function(x) ifelse(x < 1 | x > 10, 0, 1 / 10)

  x <- Distribution$new("Test",
    pdf = pdf,
    support = set6::Interval$new(1, 10, class = "integer"),
    type = set6::Naturals$new()
  )
  decorate(x, "FunctionImputation", n = 1000)

  x <- Distribution$new("Test",
    pdf = pdf,
    support = set6::Interval$new(1, 10, class = "integer"),
    type = set6::Naturals$new(),
    decorators = "FunctionImputation"
  )

  x <- Distribution$new("Test",
    pdf = pdf,
    support = set6::Interval$new(1, 10, class = "integer"),
    type = set6::Naturals$new()
  )
  FunctionImputation$new()$decorate(x, n = 1000)

  x$pdf(1:10)
x$cdf(1:10)
x$quantile(0.42)
x$rand(4)
}
**Gamma**

**Description**
Mathematical and statistical functions for the Gamma distribution, which is commonly used as the prior in Bayesian modelling, the convolution of exponential distributions, and to model waiting times.

**Details**
The Gamma distribution parameterised with shape, $\alpha$, and rate, $\beta$, is defined by the pdf,

$$f(x) = \frac{(\beta^\alpha)/\Gamma(\alpha)x^{\alpha-1}exp(-x\beta)}{}$$

for $\alpha, \beta > 0$.

**Value**
Returns an R6 object inheriting from class SDistribution.

**Distribution support**
The distribution is supported on the Positive Reals.

**Default Parameterisation**
Gamma(shape = 1, rate = 1)

**Omitted Methods**
N/A

**Also known as**
N/A

**Super classes**

```r
distr6::Distribution -> distr6::SDistribution -> Gamma
```

**Public fields**

- `name` Full name of distribution.
- `short_name` Short name of distribution for printing.
- `description` Brief description of the distribution.
- `packages` Packages required to be installed in order to construct the distribution.
Methods

Public methods:

- \texttt{Gamma$new()} \\
- \texttt{Gamma$mean()} \\
- \texttt{Gamma$mode()} \\
- \texttt{Gamma$variance()} \\
- \texttt{Gamma$skewness()} \\
- \texttt{Gamma$kurtosis()} \\
- \texttt{Gamma$entropy()} \\
- \texttt{Gamma$mgf()} \\
- \texttt{Gamma$cf()} \\
- \texttt{Gamma$pgf()} \\
- \texttt{Gamma$clone()}

\textbf{Method} \texttt{new()}: Creates a new instance of this \texttt{R6} class.

\textit{Usage:}
\begin{verbatim}
Gamma$new(
    shape = NULL,
    rate = NULL,
    scale = NULL,
    mean = NULL,
    decorators = NULL
)
\end{verbatim}

\textit{Arguments:}

- \texttt{shape (numeric(1))}
  Shape parameter, defined on the positive Reals.
- \texttt{rate (numeric(1))}
  Rate parameter of the distribution, defined on the positive Reals.
- \texttt{scale numeric(1))}
  Scale parameter of the distribution, defined on the positive Reals. \(scale = 1/rate\). If provided \texttt{rate} is ignored.
- \texttt{mean (numeric(1))}
  Alternative parameterisation of the distribution, defined on the positive Reals. If given then \texttt{rate} and \texttt{scale} are ignored. Related by \(mean = shape/rate\).
- \texttt{decorators (character())}
  Decorators to add to the distribution during construction.

\textbf{Method} \texttt{mean()}: The arithmetic mean of a (discrete) probability distribution \(X\) is the expectation

\[ E_X(X) = \sum p_X(x) \ast x \]

with an integration analogue for continuous distributions.

\textit{Usage:}
\begin{verbatim}
Gamma$mean(...)\end{verbatim}
Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Gamma$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula

\[
\text{var}_X = E[X^2] - E[X]^2
\]

where \(E_X\) is the expectation of distribution \(X\). If the distribution is multivariate the covariance matrix is returned.

Usage:
Gamma$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[
\text{sk}_X = E_X \left( \frac{x - \mu}{\sigma} \right)^3
\]

where \(E_X\) is the expectation of distribution \(X\), \(\mu\) is the mean of the distribution and \(\sigma\) is the standard deviation of the distribution.

Usage:
Gamma$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[
\text{k}_X = E_X \left( \frac{x - \mu}{\sigma} \right)^4
\]

where \(E_X\) is the expectation of distribution \(X\), \(\mu\) is the mean of the distribution and \(\sigma\) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Gamma$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
Method entropy(): The entropy of a (discrete) distribution is defined by
\[ -\sum (f_X)\log(f_X) \]
where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions.

Usage:
\[ \text{Gamma}\$\text{entropy}(\text{base} = 2, \ldots) \]

Arguments:
- base (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)

Method mgf(): The moment generating function is defined by
\[ mgf_X(t) = E_X[\exp(xt)] \]
where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:
\[ \text{Gamma}\$\text{mgf}(t, \ldots) \]

Arguments:
- t (integer(1))
  t integer to evaluate function at.

Method cf(): The characteristic function is defined by
\[ cf_X(t) = E_X[\exp(xti)] \]
where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:
\[ \text{Gamma}\$\text{cf}(t, \ldots) \]

Arguments:
- t (integer(1))
  t integer to evaluate function at.

Method pgf(): The probability generating function is defined by
\[ pgf_X(z) = E_X[\exp(zt)] \]
where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:
\[ \text{Gamma}\$\text{pgf}(z, \ldots) \]

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

*Usage:*

\[ \text{Gamma}\$\text{clone}(\text{deep} = \text{FALSE}) \]

*Arguments:*

- deep Whether to make a deep clone.

**References**


**See Also**

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

---

**generalPNorm**

*Generalised P-Norm*

**Description**

Calculate the p-norm of any function between given limits.

**Usage**

\[ \text{generalPNorm}(\text{fun}, p, \text{lower}, \text{upper}, \text{range} = \text{NULL}) \]

**Arguments**

- **fun** function to calculate the p-norm of.
- **p** the pth norm to calculate
- **lower** lower bound for the integral
- **upper** upper bound for the integral
- **range** if discrete then range of the function to sum over
Details

The p-norm of a continuous function $f$ is given by,

$$ (\int_S |f|^p d\mu)^{1/p} $$

where $S$ is the function support. And for a discrete function by

$$ \sum (x_{i+1} - x_i) \ast |f(x_i)|^p $$

where $i$ is over a given range.

The p-norm is calculated numerically using the `integrate` function and therefore results are approximate only.

Value

Returns a numeric value for the p norm of a function evaluated between given limits.

Examples

```r
generalPNorm(Exponential$new()$pdf, 2, 0, 10)
```

---

**Geometric Distribution Class**

**Description**

Mathematical and statistical functions for the Geometric distribution, which is commonly used to model the number of trials (or number of failures) before the first success.

**Details**

The Geometric distribution parameterised with probability of success, $p$, is defined by the pmf,

$$ f(x) = (1 - p)^{k-1} p $$

for probability $p$.

The Geometric distribution is used to either model the number of trials (trials = TRUE) or number of failures (trials = FALSE) before the first success.

**Value**

Returns an R6 object inheriting from class `SDistribution`.

**Distribution support**

The distribution is supported on the Naturals (zero is included if modelling number of failures before success).
Default Parameterisation

Geom(prob = 0.5, trials = FALSE)

Omitted Methods

N/A

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Geometric

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.

Methods

Public methods:

• Geometric$new()
• Geometric$mean()
• Geometric$mode()
• Geometric$variance()
• Geometric$skewness()
• Geometric$kurtosis()
• Geometric$entropy()
• Geometric$mgf()
• Geometric$cf()
• Geometric$pgf()
• Geometric$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Geometric$new(prob = NULL, qprob = NULL, trials = NULL, decorators = NULL)

Arguments:
prob (numeric(1))
  Probability of success.
qprob (numeric(1))
  Probability of failure. If provided then prob is ignored. qprob = 1 -prob.
trials (logical(1))
    If TRUE then the distribution models the number of trials, \( x \), before the first success. Otherwise the distribution calculates the probability of \( y \) failures before the first success. Mathematically these are related by \( Y = X - 1 \).

Decorators (character())
    Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution \( X \) is the expectation

\[
E_X(X) = \sum p_X(x) \cdot x
\]

with an integration analogue for continuous distributions.

Usage:
    Geometric$mean(...)

Arguments:
    ... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
    Geometric$mode(which = "all")

Arguments:
    which (character(1) | numeric(1))
        Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula

\[
var_X = E[X^2] - E[X]^2
\]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
    Geometric$variance(...)

Arguments:
    ... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[
sk_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3
\]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:
    Geometric$skewness(...)

Arguments:
Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^4 \]

where \( E_X \) is the expectation of distribution X, \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Geometric$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
   If TRUE (default) excess kurtosis returned.
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

\[-\sum (f_X) \log(f_X)\]

where \( f_X \) is the pdf of distribution X, with an integration analogue for continuous distributions.

Usage:
Geometric$entropy(base = 2, ...)

Arguments:
base (integer(1))
   Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method mgf(): The moment generating function is defined by

\[ mgf_X(t) = E_X[\exp(xt)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
Geometric$mgf(t, ...)

Arguments:
t (integer(1))
   t integer to evaluate function at.
... Unused.

Method cf(): The characteristic function is defined by

\[ cf_X(t) = E_X[\exp(xti)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
Geometric$cf(t, ...)
**Geometric**

**Arguments:**

- `t` (integer(1))
  - `t` integer to evaluate function at.
  - ... Unused.

**Method pgf():** The probability generating function is defined by

\[
pgf_X(z) = E_X[exp(z^t)]
\]

where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

**Usage:**

Geometric$pgf(z, ...)

**Arguments:**

- `z` (integer(1))
  - `z` integer to evaluate probability generating function at.
  - ... Unused.

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

**Usage:**

Geometric$clone(deep = FALSE)

**Arguments:**

- `deep` Whether to make a deep clone.

**References**


**See Also**

Other discrete distributions: Bernoulli, Binomial, Categorical, Degenerate, DiscreteUniform, EmpiricalMV, Empirical, Hypergeometric, Logarithmic, Multinomial, NegativeBinomial, WeightedDiscrete

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete
Description

Mathematical and statistical functions for the Gompertz distribution, which is commonly used in survival analysis particularly to model adult mortality rates.

Details

The Gompertz distribution parameterised with shape, $\alpha$, and scale, $\beta$, is defined by the pdf,

$$f(x) = \alpha \beta \exp(x\beta) \exp(\alpha \exp(-x\beta)\alpha)$$

for $\alpha, \beta > 0$.

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on the Non-Negative Reals.

Default Parameterisation

Gomp(shape = 1, scale = 1)

Omitted Methods

N/A

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Gompertz

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.
Methods

Public methods:

• Gompertz$new()
• Gompertz$median()
• Gompertz$pgf()
• Gompertz$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Gompertz$new(shape = NULL, scale = NULL, decorators = NULL)

Arguments:
shape (numeric(1))
  Shape parameter, defined on the positive Reals.
scale (numeric(1))
  Scale parameter, defined on the positive Reals.
decorators (character())
  Decorators to add to the distribution during construction.

Method median(): Returns the median of the distribution. If an analytical expression is available returns distribution median, otherwise if symmetric returns self$mean, otherwise returns self$quantile(0.5).

Usage:
Gompertz$median()

Method pgf(): The probability generating function is defined by

$$pgf_X(z) = E_X[exp(z^x)]$$

where X is the distribution and $E_X$ is the expectation of the distribution X.

Usage:
Gompertz$pgf(z, ...)

Arguments:
z (integer(1))
  z integer to evaluate probability generating function at.
... Unused.

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

Usage:
Gompertz$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References

See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

---

**Gumbel**

**Gumbel Distribution Class**

**Description**

Mathematical and statistical functions for the Gumbel distribution, which is commonly used to model the maximum (or minimum) of a number of samples of different distributions, and is a special case of the Generalised Extreme Value distribution.

**Details**

The Gumbel distribution parameterised with location, \( \mu \), and scale, \( \beta \), is defined by the pdf,

\[
f(x) = \exp(-(z + \exp(-z))) / \beta
\]

for \( z = (x - \mu) / \beta \), \( \mu \in \mathbb{R} \) and \( \beta > 0 \).

**Value**

Returns an R6 object inheriting from class SDistribution.

**Distribution support**

The distribution is supported on the Reals.

**Default Parameterisation**

\( \text{Gumb}(\text{location} = 0, \text{scale} = 1) \)

**Omitted Methods**

N/A
Also known as
N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Gumbel

Public fields

name  Full name of distribution.
short_name  Short name of distribution for printing.
description  Brief description of the distribution.
packages  Packages required to be installed in order to construct the distribution.

Methods

Public methods:

•  Gumbel$new()
•  Gumbel$mean()
•  Gumbel$mode()
•  Gumbel$median()
•  Gumbel$variance()
•  Gumbel$skewness()
•  Gumbel$kurtosis()
•  Gumbel$entropy()
•  Gumbel$mgf()
•  Gumbel$cf()
•  Gumbel$pgf()
•  Gumbel$clone()

Method new(): Creates a new instance of this R6 class.
Usage:
Gumbel$new(location = NULL, scale = NULL, decorators = NULL)

Arguments:
location (numeric(1))
   Location parameter defined on the Reals.
scale (numeric(1))
   Scale parameter defined on the positive Reals.
decorators (character())
   Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) * x \]

with an integration analogue for continuous distributions.
Usage:
Gumbel$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Gumbel$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method median(): Returns the median of the distribution. If an analytical expression is available returns distribution median, otherwise if symmetric returns self$mean, otherwise returns self$quantile(0.5).

Usage:
Gumbel$median()

Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
Gumbel$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[ sk_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Apery’s Constant to 16 significant figures is used in the calculation.

Usage:
Gumbel$skewness(...)

Arguments:
... Unused.
**Method kurtosis()**: The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^4 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

*Usage:*

Gumbel$kurtosis(excess = TRUE, ...)

*Arguments:*

excess (logical(1))

If TRUE (default) excess kurtosis returned.

... Unused.

**Method entropy()**: The entropy of a (discrete) distribution is defined by

\[ -\sum (f_X) \log(f_X) \]

where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions.

*Usage:*

Gumbel$entropy(base = 2, ...)

*Arguments:*

base (integer(1))

Base of the entropy logarithm, default = 2 (Shannon entropy)

... Unused.

**Method mgf()**: The moment generating function is defined by

\[ mgf_X(t) = E_X [\exp(\mu t)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

*Usage:*

Gumbel$mgf(t, ...)

*Arguments:*

t (integer(1))

\( t \) integer to evaluate function at.

... Unused.

**Method cf()**: The characteristic function is defined by

\[ cf_X(t) = E_X [\exp(\mu t i)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

`pracma::gammaz()` is used in this function to allow complex inputs.

*Usage:*

Gumbel$cf(t, ...)
**Gumbel**

**Arguments:**
t (integer(1))
    t integer to evaluate function at.

... Unused.

**Method pgf():** The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(z^t)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X.

**Usage:**
Gumbel$pgf(z, ...)

**Arguments:**
z (integer(1))
    z integer to evaluate probability generating function at.

... Unused.

**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.

**References**


**See Also**

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete
**huberize**  

*Huberize a Distribution*

---

**Description**

S3 functionality to huberize an R6 distribution.

**Usage**

huberize(x, lower, upper)

**Arguments**

- **x**: distribution to huberize.
- **lower**: lower limit for huberization.
- **upper**: upper limit for huberization.

**See Also**

HuberizedDistribution

---

**HuberizedDistribution**  

*Distribution Huberization Wrapper*

---

**Description**

A wrapper for huberizing any probability distribution at given limits.

**Details**

The pdf and cdf of the distribution are required for this wrapper, if unavailable decorate with FunctionImputation first.

Huberizes a distribution at lower and upper limits, using the formula

\[
\begin{align*}
    f_H(x) &= F(x), \text{if } x \leq \text{lower} \\
    f_H(x) &= f(x), \text{if } \text{lower} < x < \text{upper} \\
    f_H(x) &= F(x), \text{if } x \geq \text{upper}
\end{align*}
\]

where \( f_H \) is the pdf of the truncated distribution \( H = \text{Huberize}(X, \text{lower}, \text{upper}) \) and \( f_X/F_X \) is the pdf/cdf of the original distribution.

**Super classes**

\[\text{distr6::Distribution} \rightarrow \text{distr6::DistributionWrapper} \rightarrow \text{HuberizedDistribution}\]
Active bindings

properties  Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:

• HuberizedDistribution$new()
• HuberizedDistribution$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
HuberizedDistribution$new(distribution, lower = NULL, upper = NULL)

Arguments:
distribution ([Distribution])  Distribution to wrap.
lower (numeric(1))  Lower limit to huberize the distribution at. If NULL then the lower bound of the Distribution is used.
upper (numeric(1))  Upper limit to huberize the distribution at. If NULL then the upper bound of the Distribution is used.

Examples:
HuberizedDistribution$new(
  Binomial$new(prob = 0.5, size = 10),
  lower = 2, upper = 4
)

# alternate constructor
huberize(Binomial$new(), lower = 2, upper = 4)

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

Usage:
HuberizedDistribution$clone(deep = FALSE)

Arguments:
depth  Whether to make a deep clone.

See Also

Other wrappers: Convolution, DistributionWrapper, MixtureDistribution, ProductDistribution, TruncatedDistribution, VectorDistribution
Examples

```r
# Method `HuberizedDistribution$new`
HuberizedDistribution$new(
  Binomial$new(prob = 0.5, size = 10),
  lower = 2, upper = 4
)

# alternate constructor
huberize(Binomial$new(), lower = 2, upper = 4)
```

### Hypergeometric Distribution Class

**Description**

Mathematical and statistical functions for the Hypergeometric distribution, which is commonly used to model the number of successes out of a population containing a known number of possible successes, for example the number of red balls from an urn or red, blue and yellow balls.

**Details**

The Hypergeometric distribution parameterised with population size, \( N \), number of possible successes, \( K \), and number of draws from the distribution, \( n \), is defined by the pmf,

\[
f(x) = \binom{K}{x} \binom{N-K}{n-x} / \binom{N}{n}
\]

for \( N = \{0, 1, 2, \ldots\} \), \( n, K = \{0, 1, 2, \ldots, N\} \) and \( \binom{a}{b} \) is the combination (or binomial coefficient) function.

**Value**

Returns an R6 object inheriting from class SDistribution.

**Distribution support**

The distribution is supported on \( \{\max(0, n + K - N), \ldots, \min(n, K)\} \).

**Default Parameterisation**

Hyper(size = 50, successes = 5, draws = 10)

**Omitted Methods**

N/A
Hypergeometric

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Hypergeometric

Public fields

name  Full name of distribution.
short_name  Short name of distribution for printing.
description  Brief description of the distribution.
packages  Packages required to be installed in order to construct the distribution.

Active bindings

properties  Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:

• Hypergeometric$new()
• Hypergeometric$mean()
• Hypergeometric$mode()
• Hypergeometric$variance()
• Hypergeometric$skewness()
• Hypergeometric$kurtosis()
• Hypergeometric$setParameterValue()
• Hypergeometric$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Hypergeometric$new(
  size = NULL,
  successes = NULL,
  failures = NULL,
  draws = NULL,
  decorators = NULL
)

Arguments:

size (integer(1))
  Population size. Defined on positive Naturals.
successes (integer(1))
  Number of population successes. Defined on positive Naturals.
failures (integer(1))
   Number of population failures. failures = size - successes. If given then successes is
   ignored. Defined on positive Naturals.

draws (integer(1))
   Number of draws from the distribution, defined on the positive Naturals.

decorators (character(1))
   Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution \( X \) is the expectation
\[
E_X(X) = \sum p_X(x) \cdot x
\]
with an integration analogue for continuous distributions.

Usage:
Hypergeometric$mean(\ldots)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local
maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Hypergeometric$mode(which = "all"

Arguments:
which (character(1) | numeric(1)
   Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies
   which mode to return.

Method variance(): The variance of a distribution is defined by the formula
\[
var_X = E[X^2] - E[X]^2
\]
where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance
matrix is returned.

Usage:
Hypergeometric$variance(\ldots)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised mo-
ment,
\[
sk_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3
\]
where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the
standard deviation of the distribution.

Usage:
Hypergeometric$skewness(\ldots)
**Arguments:**

... Unused.

**Method kurtosis():** The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left[ \frac{X - \mu}{\sigma} \right]^4 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

**Usage:**

Hypergeometric$kurtosis(excess = TRUE, ...)

**Arguments:**

excess (logical(1))

If TRUE (default) excess kurtosis returned.

... Unused.

**Method setParameterValue():** Sets the value(s) of the given parameter(s).

**Usage:**

Hypergeometric$setParameterValue(

...,

lst = list(...),

error = "warn",

resolveConflicts = FALSE
)

**Arguments:**

... ANY

Named arguments of parameters to set values for. See examples.

lst (list(1))

Alternative argument for passing parameters. List names should be parameter names and list values are the new values to set.

error (character(1))

If "warn" then returns a warning on error, otherwise breaks if "stop".

resolveConflicts (logical(1))

If FALSE (default) throws error if conflicting parameterisations are provided, otherwise automatically resolves them by removing all conflicting parameters.

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

**Usage:**

Hypergeometric$clone(deep = FALSE)

**Arguments:**

deep Whether to make a deep clone.

**References**

See Also

Other discrete distributions: Bernoulli, Binomial, Categorical, Degenerate, DiscreteUniform, EmpiricalMV, Empirical, Geometric, Logarithmic, Multinomial, NegativeBinomial, WeightedDiscrete

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

**InverseGamma**

*Inverse Gamma Distribution Class*

**Description**

Mathematical and statistical functions for the Inverse Gamma distribution, which is commonly used in Bayesian statistics as the posterior distribution from the unknown variance in a Normal distribution.

**Details**

The Inverse Gamma distribution parameterised with shape, $\alpha$, and scale, $\beta$, is defined by the pdf,

$$f(x) = \frac{(\beta^\alpha)}{\Gamma(\alpha)}x^{-\alpha-1}\exp(-\beta/x)$$

for $\alpha, \beta > 0$, where $\Gamma$ is the gamma function.

**Value**

Returns an R6 object inheriting from class SDistribution.

**Distribution support**

The distribution is supported on the Positive Reals.

**Default Parameterisation**

InvGamma(shape = 1, scale = 1)

**Omitted Methods**

N/A

**Also known as**

N/A
InverseGamma

Super classes

distr6::Distribution -> distr6::SDistribution -> InverseGamma

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.

Methods

Public methods:

• InverseGamma$new()
• InverseGamma$mean()
• InverseGamma$mode()
• InverseGamma$variance()
• InverseGamma$skewness()
• InverseGamma$kurtosis()
• InverseGamma$entropy()
• InverseGamma$mgf()
• InverseGamma$pgf()
• InverseGamma$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
InverseGamma$new(shape = NULL, scale = NULL, decorators = NULL)

Arguments:
shape (numeric(1))
  Shape parameter, defined on the positive Reals.
scale (numeric(1))
  Scale parameter, defined on the positive Reals.
decorators (character())
  Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) \times x \]

with an integration analogue for continuous distributions.

Usage:
InverseGamma$mean(...) 

Arguments:
... Unused.
Method `mode()`: The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
InverseGamma$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method `variance()`: The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \(E_X\) is the expectation of distribution \(X\). If the distribution is multivariate the covariance matrix is returned.

Usage:
InverseGamma$variance(...) 

Arguments:
... Unused.

Method `skewness()`: The skewness of a distribution is defined by the third standardised moment,

\[ \text{sk}_X = E_X\left[\frac{x - \mu}{\sigma}\right]^3 \]

where \(E_X\) is the expectation of distribution \(X\), \(\mu\) is the mean of the distribution and \(\sigma\) is the standard deviation of the distribution.

Usage:
InverseGamma$skewness(...) 

Arguments:
... Unused.

Method `kurtosis()`: The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X\left[\frac{x - \mu}{\sigma}\right]^4 \]

where \(E_X\) is the expectation of distribution \(X\), \(\mu\) is the mean of the distribution and \(\sigma\) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
InverseGamma$kurtosis(excess = TRUE, ...) 

Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
... Unused.
Method entropy(): The entropy of a (discrete) distribution is defined by

$$- \sum (f_X) \log(f_X)$$

where $f_X$ is the pdf of distribution $X$, with an integration analogue for continuous distributions.

Usage:
InverseGamma$entropy(base = 2, ...)

Arguments:
base (integer(1))
Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method mgf(): The moment generating function is defined by

$$mgf_X(t) = E_X[exp(xt)]$$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:
InverseGamma$mgf(t, ...)

Arguments:
t (integer(1))
t integer to evaluate function at.
... Unused.

Method pgf(): The probability generating function is defined by

$$pgf_X(z) = E_X[exp(zx)]$$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:
InverseGamma$pgf(z, ...)

Arguments:
z (integer(1))
z integer to evaluate probability generating function at.
... Unused.

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.

References

See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

<table>
<thead>
<tr>
<th>Kernel</th>
</tr>
</thead>
</table>

**Abstract Kernel Class**

**Description**

Abstract class that cannot be constructed directly.

**Value**

Returns error. Abstract classes cannot be constructed directly.

**Super class**

distr6::Distribution -> Kernel

**Public fields**

package  Deprecated, use $packages instead.

packages  Packages required to be installed in order to construct the distribution.

**Methods**

**Public methods:**

- Kernel$new()
- Kernel$mode()
- Kernel$mean()
- Kernel$median()
- Kernel$pdfSquared2Norm()
- Kernel$cdfSquared2Norm()
- Kernel$skewness()
- Kernel$clone()

**Method** new(): Creates a new instance of this R6 class.
**Kernel**

*Usage:*

```r
Kernel$new(decorators = NULL, support = Interval$new(-1, 1))
```

*Arguments:*

- `decorators` (character())
  - Decorators to add to the distribution during construction.
- `support` [set6::Set]
  - Support of the distribution.

**Method** `mode()`: Calculates the mode of the distribution.

*Usage:*

```r
Kernel$mode(which = "all")
```

*Arguments:*

- `which` (character(1) | numeric(1))
  - Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

**Method** `mean()`: Calculates the mean (expectation) of the distribution.

*Usage:*

```r
Kernel$mean(...) 
```

*Arguments:*

- `...` Unused.

**Method** `median()`: Calculates the median of the distribution.

*Usage:*

```r
Kernel$median()
```

**Method** `pdfSquared2Norm()`: The squared 2-norm of the pdf is defined by

\[
\int_a^b (f_X(u))^2 du
\]

where \( X \) is the Distribution, \( f_X \) is its pdf and \( a, b \) are the distribution support limits.

*Usage:*

```r
Kernel$pdfSquared2Norm(x = 0, upper = Inf)
```

*Arguments:*

- `x` (numeric(1))
  - Amount to shift the result.
- `upper` (numeric(1))
  - Upper limit of the integral.

**Method** `cdfSquared2Norm()`: The squared 2-norm of the cdf is defined by

\[
\int_a^b (F_X(u))^2 du
\]

where \( X \) is the Distribution, \( F_X \) is its pdf and \( a, b \) are the distribution support limits.
Laplace Distribution Class

Description

Mathematical and statistical functions for the Laplace distribution, which is commonly used in signal processing and finance.

Details

The Laplace distribution parameterised with mean, \( \mu \), and scale, \( \beta \), is defined by the pdf,

\[
f(x) = \exp(-|x - \mu|/\beta)/(2\beta)
\]

for \( \mu \in \mathbb{R} \) and \( \beta > 0 \).

Value

Returns an R6 object inheriting from class SDistribution.
Distribution support

The distribution is supported on the Reals.

Default Parameterisation

Lap(mean = 0, scale = 1)

Omitted Methods

N/A

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Laplace

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.

Methods

Public methods:

• Laplace$new()
• Laplace$mean()
• Laplace$mode()
• Laplace$variance()
• Laplace$skewness()
• Laplace$kurtosis()
• Laplace$entropy()
• Laplace$mgf()
• Laplace$cf()
• Laplace$pgf()
• Laplace$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Laplace$new(mean = NULL, scale = NULL, var = NULL, decorators = NULL)

Arguments:
Mean of the distribution, defined on the Reals.

Scale parameter, defined on the positive Reals.

Variance of the distribution, defined on the positive Reals. \( \text{var} = 2 \times \text{scale}^2 \). If \( \text{var} \) is provided then \( \text{scale} \) is ignored.

Decorators to add to the distribution during construction.

**Method** `mean()`: The arithmetic mean of a (discrete) probability distribution \( X \) is the expectation

\[
E_X(X) = \sum p_X(x) \times x
\]

with an integration analogue for continuous distributions.

*Usage:*

\`
Laplace$mean(...)
``

*Arguments:*

... Unused.

**Method** `mode()`: The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

*Usage:*

\`
Laplace$mode(which = "all")
``

*Arguments:*

*which* (character(1) | numeric(1))

Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

**Method** `variance()`: The variance of a distribution is defined by the formula

\[
\text{var}_X = E[X^2] - E[X]^2
\]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

*Usage:*

\`
Laplace$variance(...)
``

*Arguments:*

... Unused.

**Method** `skewness()`: The skewness of a distribution is defined by the third standardised moment,

\[
\text{sk}_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3
\]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.
Usage:
Laplace$skewness(...)

Arguments:
... Unused.

**Method** kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

$$k_X = EX\left[ \frac{x - \mu}{\sigma}^4 \right]$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Laplace$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
   If TRUE (default) excess kurtosis returned.
... Unused.

**Method** entropy(): The entropy of a (discrete) distribution is defined by

$$- \sum (f_X) \log(f_X)$$

where $f_X$ is the pdf of distribution $X$, with an integration analogue for continuous distributions.

Usage:
Laplace$entropy(base = 2, ...)

Arguments:
base (integer(1))
   Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

**Method** mgf(): The moment generating function is defined by

$$mgf_X(t) = E_X[exp(xt)]$$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:
Laplace$mgf(t, ...)

Arguments:
t (integer(1))
   t integer to evaluate function at.
... Unused.

**Method** cf(): The characteristic function is defined by

$$cf_X(t) = E_X[exp(xti)]$$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$. 
Usage:
Laplace$cf(t, ...)

Arguments:
t (integer(1))
    t integer to evaluate function at.
... Unused.

Method pgf(): The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(z^x)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
Laplace$pgf(z, ...)

Arguments:
z (integer(1))
    z integer to evaluate probability generating function at.
... Unused.

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

Usage:
Laplace$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.

References
Michael P. McLaughlin.

See Also
Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral,
ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution,
Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Logistic, Loglogistic, Lognormal, MultivariateNormal,
Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT,
Triangular, Uniform, Wald, Weibull
Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical,
Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang,
Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz,
Gumbel, Hypergeometric, InverseGamma, Logarithmic, Logistic, Loglogistic, Lognormal,
NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral,
StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete
**length.VectorDistribution**

*Get Number of Distributions in Vector Distribution*

**Description**

Gets the number of distributions in an object inheriting from `VectorDistribution`.

**Usage**

```r
## S3 method for class 'VectorDistribution'
length(x)
```

**Arguments**

- `x` \texttt{VectorDistribution}

**lines.Distribution**

*Superimpose Distribution Functions Plots for a distr6 Object*

**Description**

One of six plots can be selected to be superimposed in the plotting window, including: pdf, cdf, quantile, survival, hazard and cumulative hazard.

**Usage**

```r
## S3 method for class 'Distribution'
lines(x, fun, npoints = 3000, ...)
```

**Arguments**

- `x` \texttt{distr6 object.}
- `fun` vector of functions to plot, one or more of: "pdf","cdf","quantile", "survival", "hazard", and "cumhazard"; partial matching available.
- `npoints` number of evaluation points.
- `...` graphical parameters.

**Details**

Unlike the `plot.Distribution` function, no internal checks are performed to ensure that the added plot makes sense in the context of the current plotting window. Therefore this function assumes that the current plot is of the same value support, see examples.
Author(s)
Chengyang Gao, Runlong Yu and Shuhan Liu

See Also
plot.Distribution for plotting a distr6 object.

Examples

plot(Normal$new(mean = 2), "pdf")
lines(Normal$new(mean = 3), "pdf", col = "red", lwd = 2)

## Not run:
# The code below gives examples of how not to use this function.
# Different value supports
plot(Binomial$new(), "cdf")
lines(Normal$new(), "cdf")

# Different functions
plot(Binomial$new(), "pdf")
lines(Binomial$new(), "cdf")

# Too many functions
plot(Binomial$new(), c("pdf", "cdf"))
lines(Binomial$new(), "cdf")

## End(Not run)

---

listDecorators

Lists Implemented Distribution Decorators

Description
Lists decorators that can decorate an R6 Distribution.

Usage

listDecorators(simplify = TRUE)

Arguments

simplify logical. If TRUE (default) returns results as characters, otherwise as R6 classes.

Value
Either a list of characters (if simplify is TRUE) or a list of DistributionDecorator classes.
**listDistributions**

Lists implemented distributions in a `data.table` or a character vector, can be filtered by traits, implemented package, and tags.

**Usage**

```r
listDistributions(simplify = FALSE, filter = NULL)
```

**Arguments**

- `simplify` logical. If FALSE (default) returns distributions with traits as a `data.table`, otherwise returns distribution names as characters.
- `filter` list to filter distributions by. See examples.

**Value**

Either a list of characters (if `simplify` is TRUE) or a `data.table` of `SDistributions` and their traits.

**See Also**

- `SDistribution`

**Examples**

```r
listDistributions()

# Filter list
listDistributions(filter = list(VariateForm = "univariate"))

# Filter is case-insensitive
listDistributions(filter = list(ValueSupport = "discrete"))

# Multiple filters
listDistributions(filter = list(ValueSupport = "discrete", package = "extraDistr"))
```
listKernels  Lists Implemented Kernels

Description
Lists all implemented kernels in distr6.

Usage
listKernels(simplify = FALSE)

Arguments
simplify logical. If FALSE (default) returns kernels with support as a data.table, otherwise returns kernel names as characters.

Value
Either a list of characters (if simplify is TRUE) or a data.table of Kernels and their traits.

See Also
Kernel

Examples
listKernels()

listWrappers  Lists Implemented Distribution Wrappers

Description
Lists wrappers that can wrap an R6 Distribution.

Usage
listWrappers(simplify = TRUE)

Arguments
simplify logical. If TRUE (default) returns results as characters, otherwise as R6 classes.

Value
Either a list of characters (if simplify is TRUE) or a list of Wrapper classes.
Logarithmic

See Also

DistributionWrapper

Examples

listWrappers()
lwWrappers(TRUE)

Logarithmic Distribution Class

Description

Mathematical and statistical functions for the Logarithmic distribution, which is commonly used to model consumer purchase habits in economics and is derived from the Maclaurin series expansion of $-ln(1 - p)$.

Details

The Logarithmic distribution parameterised with a parameter, $\theta$, is defined by the pmf,

$$f(x) = -\frac{\theta^x}{x log(1 - \theta)}$$

for $0 < \theta < 1$.

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on 1, 2, 3, . . . .

Default Parameterisation

Log(theta = 0.5)

Omitted Methods

N/A

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Logarithmic
Public fields

- **name**  Full name of distribution.
- **short_name**  Short name of distribution for printing.
- **description**  Brief description of the distribution.
- **packages**  Packages required to be installed in order to construct the distribution.

Methods

**Public methods:**

- `Logarithmic$new()`
- `Logarithmic$mean()`
- `Logarithmic$mode()`
- `Logarithmic$variance()`
- `Logarithmic$skewness()`
- `Logarithmic$kurtosis()`
- `Logarithmic$mgf()`
- `Logarithmic$cf()`
- `Logarithmic$pgf()`
- `Logarithmic$clone()`

**Method new():** Creates a new instance of this R6 class.

*Usage:*

`Logarithmic$new(theta = NULL, decorators = NULL)`

*Arguments:*

- **theta**  (numeric(1))
  
  Theta parameter defined as a probability between 0 and 1.

- **decorators**  (character())
  
  Decorators to add to the distribution during construction.

**Method mean():** The arithmetic mean of a (discrete) probability distribution \( X \) is the expectation

\[
E_X(X) = \sum p_X(x) \times x
\]

with an integration analogue for continuous distributions.

*Usage:*

`Logarithmic$mean(...)`

*Arguments:*

- **...**  Unused.

**Method mode():** The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

*Usage:*

`Logarithmic$mode(which = "all")`
Method variance(): The variance of a distribution is defined by the formula

$$\text{var}_X = E[X^2] - E[X]^2$$

where $E_X$ is the expectation of distribution $X$. If the distribution is multivariate the covariance matrix is returned.

Usage:
Logarithmic$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

$$sk_X = E_X\left[\frac{x - \mu^3}{\sigma}\right]$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution.

Usage:
Logarithmic$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

$$k_X = E_X\left[\frac{x - \mu^4}{\sigma}\right]$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Logarithmic$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
... Unused.

Method mgf(): The moment generating function is defined by

$$mgf_X(t) = E_X[exp(xt)]$$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.  
Usage:
Logarithmic$mgf(t, ...) 

*Arguments:*

- **t** (integer(1))
  - t integer to evaluate function at.
- ... Unused.

**Method cf():** The characteristic function is defined by

\[
  cf_X(t) = E_X[exp(xt)]
\]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

*Usage:*

Logarithmic$cf(t, ...)

*Arguments:*

- **t** (integer(1))
  - t integer to evaluate function at.
- ... Unused.

**Method pgf():** The probability generating function is defined by

\[
  pgf_X(z) = E_X[exp(xz)]
\]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

*Usage:*

Logarithmic$pgf(z, ...)

*Arguments:*

- **z** (integer(1))
  - z integer to evaluate probability generating function at.
- ... Unused.

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

*Usage:*

Logarithmic$clone(deep = FALSE)

*Arguments:*

- **deep** Whether to make a deep clone.

**References**


Michael P. McLaughlin.
Logistic

See Also

Other discrete distributions: Bernoulli, Binomial, Categorical, Degenerate, DiscreteUniform, EmpiricalMV, Empirical, Geometric, Hypergeometric, Multinomial, NegativeBinomial, WeightedDiscrete

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

Logistic Distribution Class

Description

Mathematical and statistical functions for the Logistic distribution, which is commonly used in logistic regression and feedforward neural networks.

Details

The Logistic distribution parameterised with mean, \( \mu \), and scale, \( s \), is defined by the pdf,

\[
f(x) = \frac{exp(- (x - \mu)/s)}{(s(1 + exp(- (x - \mu)/s))^2)}
\]

for \( \mu \epsilon R \) and \( s > 0 \).

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on the Reals.

Default Parameterisation

Logis(mean = 0, scale = 1)

Omitted Methods

N/A

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Logistic
Public fields

- name: Full name of distribution.
- short_name: Short name of distribution for printing.
- description: Brief description of the distribution.
- packages: Packages required to be installed in order to construct the distribution.

Methods

Public methods:

- Logistic$new()
- Logistic$mean()
- Logistic$mode()
- Logistic$variance()
- Logistic$skewness()
- Logistic$kurtosis()
- Logistic$entropy()
- Logistic$mgf()
- Logistic$cf()
- Logistic$pgf()
- Logistic$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Logistic$new(mean = NULL, scale = NULL, sd = NULL, decorators = NULL)

Arguments:
- mean (numeric(1))
  Mean of the distribution, defined on the Reals.
- scale (numeric(1))
  Scale parameter, defined on the positive Reals.
- sd (numeric(1))
  Standard deviation of the distribution as an alternate scale parameter, sd = scale*pi/sqrt(3).
  If given then scale is ignored.
- decorators (character())
  Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

$$E_X(X) = \sum p_X(x) * x$$

with an integration analogue for continuous distributions.

Usage:
Logistic$mean(...)
Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Logistic$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
Logistic$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[ sk_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3 \]

where \( E_X \) is the expectation of distribution X, \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:
Logistic$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^4 \]

where \( E_X \) is the expectation of distribution X, \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Logistic$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
... Unused.
Method entropy(): The entropy of a (discrete) distribution is defined by

\[ -\sum (f_X) \log(f_X) \]

where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions.

Usage:
Logistic$entropy(base = 2, ...)

Arguments:
base (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method mgf(): The moment generating function is defined by

\[ mgf_X(t) = E_X[\exp(xt)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:
Logistic$mgf(t, ...)

Arguments:
t (integer(1))
  t integer to evaluate function at.
... Unused.

Method cf(): The characteristic function is defined by

\[ cf_X(t) = E_X[\exp(xti)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:
Logistic$cf(t, ...)

Arguments:
t (integer(1))
  t integer to evaluate function at.
... Unused.

Method pgf(): The probability generating function is defined by

\[ pgf_X(z) = E_X[\exp(zt)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:
Logistic$pgf(z, ...)

Arguments:
z (integer(1))
  z integer to evaluate probability generating function at.
Method clone(): The objects of this class are cloneable with this method.

Usage:
Logistic$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References

See Also
Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull
Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FdistributionNoncentral, Fdistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

LogisticKernel  Logistic Kernel

Description
Mathematical and statistical functions for the LogisticKernel kernel defined by the pdf,

\[ f(x) = (\exp(x) + 2 + \exp(-x))^{-1} \]

over the support \( x \in \mathbb{R} \).

Super classes
distr6::Distribution -> distr6::Kernel -> LogisticKernel

Public fields
name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
LogisticKernel

Methods

Public methods:

• LogisticKernel$new()
• LogisticKernel$pdfSquared2Norm()
• LogisticKernel$cdfSquared2Norm()
• LogisticKernel$variance()
• LogisticKernel$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
LogisticKernel$new(decorators = NULL)

Arguments:

decorators (character())
Decorators to add to the distribution during construction.

Method pdfSquared2Norm(): The squared 2-norm of the pdf is defined by

\[ \int_a^b (f_X(u))^2 \, du \]

where X is the Distribution, \( f_X \) is its pdf and \( a, b \) are the distribution support limits.

Usage:
LogisticKernel$pdfSquared2Norm(x = 0, upper = Inf)

Arguments:

x (numeric(1))
Amount to shift the result.
upper (numeric(1))
Upper limit of the integral.

Method cdfSquared2Norm(): The squared 2-norm of the cdf is defined by

\[ \int_a^b (F_X(u))^2 \, du \]

where X is the Distribution, \( F_X \) is its pdf and \( a, b \) are the distribution support limits.

Usage:
LogisticKernel$cdfSquared2Norm(x = 0, upper = 0)

Arguments:

x (numeric(1))
Amount to shift the result.
upper (numeric(1))
Upper limit of the integral.
Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
LogisticKernel$variance(...)  
Arguments:  
... Unused.

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

Usage:
LogisticKernel$clone(deep = FALSE)  
Arguments:  
deep Whether to make a deep clone.

See Also

Other kernels: Cosine, Epanechnikov, NormalKernel, Quartic, Sigmoid, Silverman, TriangularKernel, Tricube, Triweight, UniformKernel

Description

Mathematical and statistical functions for the Log-Logistic distribution, which is commonly used in survival analysis for its non-monotonic hazard as well as in economics.

Details

The Log-Logistic distribution parameterised with shape, \( \beta \), and scale, \( \alpha \) is defined by the pdf,

\[ f(x) = (\beta/\alpha)(x/\alpha)^{\beta-1}(1 + (x/\alpha)^\beta)^{-2} \]

for \( \alpha, \beta > 0 \).

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on the non-negative Reals.
Default Parameterisation

LLogis(scale = 1, shape = 1)

Omitted Methods

N/A

Also known as

Also known as the Fisk distribution.

Super classes

distr6::Distribution -> distr6::SDistribution -> Loglogistic

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.

Methods

Public methods:

• Loglogistic$new()
• Loglogistic$mean()
• Loglogistic$mode()
• Loglogistic$median()
• Loglogistic$variance()
• Loglogistic$skewness()
• Loglogistic$kurtosis()
• Loglogistic$pgf()
• Loglogistic$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Loglogistic$new(scale = NULL, shape = NULL, rate = NULL, decorators = NULL)

Arguments:

scale (numeric(1))
Scale parameter, defined on the positive Reals.
shape (numeric(1))
Shape parameter, defined on the positive Reals.
rate (numeric(1))
Alternate scale parameter, rate = 1/scale. If given then scale is ignored.
Method mean(): The arithmetic mean of a (discrete) probability distribution \( X \) is the expectation

\[
E_X(X) = \sum p_X(x) * x
\]

with an integration analogue for continuous distributions.

Usage:
Loglogistic$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Loglogistic$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
    Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method median(): Returns the median of the distribution. If an analytical expression is available returns distribution median, otherwise if symmetric returns self$mean, otherwise returns self$quantile(0.5).

Usage:
Loglogistic$median()

Method variance(): The variance of a distribution is defined by the formula

\[
var_X = E[X^2] - E[X]^2
\]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
Loglogistic$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[
sk_X = E_X \left( \frac{x - \mu}{\sigma} \right)^3
\]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:
Loglogistic$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

$$k_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^4$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Loglogistic$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
... Unused.

Method pgf(): The probability generating function is defined by

$$pgf_X(z) = E_X[exp(z^x)]$$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:
Loglogistic$pgf(z, ...)

Arguments:
z (integer(1))
  $z$ integer to evaluate probability generating function at.
... Unused.

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

Usage:
Loglogistic$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References
Lognormal

See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

---

Lognormal  

Log-Normal Distribution Class

Description

Mathematical and statistical functions for the Log-Normal distribution, which is commonly used to model many natural phenomena as a result of growth driven by small percentage changes.

Details

The Log-Normal distribution parameterised with logmean, \( \mu \), and logvar, \( \sigma \), is defined by the pdf,

\[
\exp\left(\frac{-\left(\log(x) - \mu\right)^2}{2\sigma^2}\right)/\left(x\sigma \sqrt{2\pi}\right)
\]

for \( \mu \in \mathbb{R} \) and \( \sigma > 0 \).

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on the Positive Reals.

Default Parameterisation

\text{Lnorm}(\text{meanlog} = 0, \text{varlog} = 1)

Omitted Methods

N/A

Also known as

Also known as the Log-Gaussian distribution.
Super classes

distr6::Distribution -> distr6::SDistribution -> Lognormal

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.

Methods

Public methods:

• Lognormal$new()
• Lognormal$mean()
• Lognormal$mode()
• Lognormal$median()
• Lognormal$variance()
• Lognormal$skewness()
• Lognormal$kurtosis()
• Lognormal$entropy()
• Lognormal$mgf()
• Lognormal$pgf()
• Lognormal$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Lognormal$new(
  meanlog = NULL,
  varlog = NULL,
  sdlog = NULL,
  preclog = NULL,
  mean = NULL,
  var = NULL,
  sd = NULL,
  prec = NULL,
  decorators = NULL
)

Arguments:

meanlog (numeric(1))
  Mean of the distribution on the log scale, defined on the Reals.
varlog (numeric(1))
  Variance of the distribution on the log scale, defined on the positive Reals.
sdlog (numeric(1))
Standard deviation of the distribution on the log scale, defined on the positive Reals.

\[ sdlog = \text{varlog}^2 \]

If preclog missing and sdlog given then all other parameters except meanlog are ignored.
preclog (numeric(1))
Precision of the distribution on the log scale, defined on the positive Reals.

\[ preclog = 1/\text{varlog} \]

If given then all other parameters except meanlog are ignored.
mean (numeric(1))
Mean of the distribution on the natural scale, defined on the positive Reals.
var (numeric(1))
Variance of the distribution on the natural scale, defined on the positive Reals.

\[ \text{var} = (\exp(\text{var}) - 1) * \exp(2 * \text{meanlog} + \text{varlog}) \]

sd (numeric(1))
Standard deviation of the distribution on the natural scale, defined on the positive Reals.

\[ sd = \text{var}^2 \]

If prec missing and sd given then all other parameters except mean are ignored.
prec (numeric(1))
Precision of the distribution on the natural scale, defined on the Reals.

\[ prec = 1/\text{var} \]

If given then all other parameters except mean are ignored.
decorators (character())
Decorators to add to the distribution during construction.

Examples:
Lognormal$new(var = 2, mean = 1)
Lognormal$new(meanlog = 2, preclog = 5)

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) * x \]

with an integration analogue for continuous distributions.

Usage:
Lognormal$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).
Usage:
Lognormal$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies
  which mode to return.

... Unused.

Method median(): Returns the median of the distribution. If an analytical expression is available
returns distribution median, otherwise if symmetric returns self$mean, otherwise returns
self$quantile(0.5).
Usage:
Lognormal$median()

Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance
matrix is returned.
Usage:
Lognormal$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[ \text{sk}_X = E_X \left( \frac{x - \mu}{\sigma} \right)^3 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the
standard deviation of the distribution.
Usage:
Lognormal$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left( \frac{x - \mu}{\sigma} \right)^4 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the
standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.
Usage:
Lognormal$kurtosis(excess = TRUE, ...)

Arguments:
Method `entropy()`: The entropy of a (discrete) distribution is defined by

$$ - \sum (f_X) \log(f_X) $$

where $f_X$ is the pdf of distribution $X$, with an integration analogue for continuous distributions.

Usage:

```
Lognormal$entropy(base = 2, ...)
```

Arguments:

- `base` (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)
  ... Unused.

Method `mgf()`: The moment generating function is defined by

$$ mgf_X(t) = E_X[exp(xt)] $$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:

```
Lognormal$mgf(t, ...)
```

Arguments:

- `t` (integer(1))
  t integer to evaluate function at.
  ... Unused.

Method `pgf()`: The probability generating function is defined by

$$ pgf_X(z) = E_X[exp(z^x)] $$

where $X$ is the distribution and $E_X$ is the expectation of the distribution $X$.

Usage:

```
Lognormal$pgf(z, ...)
```

Arguments:

- `z` (integer(1))
  z integer to evaluate probability generating function at.
  ... Unused.

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

Usage:

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

Arguments:

- `deep` Whether to make a deep clone.
makeUniqueDistributions

References

See Also
Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull
Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

Examples

```r
## ------------------------------------------------
## Method `Lognormal$new`
## ------------------------------------------------

Lognormal$new(var = 2, mean = 1)
Lognormal$new(meanlog = 2, preclog = 5)
```

makeUniqueDistributions

De-Duplicate Distribution Names

Description
Helper function to lapply over the given distribution list, and make the short_names unique.

Usage
makeUniqueDistributions(distlist)

Arguments
distlist list of Distributions.

Details
The short_names are made unique by suffixing each with a consecutive number so that the names are no longer duplicated.
**Value**

The list of inputted distributions except with the short_names manipulated as necessary to make them unique.

**Examples**

```r
makeUniqueDistributions(list(Binomial$new(), Binomial$new()))
```

---

**Description**

Wrapper used to construct a mixture of two or more distributions.

**Details**

A mixture distribution is defined by

\[
F_P(x) = w_1 F_{X_1}(x) \ast \ldots \ast w_n F_{X_N}(x)
\]

#nolint where \(F_P\) is the cdf of the mixture distribution, \(X_1, \ldots, X_N\) are independent distributions, and \(w_1, \ldots, w_N\) are weights for the mixture.

**Super classes**

distr6::Distribution -> distr6::DistributionWrapper -> distr6::VectorDistribution -> MixtureDistribution

**Methods**

**Public methods:**

- MixtureDistribution$new()
- MixtureDistribution$strprint()
- MixtureDistribution$pdf()
- MixtureDistribution$cdf()
- MixtureDistribution$quantile()
- MixtureDistribution$rand()
- MixtureDistribution$clone()

**Method new():** Creates a new instance of this R6 class.

*Usage:*
MixtureDistribution$new(
  distlist = NULL,
  weights = "uniform",
  distribution = NULL,
  params = NULL,
  shared_params = NULL,
  name = NULL,
  short_name = NULL,
  decorators = NULL,
  vecdist = NULL,
  ids = NULL
)

Arguments:

distlist (list())
  List of Distributions.

weights (character(1)|numeric())
  Weights to use in the resulting mixture. If all distributions are weighted equally then
  "uniform" provides a much faster implementation, otherwise a vector of length equal to
  the number of wrapped distributions, this is automatically scaled internally.

distribution (character(1))
  Should be supplied with params and optionally shared_params as an alternative to distlist.
  Much faster implementation when only one class of distribution is being wrapped. distribution
  is the full name of one of the distributions in listDistributions(), or "Distribution"
  if constructing custom distributions. See examples in VectorDistribution.

params (list()|data.frame())
  Parameters in the individual distributions for use with distribution. Can be supplied as
  a list, where each element is the list of parameters to set in the distribution, or as an object
  coercable to data.frame, where each column is a parameter and each row is a distribution.
  See examples in VectorDistribution.

shared_params (list())
  If any parameters are shared when using the distribution constructor, this provides a
  much faster implementation to list and query them together. See examples in VectorDistribution.

name (character(1))
  Optional name of wrapped distribution.

short_name (character(1))
  Optional short name/ID of wrapped distribution.

decorators (character())
  Decorators to add to the distribution during construction.

vecdist VectorDistribution
  Alternative constructor to directly create this object from an object inheriting from VectorDistribu-
  tion.

ids (character())
  Optional ids for wrapped distributions in vector, should be unique and of same length as the
  number of distributions.

Examples:
MixtureDistribution$new(list(Binomial$new(prob = 0.5, size = 10), Binomial$new()), weights = c(0.2, 0.8))

**Method** strprint(): Printable string representation of the MixtureDistribution. Primarily used internally.

**Usage:**
MixtureDistribution$strprint(n = 10)

**Arguments:**
n (integer(1))
- Number of distributions to include when printing.

**Method** pdf(): Probability density function of the mixture distribution. Computed by

\[ f_M(x) = \sum_i (f_i)(x) \times w_i \]

where \(w_i\) is the vector of weights and \(f_i\) are the pdfs of the wrapped distributions.

Note that as this class inherits from VectorDistribution, it is possible to evaluate the distributions at different points, but that this is not the usual use-case for mixture distributions.

**Usage:**
MixtureDistribution$pdf(..., log = FALSE, simplify = TRUE, data = NULL)

**Arguments:**
... (numeric())
- Points to evaluate the function at, Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.

log (logical(1))
- If TRUE returns the logarithm of the probabilities. Default is FALSE.

simplify logical(1)
- If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.

data array
- Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

**Examples:**
m <- MixtureDistribution$new(list(Binomial$new(prob = 0.5, size = 10), Binomial$new()), weights = c(0.2, 0.8))
m$pdf(1:5)
m$pdf(1)
# also possible but unlikely to be used
m$pdf(1, 2)
**Method cdf()**: Cumulative distribution function of the mixture distribution. Computed by

\[ F_M(x) = \sum_{i} (F_i(x)) * w_i \]

where \( w_i \) is the vector of weights and \( F_i \) are the cdfs of the wrapped distributions.

**Usage**:

\[ \text{MixtureDistribution$cdf( ..., \lower.tail = \text{TRUE}, \log.p = \text{FALSE}, \simplify = \text{TRUE}, \data = \text{NULL} )} \]

**Arguments**:

... (numeric())

Points to evaluate the function at Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.

```r
m <- MixtureDistribution$new(list(Binomial$new(prob = 0.5, size = 10), Binomial$new()), weights = c(0.2, 0.8) )
m$cdf(1:5)
```

lower.tail (logical(1))

If TRUE (default), probabilities are \( X \leq x \), otherwise, \( P(X > x) \).

log.p (logical(1))

If TRUE returns the logarithm of the probabilities. Default is FALSE.

simplify (logical(1))

If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a `data.table::data.table`.

data array

Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

**Method quantile()**: The quantile function is not implemented for mixture distributions.

**Usage**:

\[ \text{MixtureDistribution$quantile( ..., \lower.tail = \text{TRUE}, \log.p = \text{FALSE}, \simplify = \text{TRUE}, \data = \text{NULL} )} \]

**Arguments**:

... (numeric())

Points to evaluate the function at Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.
lower.tail (logical(1))
   If TRUE (default), probabilities are \( X \leq x \), otherwise, \( P(X > x) \).
log.p (logical(1))
   If TRUE returns the logarithm of the probabilities. Default is FALSE.
simplify logical(1)
   If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.
data array
   Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

**Method rand()**: Simulation function for mixture distributions. Samples are drawn from a mixture by first sampling Multinomial(probs = weights, size = n), then sampling each distribution according to the samples from the Multinomial, and finally randomly permuting these draws.

*Usage:*
MixtureDistribution$rand(n, simplify = TRUE)

*Arguments:*
n (numeric(1))
   Number of points to simulate from the distribution. If length greater than 1, then \( n \leftarrow \text{length}(n) \), simplify logical(1)
   If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.

*Examples:*
m <- MixtureDistribution$new(distribution = "Normal",
params = data.frame(mean = 1:2, sd = 1))
m$rand(5)

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

*Usage:*
MixtureDistribution$clone(deep = FALSE)

*Arguments:*
deep Whether to make a deep clone.

**See Also**
Other wrappers: Convolution, DistributionWrapper, HuberizedDistribution, ProductDistribution, TruncatedDistribution, VectorDistribution

**Examples**

```r
## ------------------------------------------------
## Method /grave.Var
## ------------------------------------------------
MixtureDistribution$new
```

```r
MixtureDistribution$new(list(Binomial$new(prob = 0.5, size = 10), Binomial$new()),
weights = c(0.2, 0.8)
```
mixturiseVector

Create Mixture Distribution From Multiple Vectors

Description

Given \( m \) vector distributions of length \( N \), creates a single vector distribution consisting of \( n \) mixture distributions mixing the \( m \) vectors.

Usage

mixturiseVector(vecdists, weights = "uniform")

Arguments

vecdists (list())
List of VectorDistributions, should be of same length and with the non-'distlist' constructor with the same distribution.

weights (character(1)|numeric())
Weights passed to MixtureDistribution. Default uniform weighting.

Details

Let \( v_1 = (D_{11}, D_{12}, ..., D_{1N}) \) and \( v_2 = (D_{21}, D_{22}, ..., D_{2N}) \) then the mixturiseVector function creates the vector distribution \( v_3 = (D_{31}, D_{32}, ..., D_{3N}) \) where \( D_{3N} = m(D_{1N}, D_{2N}, w_N) \) where \( m \) is a mixture distribution with weights \( w_N \).
Examples

```r
# Not run:
v1 <- VectorDistribution$new(distribution = "Binomial", params = data.frame(size = 1:2))
v2 <- VectorDistribution$new(distribution = "Binomial", params = data.frame(size = 3:4))
mv1 <- mixturiseVector(list(v1, v2))

# equivalently
mv2 <- VectorDistribution$new(list(
    MixtureDistribution$new(distribution = "Binomial", params = data.frame(size = c(1, 3))),
    MixtureDistribution$new(distribution = "Binomial", params = data.frame(size = c(2, 4)))
))
mv1$pdf(1:5)
mv2$pdf(1:5)

# End(Not run)
```

**Multinomial**

**Multinomial Distribution Class**

**Description**

Mathematical and statistical functions for the Multinomial distribution, which is commonly used to extend the binomial distribution to multiple variables, for example to model the rolls of multiple dice multiple times.

**Details**

The Multinomial distribution parameterised with number of trials, \(n\), and probabilities of success, \(p_1, \ldots, p_k\), is defined by the pmf,

\[
f(x_1, x_2, \ldots, x_k) = \frac{n!}{x_1! \cdot x_2! \cdot \ldots \cdot x_k!} \cdot p_1^{x_1} \cdot p_2^{x_2} \cdot \ldots \cdot p_k^{x_k}
\]

for \(p_i, i = 1, \ldots, k; \sum p_i = 1\) and \(n = 1, 2, \ldots\).

**Value**

Returns an R6 object inheriting from class `SDistribution`.

**Distribution support**

The distribution is supported on \(\sum x_i = N\).

**Default Parameterisation**

Multinom(size = 10, probs = c(0.5, 0.5))
Omitted Methods

cdf and quantile are omitted as no closed form analytic expression could be found, decorate with `FunctionImputation` for a numerical imputation.

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Multinomial

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.

Active bindings

properties Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:

• Multinomial$new()
• Multinomial$mean()
• Multinomial$variance()
• Multinomial$skewness()
• Multinomial$kurtosis()
• Multinomial$entropy()
• Multinomial$mgf()
• Multinomial$cf()
• Multinomial$pgf()
• Multinomial$setParameterValue()
• Multinomial$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Multinomial$new(size = NULL, probs = NULL, decorators = NULL)

Arguments:
size (integer(1))
  Number of trials, defined on the positive Naturals.
probs (numeric())
   Vector of probabilities. Automatically normalised by \( \text{probs} = \text{probs}/\text{sum(probs)} \).
decorators (character())
   Decorators to add to the distribution during construction.

**Method** `mean()`: The arithmetic mean of a (discrete) probability distribution \( X \) is the expectation

\[
E_X(X) = \sum p_X(x) \times x
\]

with an integration analogue for continuous distributions.

*Usage:*
`Multinomial$mean(...)`

*Arguments:*
... Unused.

**Method** `variance()`: The variance of a distribution is defined by the formula

\[
\text{var}_X = E[X^2] - E[X]^2
\]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

*Usage:*
`Multinomial$variance(...)`

*Arguments:*
... Unused.

**Method** `skewness()`: The skewness of a distribution is defined by the third standardised moment,

\[
\text{sk}_X = E_X\left[\frac{x - \mu}{\sigma}\right]^3
\]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

*Usage:*
`Multinomial$skewness(...)`

*Arguments:*
... Unused.

**Method** `kurtosis()`: The kurtosis of a distribution is defined by the fourth standardised moment,

\[
\text{k}_X = E_X\left[\frac{x - \mu}{\sigma}\right]^4
\]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

*Usage:*
`Multinomial$kurtosis(excess = TRUE, ...)`

*Arguments:*
excess (logical(1))
   If TRUE (default) excess kurtosis returned.
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by
   \[- \sum (f_X) \log(f_X)\]
where \( f_X \) is the pdf of distribution X, with an integration analogue for continuous distributions.

Usage:
Multinomial$entropy(base = 2, ...)
Arguments:
base (integer(1))
   Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method mgf(): The moment generating function is defined by
   \( mgf_X(t) = E_X[exp(xt)] \)
where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
Multinomial$mgf(t, ...)
Arguments:
t (integer(1))
   t integer to evaluate function at.
... Unused.

Method cf(): The characteristic function is defined by
   \( cf_X(t) = E_X[exp(xti)] \)
where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
Multinomial$cf(t, ...)
Arguments:
t (integer(1))
   t integer to evaluate function at.
... Unused.

Method pgf(): The probability generating function is defined by
   \( pgf_X(z) = E_X[exp(z^x)] \)
where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
Multinomial$pgf(z, ...)
**Arguments:**

- `z (integer(1))`
  - `z` integer to evaluate probability generating function at.
  - ... Unused.

**Method** `setParameterValue()`: Sets the value(s) of the given parameter(s).

**Usage:**

```r
Multinomial$setParameterValue(
  ...,
  lst = list(...),
  error = "warn",
  resolveConflicts = FALSE
)
```

**Arguments:**

- `... ANY`
  - Named arguments of parameters to set values for. See examples.
- `lst (list(1))`
  - Alternative argument for passing parameters. List names should be parameter names and list values are the new values to set.
- `error (character(1))`
  - If "warn" then returns a warning on error, otherwise breaks if "stop".
- `resolveConflicts (logical(1))`
  - If FALSE (default) throws error if conflicting parameterisations are provided, otherwise automatically resolves them by removing all conflicting parameters.

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

**Usage:**

```r
Multinomial$clone(deep = FALSE)
```

**Arguments:**

- `deep` Whether to make a deep clone.

**References**


**See Also**

Other discrete distributions: Bernoulli, Binomial, Categorical, Degenerate, DiscreteUniform, EmpiricalMV, Empirical, Geometric, Hypergeometric, Logarithmic, NegativeBinomial, WeightedDiscrete

Other multivariate distributions: Dirichlet, EmpiricalMV, MultivariateNormal
MultivariateNormal

Multivariate Normal Distribution Class

Description

Mathematical and statistical functions for the Multivariate Normal distribution, which is commonly used to generalise the Normal distribution to higher dimensions, and is commonly associated with Gaussian Processes.

Details

The Multivariate Normal distribution parameterised with mean, \( \mu \), and covariance matrix, \( \Sigma \), is defined by the pdf,

\[
f(x_1, \ldots, x_k) = (2 \pi)^{-k/2} \det(\Sigma)^{-1/2} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)
\]

for \( \mu \in \mathbb{R}^k \) and \( \Sigma \in \mathbb{R}^{k \times k} \).

Sampling is performed via the Cholesky decomposition using \texttt{chol}.

Number of variables cannot be changed after construction.

Value

Returns an R6 object inheriting from class \texttt{SDistribution}.

Distribution support

The distribution is supported on the Reals and only when the covariance matrix is positive-definite.

Default Parameterisation

\texttt{MultiNorm(mean = rep(0, 2), cov = c(1, 0, 0, 1))}

Omitted Methods

\texttt{cdf} and \texttt{quantile} are omitted as no closed form analytic expression could be found, decorate with \texttt{FunctionImputation} for a numerical imputation.

Also known as

\texttt{N/A}

Super classes

\texttt{distr6::Distribution} \rightarrow \texttt{distr6::SDistribution} \rightarrow \texttt{MultivariateNormal}
**Public fields**

- **name**  Full name of distribution.
- **short_name**  Short name of distribution for printing.
- **description**  Brief description of the distribution.

**Active bindings**

- **properties**  Returns distribution properties, including skewness type and symmetry.

**Methods**

**Public methods:**

- `MultivariateNormal$new()`  
- `MultivariateNormal$mean()`  
- `MultivariateNormal$mode()`  
- `MultivariateNormal$variance()`  
- `MultivariateNormal$entropy()`  
- `MultivariateNormal$mgf()`  
- `MultivariateNormal$cf()`  
- `MultivariateNormal$pgf()`  
- `MultivariateNormal$getParameterValue()`  
- `MultivariateNormal$setParameterValue()`  
- `MultivariateNormal$clone()`

**Method** `new()`: Creates a new instance of this R6 class. Number of variables cannot be changed after construction.

**Usage:**

```r
MultivariateNormal$new(
  mean = rep(0, 2),
  cov = c(1, 0, 0, 1),
  prec = NULL,
  decorators = NULL
)
```

**Arguments:**

- **mean**  (numeric())  
  Vector of means, defined on the Reals.
- **cov**  (matrix()\|vector())  
  Covariance of the distribution, either given as a matrix or vector coerced to a matrix via `matrix(cov,nrow = K,byrow = FALSE)`. Must be semi-definite.
- **prec**  (matrix()\|vector())  
  Precision of the distribution, inverse of the covariance matrix. If supplied then cov is ignored. Given as a matrix or vector coerced to a matrix via `matrix(cov,nrow = K,byrow = FALSE)`. Must be semi-definite.
- **decorators**  (character())  
  Decorators to add to the distribution during construction.
Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) \times x \]

with an integration analogue for continuous distributions.

Usage:
MultivariateNormal$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
MultivariateNormal$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
MultivariateNormal$variance(...)

Arguments:
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

\[-\sum (f_X)\log(f_X)\]

where \( f_X \) is the pdf of distribution X, with an integration analogue for continuous distributions.

Usage:
MultivariateNormal$entropy(base = 2, ...)

Arguments:
base (integer(1))
Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method mgf(): The moment generating function is defined by

\[ mgf_X(t) = E_X[exp(xt)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X.
Usage:
MultivariateNormal$mgf(t, ...)

Arguments:
t (integer(1))
   t integer to evaluate function at.
... Unused.

Method cf(): The characteristic function is defined by
\[ cf_X(t) = E_X[exp(xt)] \]
where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
MultivariateNormal$cf(t, ...)

Arguments:
t (integer(1))
   t integer to evaluate function at.
... Unused.

Method pgf(): The probability generating function is defined by
\[ pgf_X(z) = E_X[exp(z^x)] \]
where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
MultivariateNormal$pgf(z, ...)

Arguments:
z (integer(1))
   z integer to evaluate probability generating function at.
... Unused.

Method getParameterValue(): Returns the value of the supplied parameter.

Usage:
MultivariateNormal$getParameterValue(id, error = "warn")

Arguments:
id character()
   id of parameter support to return.
error (character(1))
   If "warn" then returns a warning on error, otherwise breaks if "stop".

Method setParameterValue(): Sets the value(s) of the given parameter(s).

Usage:
MultivariateNormal$setParameterValue(
   ...,
   lst = list(...),
   error = "warn",
   resolveConflicts = FALSE
)
Negative Binomial Distribution Class

Arguments:

... ANY
  Named arguments of parameters to set values for. See examples.
lst (list(1))
  Alternative argument for passing parameters. List names should be parameter names and
  list values are the new values to set.
error (character(1))
  If "warn" then returns a warning on error, otherwise breaks if "stop".
resolveConflicts (logical(1))
  If FALSE (default) throws error if conflicting parameterisations are provided, otherwise au-
  tomatically resolves them by removing all conflicting parameters.

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

Usage:
MultivariateNormal$clone(deep = FALSE)

Arguments:
  deep  Whether to make a deep clone.

References

Michael P. McLaughlin.


See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral,
ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution,
Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal,\nNormal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT,
Triangular, Uniform, Wald, Weibull

Other multivariate distributions: Dirichlet, EmpiricalMV, Multinomial

Description

Mathematical and statistical functions for the Negative Binomial distribution, which is commonly
used to model the number of successes, trials or failures before a given number of failures or suc-
cesses.
NegativeBinomial

Details
The Negative Binomial distribution parameterised with number of failures before successes, \( n \), and
probability of success, \( p \), is defined by the pmf,

\[
f(x) = C(x + n - 1, n - 1) p^n (1 - p)^x
\]

for \( n = 0, 1, 2, \ldots \) and probability \( p \), where \( C(a, b) \) is the combination (or binomial coefficient)
function.

The Negative Binomial distribution can refer to one of four distributions (forms):

1. The number of failures before \( K \) successes (fbs)
2. The number of successes before \( K \) failures (sbf)
3. The number of trials before \( K \) failures (tbf)
4. The number of trials before \( K \) successes (tbs)

For each we refer to the number of \( K \) successes/failures as the size parameter.

Value
Returns an R6 object inheriting from class SDistribution.

Distribution support
The distribution is supported on \( 0, 1, 2, \ldots \) (for fbs and sbf) or \( n, n + 1, n + 2, \ldots \) (for tbf and tbs)
(see below).

Default Parameterisation
\[
\text{NBinom(size = 10, prob = 0.5, form = "fbs")}
\]

Omitted Methods
N/A

Also known as
N/A

Super classes
\[
distr6::Distribution \rightarrow distr6::SDistribution \rightarrow \text{NegativeBinomial}
\]

Public fields
name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.
Active bindings

properties Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:

- NegativeBinomial$new()
- NegativeBinomial$mean()
- NegativeBinomial$mode()
- NegativeBinomial$variance()
- NegativeBinomial$skewness()
- NegativeBinomial$kurtosis()
- NegativeBinomial$mgf()
- NegativeBinomial$cf()
- NegativeBinomial$pgf()
- NegativeBinomial$clone()

Method new(): Creates a new instance of this R6 class.

Usage:

NegativeBinomial$new(  
  size = NULL,  
  prob = NULL,  
  qprob = NULL,  
  mean = NULL,  
  form = NULL,  
  decorators = NULL  
)

Arguments:

size (integer(1))
  Number of trials/successes.
prob (numeric(1))
  Probability of success.
qprob (numeric(1))
  Probability of failure. If provided then prob is ignored. qprob = 1 - prob.
mean (numeric(1))
  Mean of distribution, alternative to prob and qprob.
form character(1))
  Form of the distribution, cannot be changed after construction. Options are to model the number of,
  • "fbs" - Failures before successes.
  • "sbf" - Successes before failures.
  • "tbf" - Trials before failures.
  • "tbs" - Trials before successes. Use $description to see the Negative Binomial form.
decorators (character())
  Decorators to add to the distribution during construction.
Method `mean()`: The arithmetic mean of a (discrete) probability distribution $X$ is the expectation 

$$E_X(X) = \sum p_X(x) * x$$

with an integration analogue for continuous distributions.

*Usage:*

NegativeBinomial$mean(...)

*Arguments:*

... Unused.

Method `mode()`: The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

*Usage:*

NegativeBinomial$mode(which = "all")

*Arguments:*

which (character(1) | numeric(1))

Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method `variance()`: The variance of a distribution is defined by the formula 

$$var_X = E[X^2] - E[X]^2$$

where $E_X$ is the expectation of distribution $X$. If the distribution is multivariate the covariance matrix is returned.

*Usage:*

NegativeBinomial$variance(...)

*Arguments:*

... Unused.

Method `skewness()`: The skewness of a distribution is defined by the third standardised moment, 

$$sk_X = E_X[\frac{x - \mu^3}{\sigma}]$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution.

*Usage:*

NegativeBinomial$skewness(...)

*Arguments:*

... Unused.

Method `kurtosis()`: The kurtosis of a distribution is defined by the fourth standardised moment, 

$$k_X = E_X[\frac{x - \mu^4}{\sigma}]$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.
Usage:
NegativeBinomial$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
  ... Unused.

Method mgf(): The moment generating function is defined by

$$mgf_X(t) = E_X[exp(xt)]$$

where X is the distribution and $E_X$ is the expectation of the distribution X.

Usage:
NegativeBinomial$mgf(t, ...)

Arguments:
t (integer(1))
  t integer to evaluate function at.
  ... Unused.

Method cf(): The characteristic function is defined by

$$cf_X(t) = E_X[exp(xti)]$$

where X is the distribution and $E_X$ is the expectation of the distribution X.

Usage:
NegativeBinomial$cf(t, ...)

Arguments:
t (integer(1))
  t integer to evaluate function at.
  ... Unused.

Method pgf(): The probability generating function is defined by

$$pgf_X(z) = E_X[exp(xz)]$$

where X is the distribution and $E_X$ is the expectation of the distribution X.

Usage:
NegativeBinomial$pgf(z, ...)

Arguments:
z (integer(1))
  z integer to evaluate probability generating function at.
  ... Unused.

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

Usage:
NegativeBinomial$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.
Normal

References

See Also
Other discrete distributions: Bernoulli, Binomial, Categorical, Degenerate, DiscreteUniform, Empirical, EmpiricalMV, Geometric, Hypergeometric, Logarithmic, Multinomial, WeightedDiscrete
Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

Normal Distribution Class

Description
Mathematical and statistical functions for the Normal distribution, which is commonly used in significance testing, for representing models with a bell curve, and as a result of the central limit theorem.

Details
The Normal distribution parameterised with variance, $\sigma^2$, and mean, $\mu$, is defined by the pdf,

$$f(x) = \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)/\sqrt{2\pi\sigma^2}$$

for $\mu \in \mathbb{R}$ and $\sigma^2 > 0$.

Value
Returns an R6 object inheriting from class SDistribution.

Distribution support
The distribution is supported on the Reals.

Default Parameterisation
Norm(mean = 0, var = 1)

Omitted Methods
N/A
Also known as

Also known as the Gaussian distribution.

Super classes

\[
distr6::Distribution \rightarrow distr6::SDistribution \rightarrow \text{Normal}
\]

Public fields

- name Full name of distribution.
- short_name Short name of distribution for printing.
- description Brief description of the distribution.
- packages Packages required to be installed in order to construct the distribution.

Methods

Public methods:

- \text{Normal}\$new()
- \text{Normal}\$mean()
- \text{Normal}\$mode()
- \text{Normal}\$variance()
- \text{Normal}\$skewness()
- \text{Normal}\$kurtosis()
- \text{Normal}\$entropy()
- \text{Normal}\$mgf()
- \text{Normal}\$cf()
- \text{Normal}\$pgf()
- \text{Normal}\$clone()

Method \text{new}(): Creates a new instance of this R6 class.

Usage:
\text{Normal}\$new(mean = NULL, var = NULL, sd = NULL, prec = NULL, decorators = NULL)

Arguments:

- mean (numeric(1)) Mean of the distribution, defined on the Reals.
- var (numeric(1)) Variance of the distribution, defined on the positive Reals.
- sd (numeric(1)) Standard deviation of the distribution, defined on the positive Reals. \( sd = \sqrt{\text{var}} \). If provided then var ignored.
- prec (numeric(1)) Precision of the distribution, defined on the positive Reals. \( prec = 1/\text{var} \). If provided then var ignored.
- decorators (character()) Decorators to add to the distribution during construction.
**Method** `mean()`: The arithmetic mean of a (discrete) probability distribution $X$ is the expectation

$$E_X(X) = \sum p_X(x) \times x$$

with an integration analogue for continuous distributions.

*Usage:*

`Normal$mean(...)`

*Arguments:*

... Unused.

**Method** `mode()`: The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

*Usage:*

`Normal$mode(which = "all")`

*Arguments:*

`which` (character(1) | numeric(1))

Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

**Method** `variance()`: The variance of a distribution is defined by the formula

$$var_X = E[X^2] - E[X]^2$$

where $E_X$ is the expectation of distribution $X$. If the distribution is multivariate the covariance matrix is returned.

*Usage:*

`Normal$variance(...)`

*Arguments:*

... Unused.

**Method** `skewness()`: The skewness of a distribution is defined by the third standardised moment,

$$sk_X = E_X\left(\frac{x - \mu}{\sigma}^3\right)$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution.

*Usage:*

`Normal$skewness(...)`

*Arguments:*

... Unused.

**Method** `kurtosis()`: The kurtosis of a distribution is defined by the fourth standardised moment,

$$k_X = E_X\left(\frac{x - \mu}{\sigma}^4\right)$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.
Usage:
Normal$kurtosis(excess = TRUE, ...)
Arguments:
excess (logical(1))
If TRUE (default) excess kurtosis returned.
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by
\[- \sum (f_X) \log(f_X)\]
where \(f_X\) is the pdf of distribution X, with an integration analogue for continuous distributions.
Usage:
Normal$entropy(base = 2, ...)
Arguments:
base (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method mgf(): The moment generating function is defined by
\[mgf_X(t) = E_X[exp(zt)]\]
where X is the distribution and \(E_X\) is the expectation of the distribution X.
Usage:
Normal$mgf(t, ...)
Arguments:
t (integer(1))
  t integer to evaluate function at.
... Unused.

Method cf(): The characteristic function is defined by
\[cf_X(t) = E_X[exp(zt)]\]
where X is the distribution and \(E_X\) is the expectation of the distribution X.
Usage:
Normal$cf(t, ...)
Arguments:
t (integer(1))
  t integer to evaluate function at.
... Unused.

Method pgf(): The probability generating function is defined by
\[pgf_X(z) = E_X[exp(z^t)]\]
where X is the distribution and \(E_X\) is the expectation of the distribution X.
Usage:
Normal$pgf(z, ...)

Arguments:
z (integer(1))
   z integer to evaluate probability generating function at.
...

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

Usage:
Normal$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References
Michael P. McLaughlin.

See Also
Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral,
ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution,
Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal,
MultivariateNormal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral,
StudentT, Triangular, Uniform, Wald, Weibull
Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical,
Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang,
Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz,
Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal,
NegativeBinomial, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral,
StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

NormalKernel

Normal Kernel

Description
Mathematical and statistical functions for the NormalKernel kernel defined by the pdf,

$$f(x) = \frac{\exp(-x^2/2)}{\sqrt{2\pi}}$$

over the support $$x \in \mathbb{R}$$.

Details
We use the erf and erfinv error and inverse error functions from pracma.
Super classes

\texttt{distr6::Distribution} $\rightarrow$ \texttt{distr6::Kernel} $\rightarrow$ \texttt{NormalKernel}

Public fields

- \texttt{name} Full name of distribution.
- \texttt{short.name} Short name of distribution for printing.
- \texttt{description} Brief description of the distribution.
- \texttt{packages} Packages required to be installed in order to construct the distribution.

Methods

**Public methods:**

- \texttt{NormalKernel$new()}
- \texttt{NormalKernel$pdfSquared2Norm()}
- \texttt{NormalKernel$variance()}
- \texttt{NormalKernel$clone()}

**Method new():** Creates a new instance of this R6 class.

*Usage:*
\texttt{NormalKernel$new(\texttt{decorators} = \texttt{NULL})}

*Arguments:*

- \texttt{decorators} (character())
  Decorators to add to the distribution during construction.

**Method pdfSquared2Norm():** The squared 2-norm of the pdf is defined by

\[
\int_a^b (f_X(u))^2 \, du
\]

where \(X\) is the Distribution, \(f_X\) is its pdf and \(a, b\) are the distribution support limits.

*Usage:*
\texttt{NormalKernel$pdfSquared2Norm(\texttt{x = 0}, \texttt{upper = Inf})}

*Arguments:*

- \texttt{x} (numeric(1))
  Amount to shift the result.
- \texttt{upper} (numeric(1))
  Upper limit of the integral.

**Method variance():** The variance of a distribution is defined by the formula

\[
var_X = E[X^2] - E[X]^2
\]

where \(E_X\) is the expectation of distribution \(X\). If the distribution is multivariate the covariance matrix is returned.

*Usage:*
NormalKernel$variance(...)

Arguments:
... Unused.

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

Usage:
NormalKernel$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also
Other kernels: Cosine, Epanechnikov, LogisticKernel, Quartic, Sigmoid, Silverman, TriangularKernel, Tricube, Triweight, UniformKernel

---

Pareto  Pareto Distribution Class

Description
Mathematical and statistical functions for the Pareto distribution, which is commonly used in Economics to model the distribution of wealth and the 80-20 rule.

Details
The Pareto distribution parameterised with shape, \( \alpha \), and scale, \( \beta \), is defined by the pdf,

\[
f(x) = (\alpha \beta^\alpha) / (x^{\alpha+1})
\]

for \( \alpha, \beta > 0 \).

Currently this is implemented as the Type I Pareto distribution, other types will be added in the future. Characteristic function is omitted as no suitable incomplete gamma function with complex inputs implementation could be found.

Value
Returns an R6 object inheriting from class SDistribution.

Distribution support
The distribution is supported on \([\beta, \infty)\).

Default Parameterisation
Pare(shape = 1, scale = 1)
Omitted Methods
N/A

Also known as
N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Pareto

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.

Active bindings

properties Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:
• Pareto$new()
• Pareto$mean()
• Pareto$mode()
• Pareto$median()
• Pareto$variance()
• Pareto$skewness()
• Pareto$kurtosis()
• Pareto$entropy()
• Pareto$mgf()
• Pareto$pgf()
• Pareto$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Pareto$new(shape = NULL, scale = NULL, decorators = NULL)

Arguments:
shape (numeric(1))
  Shape parameter, defined on the positive Reals.
scale (numeric(1))
  Scale parameter, defined on the positive Reals.
decorators (character())
Decorators to add to the distribution during construction.

**Method mean()**: The arithmetic mean of a (discrete) probability distribution $X$ is the expectation

$$E_X(X) = \sum p_X(x) \times x$$

with an integration analogue for continuous distributions.

*Usage:*

Pareto$mean(...)

*Arguments:*

... Unused.

**Method mode()**: The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

*Usage:*

Pareto$mode(which = "all")

*Arguments:*

which (character(1) | numeric(1))
Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

**Method median()**: Returns the median of the distribution. If an analytical expression is available returns distribution median, otherwise if symmetric returns self$mean, otherwise returns self$quantile(0.5).

*Usage:*

Pareto$median()

**Method variance()**: The variance of a distribution is defined by the formula

$$var_X = E[X^2] - E[X]^2$$

where $E_X$ is the expectation of distribution $X$. If the distribution is multivariate the covariance matrix is returned.

*Usage:*

Pareto$variance(...)

*Arguments:*

... Unused.

**Method skewness()**: The skewness of a distribution is defined by the third standardised moment,

$$sk_X = E_X \left[ \frac{x - \mu}{\sigma}^3 \right]$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution.

*Usage:*

...
**Method** `skewness()`: The skewness of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^4 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. **Usage:**

`Pareto$skewness(...)`

**Arguments:**

... Unused.

**Method** `kurtosis()`: The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^4 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3. **Usage:**

`Pareto$kurtosis(excess = TRUE, ...)`

**Arguments:**

excess (logical(1))

If `TRUE` (default) excess kurtosis returned. **Unused.**

**Method** `entropy()`: The entropy of a (discrete) distribution is defined by

\[ -\sum (f_X \log(f_X)) \]

where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions. **Usage:**

`Pareto$entropy(base = 2, ...)`

**Arguments:**

base (integer(1))

Base of the entropy logarithm, default = 2 (Shannon entropy) **Unused.**

**Method** `mgf()`: The moment generating function is defined by

\[ mgf_X(t) = E_X[exp(xt)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \). **Usage:**

`Pareto$mgf(t, ...)`

**Arguments:**

t (integer(1))

t integer to evaluate function at. **Unused.**

**Method** `pgf()`: The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(zx)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).
plot.Distribution 207

Usage:
Pareto$pgf(z, ...)

Arguments:
z (integer(1))  
z integer to evaluate probability generating function at.
... Unused.

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

Usage:
Pareto$clone(deep = FALSE)

Arguments:
   deep  Whether to make a deep clone.

References


See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

plot.Distribution  Plot Distribution Functions for a distr6 Object

Description

Six plots, which can be selected with fun are available for discrete and continuous univariate distributions: pdf, cdf, quantile, survival, hazard and cumulative hazard. By default, the first two are plotted side by side.
plot.Distribution

Usage

```r
## S3 method for class 'Distribution'
plot(
  x,
  fun = c("pdf", "cdf"),
  npoints = 3000,
  plot = TRUE,
  ask = FALSE,
  arrange = TRUE,
  ...
)
```

Arguments

- `x` : distr6 object.
- `fun` : vector of functions to plot, one or more of: "pdf","cdf","quantile", "survival", "hazard","cumhazard", and "all"; partial matching available.
- `npoints` : number of evaluation points.
- `plot` : logical; if TRUE (default), figures are displayed in the plot window; otherwise a `data.table::data.table()` of points and calculated values is returned.
- `ask` : logical; if TRUE, the user is asked before each plot, see `graphics::par()`.
- `arrange` : logical; if TRUE (default), margins are automatically adjusted with `graphics::layout()` to accommodate all plotted functions.
- `...` : graphical parameters, see details.

Details

The evaluation points are calculated using inverse transform on a uniform grid between 0 and 1 with length given by `npoints`. Therefore any distribution without an analytical quantile method will first need to be imputed with the `FunctionImputation` decorator.

The order that the functions are supplied to `fun` determines the order in which they are plotted, however this is ignored if ask is TRUE. If ask is TRUE then arrange is ignored. For maximum flexibility in plotting layouts, set arrange and ask to FALSE.

The graphical parameters passed to ... can either apply to all plots or selected plots. If parameters in `par` are prefixed with the plotted function name, then the parameter only applies to that function, otherwise it applies to them all. See examples for a clearer description.

Author(s)

Chengyang Gao, Runlong Yu and Shuhan Liu

See Also

`lines.Distribution`
Examples

```r
## Not run:
# Plot pdf and cdf of Normal
plot(Normal$new())

# Colour both plots red
plot(Normal$new(), col = "red")

# Change the colours of individual plotted functions
plot(Normal$new(), pdf_col = "red", cdf_col = "green")

# Interactive plotting in order - par still works here
plot(Geometric$new(),
     fun = "all", ask = TRUE, pdf_col = "black",
     cdf_col = "red", quantile_col = "blue", survival_col = "purple",
     hazard_col = "brown", cumhazard_col = "yellow"
)

# Return plotting structure
x <- plot(Gamma$new(), plot = FALSE)
## End(Not run)
```

plot.VectorDistribution

Plotting Distribution Functions for a VectorDistribution

Description

Helper function to more easily plot distributions inside a VectorDistribution.

Usage

```r
## S3 method for class 'VectorDistribution'
plot(x, fun = "pdf", topn, ind, cols, ...)
```

Arguments

- `x` VectorDistribution.
- `fun` function to plot, one of: "pdf","cdf","quantile", "survival", "hazard", "cumhazard".
- `topn` integer. First n distributions in the VectorDistribution to plot.
- `ind` integer. Indices of the distributions in the VectorDistribution to plot. If given then topn is ignored.
- `cols` character. Vector of colours for plotting the curves. If missing 1:9 are used.
- `...` Other parameters passed to plot.Distribution.
Details

If `topn` and `ind` are both missing then all distributions are plotted if there are 10 or less in the vector, otherwise the function will error.

See Also

`plot.Distribution`

Examples

```r
df <- VectorDistribution$new(list(Normal$new(), Normal$new(mean = 2)))
plot(df)
plot(df, fun = "surv")
plot(df, fun = "quantile", ylim = c(-4, 4), col = c("blue", "purple"))
```

---

**Poisson**

**Poisson Distribution Class**

Description

Mathematical and statistical functions for the Poisson distribution, which is commonly used to model the number of events occurring in at a constant, independent rate over an interval of time or space.

Details

The Poisson distribution parameterised with arrival rate, $\lambda$, is defined by the pmf,

$$f(x) = \frac{\lambda^x \exp(-\lambda)}{x!}$$

for $\lambda > 0$.

Value

Returns an R6 object inheriting from class `SDistribution`.

Distribution support

The distribution is supported on the Naturals.

Default Parameterisation

Pois(rate = 1)
**Poisson**

**Omitted Methods**

N/A

**Also known as**

N/A

**Super classes**

\[ \texttt{distr6::Distribution} \rightarrow \texttt{distr6::SDistribution} \rightarrow \texttt{Poisson} \]

**Public fields**

- name: Full name of distribution.
- short_name: Short name of distribution for printing.
- description: Brief description of the distribution.
- packages: Packages required to be installed in order to construct the distribution.

**Methods**

**Public methods:**

- `Poisson$new()`
- `Poisson$mean()`
- `Poisson$mode()`
- `Poisson$variance()`
- `Poisson$skewness()`
- `Poisson$kurtosis()`
- `Poisson$mgf()`
- `Poisson$cf()`
- `Poisson$pgf()`
- `Poisson$clone()`

**Method** `new()`: Creates a new instance of this R6 class.

**Usage:**

`Poisson$new(rate = NULL, decorators = NULL)`

**Arguments:**

- `rate` (numeric(1))
  
  Rate parameter of the distribution, defined on the positive Reals.
- `decorators` (character())
  
  Decorators to add to the distribution during construction.

**Method** `mean()`: The arithmetic mean of a (discrete) probability distribution \( X \) is the expectation

\[
E_X(X) = \sum p_X(x) \cdot x
\]

with an integration analogue for continuous distributions.
Usage:
Poisson$mean(...)

Arguments:
... Unused.

Method `mode()`: The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Poisson$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method `variance()`: The variance of a distribution is defined by the formula

\[
\text{var}_X = E[X^2] - [E[X]]^2
\]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
Poisson$variance(...)

Arguments:
... Unused.

Method `skewness()`: The skewness of a distribution is defined by the third standardised moment,

\[
\text{sk}_X = E_X \left( \frac{x - \mu}{\sigma} \right)^3
\]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:
Poisson$skewness(...)

Arguments:
... Unused.

Method `kurtosis()`: The kurtosis of a distribution is defined by the fourth standardised moment,

\[
\kappa_X = E_X \left( \frac{x - \mu}{\sigma} \right)^4
\]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Poisson$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
    If TRUE (default) excess kurtosis returned.
... Unused.

**Method mgf()**: The moment generating function is defined by

\[ mgf_X(t) = E_X[exp(xt)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

*Usage:*

```
Poisson$mgf(t, ...)```

*Arguments:*

- `t` (integer(1))
  - `t` integer to evaluate function at.
... Unused.

**Method cf()**: The characteristic function is defined by

\[ cf_X(t) = E_X[exp(xti)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

*Usage:*

```
Poisson$cf(t, ...)```

*Arguments:*

- `t` (integer(1))
  - `t` integer to evaluate function at.
... Unused.

**Method pgf()**: The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(z^x)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

*Usage:*

```
Poisson$pgf(z, ...)```

*Arguments:*

- `z` (integer(1))
  - `z` integer to evaluate probability generating function at.
... Unused.

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

*Usage:*

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

*Arguments:*

- `deep` Whether to make a deep clone.
ProductDistribution

Description

A wrapper for creating the product distribution of multiple independent probability distributions.

Usage

```r
## S3 method for class 'Distribution'

x * y
```

Arguments

- `x, y` Distribution

Details

A product distribution is defined by

\[
F_P(X_1 = x_1, ..., X_N = x_N) = F_{X_1}(x_1) \ast ... \ast F_{X_N}(x_N)
\]

#nolint where \( F_P \) is the cdf of the product distribution and \( X_1, ..., X_N \) are independent distributions.

Super classes

distr6::Distribution -> distr6::DistributionWrapper -> distr6::VectorDistribution

-> ProductDistribution
Methods

Public methods:

• ProductDistribution$new()
• ProductDistribution$strprint()
• ProductDistribution$pdf()
• ProductDistribution$cdf()
• ProductDistribution$quantile()
• ProductDistribution$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
ProductDistribution$new(
  distlist = NULL,
  distribution = NULL,
  params = NULL,
  shared_params = NULL,
  name = NULL,
  short_name = NULL,
  decorators = NULL,
  vecdist = NULL,
  ids = NULL
)

Arguments:

distlist (list())
  List of Distributions.

distribution (character(1))
  Should be supplied with params and optionally shared_params as an alternative to distlist. Much faster implementation when only one class of distribution is being wrapped. distribution is the full name of one of the distributions in listDistributions(), or "Distribution" if constructing custom distributions. See examples in VectorDistribution.

params (list()|data.frame())
  Parameters in the individual distributions for use with distribution. Can be supplied as a list, where each element is the list of parameters to set in the distribution, or as an object coercable to data.frame, where each column is a parameter and each row is a distribution. See examples in VectorDistribution.

shared_params (list())
  If any parameters are shared when using the distribution constructor, this provides a much faster implementation to list and query them together. See examples in VectorDistribution.

name (character(1))
  Optional name of wrapped distribution.

short_name (character(1))
  Optional short name/ID of wrapped distribution.

decorators (character())
  Decorators to add to the distribution during construction.
vecdist VectorDistribution

Alternative constructor to directly create this object from an object inheriting from VectorDistribution.

ids (character())

Optional ids for wrapped distributions in vector, should be unique and of same length as the number of distributions.

Examples:

```r
\dontrun{
ProductDistribution$new(list(Binomial$new(
  prob = 0.5,
  size = 10
), Normal$new(mean = 15)))
}
```

```r
ProductDistribution$new(
  distribution = "Binomial",
  params = list(
    list(prob = 0.1, size = 2),
    list(prob = 0.6, size = 4),
    list(prob = 0.2, size = 6)
  )
)
```

# Equivalently

```r
ProductDistribution$new(
  distribution = "Binomial",
  params = data.table::data.table(prob = c(0.1, 0.6, 0.2), size = c(2, 4, 6))
)
```

Method `strprint()`:

Printable string representation of the ProductDistribution. Primarily used internally.

Usage:

```r
ProductDistribution$strprint(n = 10)
```

Arguments:

- **n** (integer(1))
  
  Number of distributions to include when printing.

Method `pdf()`:

Probability density function of the product distribution. Computed by

\[
f_P(X_1 = x_1, ..., X_N = x_N) = \prod_i f_{X_i}(x_i)
\]

where \( f_{X_i} \) are the pdfs of the wrapped distributions.

Usage:

```r
ProductDistribution$pdf(..., log = FALSE, simplify = TRUE, data = NULL)
```

Arguments:
... (numeric())
  Points to evaluate the function at Arguments do not need to be named. The length of each
  argument corresponds to the number of points to evaluate, the number of arguments corre-
  sponds to the number of variables in the distribution. See examples.

log (logical(1))
  If TRUE returns the logarithm of the probabilities. Default is FALSE.
simplify logical(1)
  If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.
data array
  Alternative method to specify points to evaluate. If univariate then rows correspond with
  number of points to evaluate and columns correspond with number of variables to evalu-
  ate. In the special case of VectorDistributions of multivariate distributions, then the third
  dimension corresponds to the distribution in the vector to evaluate.

Examples:
  p <- ProductDistribution$new(list(
    Binomial$new(prob = 0.5, size = 10),
    Binomial$new())
  p$pdf(1:5)
  p$pdf(1, 2)
  p$pdf(1:2)

Method cdf(): Cumulative distribution function of the product distribution. Computed by

\[ F_P(X_1 = x_1, ..., X_N = x_N) = \prod_i F_{X_i}(x_i) \]

where \( F_{X_i} \) are the cdfs of the wrapped distributions.

Usage:
  ProductDistribution$cdf(
    ..., 
    lower.tail = TRUE, 
    log.p = FALSE, 
    simplify = TRUE, 
    data = NULL
  )

Arguments:
... (numeric())
  Points to evaluate the function at Arguments do not need to be named. The length of each
  argument corresponds to the number of points to evaluate, the number of arguments corre-
  sponds to the number of variables in the distribution. See examples.

lower.tail (logical(1))
  If TRUE (default), probabilities are \( X \leq x \), otherwise, \( P(X > x) \).

log.p (logical(1))
  If TRUE returns the logarithm of the probabilities. Default is FALSE.
simplify logical(1)
  If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.
data array

Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

Examples:

```r
p <- ProductDistribution$new(list(
  Binomial$new(prob = 0.5, size = 10),
  Binomial$new()))
p$cdf(1:5)
p$cdf(1, 2)
p$cdf(1:2)
```

Method quantile(): The quantile function is not implemented for product distributions.

Usage:

```r
ProductDistribution$quantile(
  ..., 
  lower.tail = TRUE,
  log.p = FALSE,
  simplify = TRUE,
  data = NULL
)
```

Arguments:

```r
... (numeric())
  Points to evaluate the function at Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.
lower.tail (logical(1))
  If TRUE (default), probabilities are \( X \leq x \), otherwise, \( P(X > x) \).
log.p (logical(1))
  If TRUE returns the logarithm of the probabilities. Default is FALSE.
simplify (logical(1))
  If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.
```

data array

Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

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

Usage:

```r
ProductDistribution$clone(deep = FALSE)
```

Arguments:

deep  Whether to make a deep clone.
See Also

Other wrappers: `Convolution`, `DistributionWrapper`, `HuberizedDistribution`, `MixtureDistribution`, `TruncatedDistribution`, `VectorDistribution`

Examples

```r
## ------------------------------------------------
## Method `ProductDistribution$new`
## ------------------------------------------------

## Not run:
ProductDistribution$new(list(Binomial$new(
    prob = 0.5,
    size = 10
), Normal$new(mean = 15)))

ProductDistribution$new(
    distribution = "Binomial",
    params = list(
        list(prob = 0.1, size = 2),
        list(prob = 0.6, size = 4),
        list(prob = 0.2, size = 6)
    )
)

# Equivalently
ProductDistribution$new(
    distribution = "Binomial",
    params = data.table::data.table(prob = c(0.1, 0.6, 0.2), size = c(2, 4, 6))
)

## End(Not run)

## ------------------------------------------------
## Method `ProductDistribution$pdf`
## ------------------------------------------------

p <- ProductDistribution$new(list(  
    Binomial$new(prob = 0.5, size = 10),
    Binomial$new())
)
p$pdf(1:5)
p$pdf(1, 2)
p$pdf(1:2)

## ------------------------------------------------
## Method `ProductDistribution$cdf`
## ------------------------------------------------

p <- ProductDistribution$new(list(  
    Binomial$new(prob = 0.5, size = 10),
    Binomial$new())
)
QQ-Plotting

Description

Quantile-quantile plots are used to compare a "theoretical" or empirical distribution to a reference distribution. They can also compare the quantiles of two reference distributions.

Usage

qqplot(x, y, npoints = 3000, idline = TRUE, plot = TRUE, ...)

Arguments

x distr6 object or numeric vector.
y distr6 object or numeric vector.
npoints number of evaluation points.
idline logical; if TRUE (default), the line \( y = x \) is plotted
plot logical; if TRUE (default), figures are displayed in the plot window; otherwise a data.table::data.table of points and calculated values is returned.
... graphical parameters.

Details

If \( x \) or \( y \) are given as numeric vectors then they are first passed to the Empirical distribution. The Empirical distribution is a discrete distribution so quantiles are equivalent to the the Type 1 method in quantile.

Author(s)

Chijing Zeng

See Also

plot.Distribution for plotting a distr6 object.

Examples

qqplot(Normal$new(mean = 15, sd = sqrt(30)), ChiSquared$new(df = 15))
qqplot(rt(200, df = 5), rt(300, df = 5),
   main = "QQ-Plot", xlab = "t-200",
   ylab = "t-300"
)
qqplot(Normal$new(mean = 2), rnorm(100, mean = 3))
Description
Mathematical and statistical functions for the Quartic kernel defined by the pdf,

\[ f(x) = \frac{15}{16}(1 - x^2)^2 \]

over the support \( x \in (-1, 1) \).

Details
Quantile is omitted as no closed form analytic expression could be found, decorate with Function-Imputation for numeric results.

Super classes
distr6::Distribution -> distr6::Kernel -> Quartic

Public fields

- name: Full name of distribution.
- short_name: Short name of distribution for printing.
- description: Brief description of the distribution.

Methods

Public methods:
- Quartic$pdfSquared2Norm()
- Quartic$cdfSquared2Norm()
- Quartic$variance()
- Quartic$clone()

Method pdfSquared2Norm(): The squared 2-norm of the pdf is defined by

\[ \int_a^b (f_X(u))^2 du \]

where \( X \) is the Distribution, \( f_X \) is its pdf and \( a, b \) are the distribution support limits.

Usage:
Quartic$pdfSquared2Norm(x = 0, upper = Inf)

Arguments:
x (numeric(1))
Amount to shift the result.
upper (numeric(1))
  Upper limit of the integral.

**Method** cdfSquared2Norm(): The squared 2-norm of the cdf is defined by

\[ \int_a^b (F_X(u))^2 du \]

where X is the Distribution, \( F_X \) is its pdf and \( a, b \) are the distribution support limits.

**Usage:**
Quartic$cdfSquared2Norm(x = 0, upper = 0)

**Arguments:**
- \( x \) (numeric(1))
  - Amount to shift the result.
- \( upper \) (numeric(1))
  - Upper limit of the integral.

**Method** variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

**Usage:**
Quartic$variance(...)

**Arguments:**
- ... Unused.

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

**Usage:**
Quartic$clone(deep = FALSE)

**Arguments:**
- deep Whether to make a deep clone.

**See Also**

Other kernels: Cosine, Epanechnikov, LogisticKernel, NormalKernel, Sigmoid, Silverman, TriangularKernel, Tricube, Triweight, UniformKernel
Rayleigh Distribution Class

Description

Mathematical and statistical functions for the Rayleigh distribution, which is commonly used to model random complex numbers..

Details

The Rayleigh distribution parameterised with mode (or scale), $\alpha$, is defined by the pdf,

$$f(x) = \frac{x}{\alpha^2} e^{-(x^2)/(2\alpha^2)}$$

for $\alpha > 0$.

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on $[0, \infty)$.

Default Parameterisation

Rayl(mode = 1)

Omitted Methods

N/A

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> Rayleigh

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.
Methods

Public methods:

• Rayleigh$new()
• Rayleigh$mean()
• Rayleigh$mode()
• Rayleigh$median()
• Rayleigh$variance()
• Rayleigh$skewness()
• Rayleigh$kurtosis()
• Rayleigh$entropy()
• Rayleigh$pgf()
• Rayleigh$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Rayleigh$new(mode = NULL, decorators = NULL)

Arguments:
mode (numeric(1))
  Mode of the distribution, defined on the positive Reals. Scale parameter.
decorators (character())
  Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) \cdot x \]

with an integration analogue for continuous distributions.

Usage:
Rayleigh$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Rayleigh$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method median(): Returns the median of the distribution. If an analytical expression is available returns distribution median, otherwise if symmetric returns self$mean, otherwise returns self$quantile(0.5).
Usage:
Rayleigh$median()

Method variance(): The variance of a distribution is defined by the formula

$$var_X = E[X^2] - E[X]^2$$

where $E_X$ is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
Rayleigh$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

$$sk_X = E_X\left[\frac{x - \mu}{\sigma}\right]^3$$

where $E_X$ is the expectation of distribution X, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution.

Usage:
Rayleigh$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

$$k_X = E_X\left[\frac{x - \mu}{\sigma}\right]^4$$

where $E_X$ is the expectation of distribution X, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Rayleigh$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
   If TRUE (default) excess kurtosis returned.
   ... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

$$- \sum (f_X log(f_X))$$

where $f_X$ is the pdf of distribution X, with an integration analogue for continuous distributions.

Usage:
Rayleigh$entropy(base = 2, ...
Arguments:
base (integer(1))
Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method pgf(): The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(zx)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X.

Usage:
Rayleigh$pgf(z, ...)

Arguments:
z (integer(1))
\( z \) integer to evaluate probability generating function at.
... Unused.

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

Usage:
Rayleigh$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References

See Also
Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete
**Description**

Replicates a constructed distribution into either a

- `VectorDistribution` (class = "vector")
- `ProductDistribution` (class = "product")
- `MixtureDistribution` (class = "mixture")

If the distribution is not a custom `Distribution` then uses the more efficient `distribution/params` constructor, otherwise uses `distlist`.

**Usage**

```r
## S3 method for class 'Distribution'
rep(x, times, class = c("vector", "product", "mixture"), ...)
```

**Arguments**

- `x`  
  `Distribution`
- `times`  
  (integer(1)) Number of times to replicate the distribution
- `class`  
  (character(1)) What type of vector to create, see description.
- `...`  
  Additional arguments, currently unused.

**Examples**

```r
rep(Binomial$new(), 10)
rep(Gamma$new(), 2, class = "product")
```

---

**SDistribution**  
*Abstract Special Distribution Class*

**Description**

Abstract class that cannot be constructed directly.

**Value**

Returns error. Abstract classes cannot be constructed directly.

**Super class**

`distr6::Distribution` -> `SDistribution`
Public fields

- **package**: Deprecated, use $packages instead.
- **packages**: Packages required to be installed in order to construct the distribution.

Methods

**Public methods:**

- `SDistribution$new()`
- `SDistribution$clone()`

**Method new()**: Creates a new instance of this R6 class.

*Usage:*

```r
SDistribution$new(
  decorators,
  support,
  type,
  symmetry = c("asymmetric", "symmetric")
)
```

*Arguments:*

- **decorators (character())**: Decorators to add to the distribution during construction.
- **support [set6::Set]**: Support of the distribution.
- **type [set6::Set]**: Type of the distribution.
- **symmetry character(1)**: Distribution symmetry type, default "asymmetric".

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

*Usage:*

```r
SDistribution$clone(deep = FALSE)
```

*Arguments:*

- **deep**: Whether to make a deep clone.

---

**ShiftedLoglogistic**

**Shifted Log-Logistic Distribution Class**

**Description**

Mathematical and statistical functions for the Shifted Log-Logistic distribution, which is commonly used in survival analysis for its non-monotonic hazard as well as in economics, a generalised variant of Loglogistic.
Details

The Shifted Log-Logistic distribution parameterised with shape, $\beta$, scale, $\alpha$, and location, $\gamma$, is defined by the pdf,

$$f(x) = \frac{\beta}{\alpha}((x - \gamma)/\alpha)^{\beta-1}(1 + ((x - \gamma)/\alpha)^{\beta})^{-2}$$

for $\alpha, \beta > 0$ and $\gamma \geq 0$.

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on the non-negative Reals.

Default Parameterisation

ShiftLLogis(scale = 1, shape = 1, location = 0)

Omitted Methods

N/A

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> ShiftedLoglogistic

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.

Active bindings

properties Returns distribution properties, including skewness type and symmetry.
## Methods

### Public methods:

- `ShiftedLoglogistic$new()`
- `ShiftedLoglogistic$mean()`
- `ShiftedLoglogistic$mode()`
- `ShiftedLoglogistic$median()`
- `ShiftedLoglogistic$variance()`
- `ShiftedLoglogistic$pgf()`
- `ShiftedLoglogistic$clone()`

#### Method `new()`:

Creates a new instance of this R6 class.

**Usage:**

```r
ShiftedLoglogistic$new(
  scale = NULL,
  shape = NULL,
  location = NULL,
  rate = NULL,
  decorators = NULL
)
```

**Arguments:**

- `scale` (numeric(1))
  
  Scale parameter of the distribution, defined on the positive Reals. `scale = 1/rate`. If provided `rate` is ignored.

- `shape` (numeric(1))
  
  Shape parameter, defined on the positive Reals.

- `location` (numeric(1))
  
  Location parameter, defined on the Reals.

- `rate` (numeric(1))
  
  Rate parameter of the distribution, defined on the positive Reals.

- `decorators` (character())
  
  Decorators to add to the distribution during construction.

#### Method `mean()`:

The arithmetic mean of a (discrete) probability distribution $X$ is the expectation

$$E_X(X) = \sum p_X(x) \times x$$

with an integration analogue for continuous distributions.

**Usage:**

```r
ShiftedLoglogistic$mean(...)
```

**Arguments:**

- `...` Unused.

#### Method `mode()`:

The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).
Usage:
ShiftedLoglogistic$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies
  which mode to return.

Method median(): Returns the median of the distribution. If an analytical expression is avail-
able returns distribution median, otherwise if symmetric returns self$mean, otherwise returns
self$quantile(0.5).

Usage:
ShiftedLoglogistic$median()

Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = \mathbb{E}[X^2] - \mathbb{E}[X]^2 \]

where \( \mathbb{E}_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance
matrix is returned.

Usage:
ShiftedLoglogistic$variance(...)

Arguments:
... Unused.

Method pgf(): The probability generating function is defined by

\[ \text{pgf}_X(z) = \mathbb{E}_X[\exp(z^X)] \]

where \( X \) is the distribution and \( \mathbb{E}_X \) is the expectation of the distribution \( X \).

Usage:
ShiftedLoglogistic$pgf(z, ...)

Arguments:
z (integer(1))
  \( z \) integer to evaluate probability generating function at.
... Unused.

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

Usage:
ShiftedLoglogistic$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References

Michael P. McLaughlin.
See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

**Sigmoid**

**Sigmoid Kernel**

**Description**

Mathematical and statistical functions for the Sigmoid kernel defined by the pdf,

\[ f(x) = \frac{2}{\pi} \left( e^{x} + e^{-x} \right)^{-1} \]

over the support \( x \in \mathbb{R} \).

**Details**

The cdf and quantile functions are omitted as no closed form analytic expressions could be found, decorate with FunctionImputation for numeric results.

**Super classes**

`distr6::Distribution -> distr6::Kernel -> Sigmoid`

**Public fields**

- `name` Full name of distribution.
- `short_name` Short name of distribution for printing.
- `description` Brief description of the distribution.

**Methods**

**Public methods:**

- `Sigmoid$new()`
- `Sigmoid$pdfSquared2Norm()`
- `Sigmoid$variance()`
- `Sigmoid$clone()`
Method `new()`: Creates a new instance of this R6 class.

Usage:
```
Sigmoid$new(decorators = NULL)
```

Arguments:
- `decorators` (character())
  Decorators to add to the distribution during construction.

Method `pdfSquared2Norm()`: The squared 2-norm of the pdf is defined by

\[
\int_a^b (f_X(u))^2 \, du
\]

where X is the Distribution, \( f_X \) is its pdf and \( a, b \) are the distribution support limits.

Usage:
```
Sigmoid$pdfSquared2Norm(x = 0, upper = Inf)
```

Arguments:
- `x` (numeric(1))
  Amount to shift the result.
- `upper` (numeric(1))
  Upper limit of the integral.

Method `variance()`: The variance of a distribution is defined by the formula

\[
\text{var}_X = E[X^2] - E[X]^2
\]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
```
Sigmoid$variance(...)  # ... Unused.
```

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

Usage:
```
Sigmoid$clone(deep = FALSE)
```

Arguments:
- `deep` Whether to make a deep clone.

See Also

Other kernels: Cosine, Epanechnikov, LogisticKernel, NormalKernel, Quartic, Silverman, TriangularKernel, Tricube, Triweight, UniformKernel
### Description

Mathematical and statistical functions for the Silverman kernel defined by the pdf,

\[ f(x) = \exp\left(-\frac{|x|}{\sqrt{2}}\right)/2 \ast \sin\left(\frac{|x|}{\sqrt{2}} + \frac{\pi}{4}\right) \]

over the support \( x \in \mathbb{R} \).

### Details

The cdf and quantile functions are omitted as no closed form analytic expressions could be found, decorate with FunctionImputation for numeric results.

### Super classes

\texttt{distr6::Distribution} -> \texttt{distr6::Kernel} -> \texttt{Silverman}

### Public fields

- \texttt{name} Full name of distribution.
- \texttt{short.name} Short name of distribution for printing.
- \texttt{description} Brief description of the distribution.

### Methods

**Public methods:**

- \texttt{Silverman$new()}  
- \texttt{Silverman$pdfSquared2Norm()}  
- \texttt{Silverman$cdfSquared2Norm()}  
- \texttt{Silverman$variance()}  
- \texttt{Silverman$clone()}  

**Method new():** Creates a new instance of this \texttt{R6} class.

**Usage:**

\texttt{Silverman$new(decorators = NULL)}

**Arguments:**

- \texttt{decorators} (character())  
  
  Decorators to add to the distribution during construction.

**Method pdfSquared2Norm():** The squared 2-norm of the pdf is defined by

\[ \int_{a}^{b} (f_X(u))^2 du \]

where \( X \) is the Distribution, \( f_X \) is its pdf and \( a, b \) are the distribution support limits.
Usage:
Silverman$pdfSquared2Norm(x = 0, upper = Inf)

Arguments:
- x (numeric(1))
  - Amount to shift the result.
- upper (numeric(1))
  - Upper limit of the integral.

Method cdfSquared2Norm(): The squared 2-norm of the cdf is defined by

\[ \int_a^b (F_X(u))^2 du \]

where X is the Distribution, \( F_X \) is its pdf and \( a, b \) are the distribution support limits.

Usage:
Silverman$cdfSquared2Norm(x = 0, upper = 0)

Arguments:
- x (numeric(1))
  - Amount to shift the result.
- upper (numeric(1))
  - Upper limit of the integral.

Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
Silverman$variance(...)

Arguments:
... Unused.

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

Usage:
Silverman$clone(deep = FALSE)

Arguments:
- deep Whether to make a deep clone.

See Also

Other kernels: Cosine, Epanechnikov, LogisticKernel, NormalKernel, Quartic, Sigmoid, TriangularKernel, Tricube, Triweight, UniformKernel
simulateEmpiricalDistribution

*Sample Empirical Distribution Without Replacement*

**Description**

Function to sample Empirical Distributions without replacement, as opposed to the `rand` method which samples with replacement.

**Usage**

```r
simulateEmpiricalDistribution(EmpiricalDist, n, seed = NULL)
```

**Arguments**

- **EmpiricalDist**: Empirical Distribution
- **n**: Number of samples to generate. See Details.
- **seed**: Numeric passed to `set.seed`. See Details.

**Details**

This function can only be used to sample from the Empirical distribution without replacement, and will return an error for other distributions.

The `seed` param ensures that the same samples can be reproduced and is more convenient than using the `set.seed()` function each time before use. If `set.seed` is `NULL` then the seed is left unchanged (NULL is not passed to the `set.seed` function).

If `n` is of length greater than one, then `n` is taken to be the length of `n`. If `n` is greater than the number of observations in the Empirical distribution, then `n` is taken to be the number of observations in the distribution.

**Value**

A vector of length `n` with elements drawn without replacement from the given Empirical distribution.

---

**skewType**

*Skewness Type*

**Description**

Gets the type of skewness

**Usage**

```r
skewType(skew)
```
**StudentT**

**Arguments**

skew numeric

**Details**

Skewness is a measure of asymmetry of a distribution.
A distribution can either have negative skew, no skew or positive skew. A symmetric distribution will always have no skew but the reverse relationship does not always hold.

**Value**

Returns one of 'negative skew', 'no skew' or 'positive skew'.

**Examples**

```r
skewType(1)
skewType(0)
skewType(-1)
```

---

**StudentT**  
*Student’s T Distribution Class*

**Description**

Mathematical and statistical functions for the Student’s T distribution, which is commonly used to estimate the mean of populations with unknown variance from a small sample size, as well as in t-testing for difference of means and regression analysis.

**Details**

The Student's T distribution parameterised with degrees of freedom, \( \nu \), is defined by the pdf,

\[
f(x) = \frac{\Gamma((\nu + 1)/2)/(\sqrt{\nu\pi})\Gamma(\nu/2)) \ast (1 + (x^2)/\nu)^{-(\nu+1)/2}}{\Gamma(\nu/2)}
\]

for \( \nu > 0 \).

**Value**

Returns an R6 object inheriting from class SDistribution.

**Distribution support**

The distribution is supported on the Reals.

**Default Parameterisation**

\( T(df = 1) \)
Omitted Methods

N/A

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> StudentT

Public fields

name  Full name of distribution.
short_name  Short name of distribution for printing.
description  Brief description of the distribution.
packages  Packages required to be installed in order to construct the distribution.

Methods

Public methods:

• StudentT$new()
• StudentT$mean()
• StudentT$mode()
• StudentT$variance()
• StudentT$skewness()
• StudentT$kurtosis()
• StudentT$entropy()
• StudentT$mgf()
• StudentT$cf()
• StudentT$pgf()
• StudentT$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
StudentT$new(df = NULL, decorators = NULL)

Arguments:
df (integer(1))
  Degrees of freedom of the distribution defined on the positive Reals.
decorators (character())
  Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) \cdot x \]

with an integration analogue for continuous distributions.
Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
StudentT$mode(which = "all")

Arguments:
which (character(1) | numeric(1))
Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula

$$\text{var}_X = E[X^2] - E[X]^2$$

where $E_X$ is the expectation of distribution $X$. If the distribution is multivariate the covariance matrix is returned.

Usage:
StudentT$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

$$sk_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution.

Usage:
StudentT$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

$$k_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^4$$

where $E_X$ is the expectation of distribution $X$, $\mu$ is the mean of the distribution and $\sigma$ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
StudentT$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
  ... Unused.

**Method entropy()**: The entropy of a (discrete) distribution is defined by

\[-\sum (f_X) \log(f_X)\]

where \(f_X\) is the pdf of distribution \(X\), with an integration analogue for continuous distributions.

**Usage**:

```r
StudentT$entropy(base = 2, ...)
```

**Arguments**:

- `base` (integer(1))
  - Base of the entropy logarithm, default = 2 (Shannon entropy)
  - ... Unused.

**Method mgf()**: The moment generating function is defined by

\[mgf_X(t) = E_X[exp(xt)]\]

where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

**Usage**:

```r
StudentT$mgf(t, ...)
```

**Arguments**:

- `t` (integer(1))
  - \(t\) integer to evaluate function at.
  - ... Unused.

**Method cf()**: The characteristic function is defined by

\[cf_X(t) = E_X[exp(xti)]\]

where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

**Usage**:

```r
StudentT$cf(t, ...)
```

**Arguments**:

- `t` (integer(1))
  - \(t\) integer to evaluate function at.
  - ... Unused.

**Method pgf()**: The probability generating function is defined by

\[pgf_X(z) = E_X[exp(z^x)]\]

where \(X\) is the distribution and \(E_X\) is the expectation of the distribution \(X\).

**Usage**:

```r
StudentT$pgf(z, ...)
```
StudentTNoncentral

Arguments:

- **z**: \( \text{integer}(1) \)
  - z integer to evaluate probability generating function at.
  - ... Unused.

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

Usage:

```r
StudentT$clone(deep = FALSE)
```

Arguments:

- **deep**: Whether to make a deep clone.

Author(s)

Chijing Zeng

References


See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

---

**StudentTNoncentral**  
*Noncentral Student’s T Distribution Class*

**Description**

Mathematical and statistical functions for the Noncentral Student’s T distribution, which is commonly used to estimate the mean of populations with unknown variance from a small sample size, as well as in t-testing for difference of means and regression analysis.
Details

The Noncentral Student’s T distribution parameterised with degrees of freedom, $\nu$ and location, $\lambda$, is defined by the pdf,

$$f(x) = \left(\frac{\nu^\nu}{\sqrt{\pi}} \exp\left(-\frac{\nu\lambda^2}{2(x^2+\nu)}\right)\right) \left(\frac{1}{\sqrt{\pi} \Gamma(\nu/2)^{2/(\nu-1)}} \right) \left(\nu \Gamma(\nu/2)^{1/2} \right) \int_0^\infty y^\nu \exp\left(-1/2(y-x\lambda/\sqrt{x^2+\nu})^2\right) \, dy$$

for $\nu > 0, \lambda \in \mathbb{R}$.

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on the Reals.

Default Parameterisation

TNS(df = 1, location = 0)

Omitted Methods

N/A

Also known as

N/A

Super classes

distr6::Distribution -> distr6::SDistribution -> StudentTNoncentral

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.

Methods

Public methods:

- StudentTNoncentral$new()
- StudentTNoncentral$mean()
- StudentTNoncentral$variance()
- StudentTNoncentral$clone()

Method $new()$: Creates a new instance of this R6 class.
Usage:
StudentTNoncentral$new(df = NULL, location = NULL, decorators = NULL)

Arguments:
df (integer(1))
   Degrees of freedom of the distribution defined on the positive Reals.
location (numeric(1))
   Location parameter, defined on the Reals.
decorators (character())
   Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) * x \]

with an integration analogue for continuous distributions.

Usage:
StudentTNoncentral$mean(...)

Arguments:
... Unused.

Method variance(): The variance of a distribution is defined by the formula

\[ var_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
StudentTNoncentral$variance(...)

Arguments:
... Unused.

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

Usage:
StudentTNoncentral$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

Author(s)
Jordan Deenichin

References
See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentT, Triangular, Uniform, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

testContinuous  
assert/check/test/Continuous

Description

Validation checks to test if Distribution is continuous.

Usage

testContinuous(
  object,
  errormsg = paste(object$short_name, "is not continuous")
)

cHECKContinuous(
  object,
  errormsg = paste(object$short_name, "is not continuous")
)

assertContinuous(
  object,
  errormsg = paste(object$short_name, "is not continuous")
)

Arguments

object Distribution
errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.
**Examples**

```r
testContinuous(Binomial$new()) # FALSE
```

---

**testDiscrete**  
`assert/check/test/Discrete`

**Description**

Validation checks to test if Distribution is discrete.

**Usage**

```r
testDiscrete(object, errormsg = paste(object$short_name, "is not discrete"))
checkDiscrete(object, errormsg = paste(object$short_name, "is not discrete"))
assertDiscrete(object, errormsg = paste(object$short_name, "is not discrete"))
```

**Arguments**

- `object`: Distribution
- `errormsg`: custom error message to return if `assert/check` fails

**Value**

If check passes then `assert` returns invisibly and `test/check` return `TRUE`. If check fails, `assert` stops code with error, `check` returns an error message as string, `test` returns `FALSE`.

**Examples**

```r
testDiscrete(Binomial$new()) # FALSE
```

---

**testDistribution**  
`assert/check/test/Distribution`

**Description**

Validation checks to test if a given object is a Distribution.
testDistributionList

Usage

testDistribution(
  object,
  errormsg = paste(object, "is not an R6 Distribution object")
)

checkDistribution(
  object,
  errormsg = paste(object, "is not an R6 Distribution object")
)

assertDistribution(
  object,
  errormsg = paste(object, "is not an R6 Distribution object")
)

Arguments

object object to test
errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testDistribution(5) # FALSE
testDistribution(Binomial$new()) # TRUE

testDistributionList(5) # FALSE
testDistributionList(Binomial$new()) # TRUE

testDistributionList "One or more items in the list are not Distributions"

Description

Validation checks to test if a given object is a list of Distributions.
assertDistributionList(
    object,
    errmsg = "One or more items in the list are not Distributions"
)

Arguments

object object to test
errmsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testDistributionList(list(Binomial$new(), 5)) # FALSE
testDistributionList(list(Binomial$new(), Exponential$new())) # TRUE
Arguments

object Distribution
errmsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testLeptokurtic(Binomial$new())

testMatrixvariate assert/check/test/Matrixvariate

Description

Validation checks to test if Distribution is matrixvariate.

Usage

testMatrixvariate(
  object,
  errmsg = paste(object$short_name, "is not matrixvariate")
)

checkMatrixvariate(
  object,
  errmsg = paste(object$short_name, "is not matrixvariate")
)

assertMatrixvariate(
  object,
  errmsg = paste(object$short_name, "is not matrixvariate")
)

Arguments

object Distribution
errmsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.
Examples

testMatrixvariate(Binomial$new()) # FALSE

description

Validation checks to test if Distribution is mesokurtic.

Usage

testMesokurtic(
    object,
    errormsg = paste(object$short_name, "is not mesokurtic")
)

checkMesokurtic(
    object,
    errormsg = paste(object$short_name, "is not mesokurtic")
)

assertMesokurtic(
    object,
    errormsg = paste(object$short_name, "is not mesokurtic")
)

Arguments

object Distribution
errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testMesokurtic(Binomial$new())
Description
Validation checks to test if Distribution is mixture.

Usage

```
testMixture(object, errormsg = paste(object$short_name, "is not mixture"))
checkMixture(object, errormsg = paste(object$short_name, "is not mixture"))
assertMixture(object, errormsg = paste(object$short_name, "is not mixture"))
```

Arguments

- `object` Distribution
- `errormsg` custom error message to return if assert/check fails

Value

If check passes then `assert` returns invisibly and `test/check` return `TRUE`. If check fails, `assert` stops code with error, `check` returns an error message as string, `test` returns `FALSE`.

Examples

```
testMixture(Binomial$new()) # FALSE
```

Description
Validation checks to test if Distribution is multivariate.

Usage

```
testMultivariate(  
    object,  
    errormsg = paste(object$short_name, "is not multivariate")  
  )  
```

```
checkMultivariate(  
    object,  
    errormsg = paste(object$short_name, "is not multivariate")  
  )  
```
testNegativeSkew

)

assertMultivariate(
object,
  errormsg = paste(object$short_name, "is not multivariate")
)

Arguments

object Distribution

errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testMultivariate(Binomial$new()) # FALSE

---

testNegativeSkew assert/check/test/NegativeSkew

Description

Validation checks to test if Distribution is negative skew.

Usage

testNegativeSkew(
  object,
  errormsg = paste(object$short_name, "is not negative skew")
)

checkNegativeSkew(
  object,
  errormsg = paste(object$short_name, "is not negative skew")
)

assertNegativeSkew(
  object,
  errormsg = paste(object$short_name, "is not negative skew")
)
Arguments

object Distribution
errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testNegativeSkew(Binomial$new())

testNoSkew

assert/check/test/NoSkew

Description

Validation checks to test if Distribution is no skew.

Usage

testNoSkew(object, errormsg = paste(object$short_name, "is not no skew"))

checkNoSkew(object, errormsg = paste(object$short_name, "is not no skew"))

assertNoSkew(object, errormsg = paste(object$short_name, "is not no skew"))

Arguments

object Distribution
errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testNoSkew(Binomial$new())
Description

Validation checks to test if a given object is a ParameterSet.

Usage

testParameterSet(
  object,
  errormsg = paste(object, "is not an R6 ParameterSet object")
)

checkParameterSet(
  object,
  errormsg = paste(object, "is not an R6 ParameterSet object")
)

assertParameterSet(
  object,
  errormsg = paste(object, "is not an R6 ParameterSet object")
)

Arguments

object object to test
errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testParameterSet(5) # FALSE
testParameterSet(Binomial$new()$parameters()) # TRUE
testParameterSetList  assert/check/test/ParameterSetList

Description

Validation checks to test if a given object is a list of ParameterSets.

Usage

testParameterSetList(
  object,
  errormsg = "One or more items in the list are not ParameterSets"
)

checkParameterSetList(
  object,
  errormsg = "One or more items in the list are not ParameterSets"
)

assertParameterSetList(
  object,
  errormsg = "One or more items in the list are not ParameterSets"
)

Arguments

object object to test
errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testParameterSetList(list(Binomial$new(), 5)) # FALSE
testParameterSetList(list(Binomial$new(), Exponential$new())) # TRUE
**Description**

Validation checks to test if Distribution is platykurtic.

**Usage**

```r
testPlatykurtic(
  object,
  errormsg = paste(object$short_name, "is not platykurtic")
)
```

```r
checkPlatykurtic(
  object,
  errormsg = paste(object$short_name, "is not platykurtic")
)
```

```r
assertPlatykurtic(
  object,
  errormsg = paste(object$short_name, "is not platykurtic")
)
```

**Arguments**

- `object` : Distribution
- `errormsg` : custom error message to return if `assert/check` fails

**Value**

If check passes then `assert` returns invisibly and `test/check` return `TRUE`. If check fails, `assert` stops code with error, `check` returns an error message as string, `test` returns `FALSE`.

**Examples**

```r
testPlatykurtic(Binomial$new())
```
Description

Validation checks to test if Distribution is positive skew.

Usage

testPositiveSkew(
  object,
  errormsg = paste(object$short_name, "is not positive skew")
)

checkPositiveSkew(
  object,
  errormsg = paste(object$short_name, "is not positive skew")
)

assertPositiveSkew(
  object,
  errormsg = paste(object$short_name, "is not positive skew")
)

Arguments

object Distribution
errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testPositiveSkew(Binomial$new())
testSymmetric

---

**Description**

Validation checks to test if Distribution is symmetric.

**Usage**

```r
testSymmetric(object, errormsg = paste(object$short_name, "is not symmetric"))
checkSymmetric(object, errormsg = paste(object$short_name, "is not symmetric"))
assertSymmetric(
  object,
  errormsg = paste(object$short_name, "is not symmetric")
)
```

**Arguments**

- `object` Distribution
- `errormsg` custom error message to return if assert/check fails

**Value**

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

**Examples**

```r
testSymmetric(Binomial$new())  # FALSE
```

---

testUnivariate  

---

**Description**

Validation checks to test if Distribution is univariate.
Usage

testUnivariate(
    object,
    errmsg = paste(object$short_name, "is not univariate")
)

checkUnivariate(
    object,
    errmsg = paste(object$short_name, "is not univariate")
)

assertUnivariate(
    object,
    errmsg = paste(object$short_name, "is not univariate")
)

Arguments

object Distribution
errmsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testUnivariate(Binomial$new()) # TRUE

Description

Mathematical and statistical functions for the Triangular distribution, which is commonly used to model population data where only the minimum, mode and maximum are known (or can be reliably estimated), also to model the sum of standard uniform distributions.

Details

The Triangular distribution parameterised with lower limit, \( a \), upper limit, \( b \), and mode, \( c \), is defined by the pdf,

\[
f(x) = \begin{cases} 
0, & x < a \\
2(x - a)/((b - a)(c - a)), & a \leq x < c \\
2/(b - a), & x = c
\end{cases}
\]
Triangular

\[ f(x) = \frac{2(b - x)}{(b - a)(b - c)}, \quad c < x \leq b \]
\[ f(x) = 0, \quad x > b \]
for \( a, b, c \in \mathbb{R}, \quad a \leq c \leq b \).

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on \([a, b]\).

Default Parameterisation

\( \text{Tri}(<lower \ = \ 0, \ upper \ = \ 1, \ mode \ = \ 0.5, \ symmetric \ = \ \text{FALSE}) \)

Omitted Methods

N/A

Also known as

N/A

Super classes

\( \text{distr6::Distribution} \rightarrow \text{distr6::SDistribution} \rightarrow \text{Triangular} \)

Public fields

- \text{name} Full name of distribution.
- \text{short_name} Short name of distribution for printing.
- \text{description} Brief description of the distribution.
- \text{packages} Packages required to be installed in order to construct the distribution.

Active bindings

- \text{properties} Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:

- \text{Triangular$new()}
- \text{Triangular$mean()}
- \text{Triangular$mode()}
- \text{Triangular$median()}
- \text{Triangular$variance()}
- \text{Triangular$skewness()}
- \text{Triangular$kurtosis()}
• Triangular$entropy()
• Triangular$mgf()
• Triangular$cf()
• Triangular$pgf()
• Triangular$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Triangular$new(
    lower = NULL,
    upper = NULL,
    mode = NULL,
    symmetric = NULL,
    decorators = NULL
)

Arguments:
lower (numeric(1))
    Lower limit of the Distribution, defined on the Reals.
upper (numeric(1))
    Upper limit of the Distribution, defined on the Reals.
mode (numeric(1))
    Mode of the distribution, if symmetric = TRUE then determined automatically.
symmetric (logical(1))
    If TRUE then the symmetric Triangular distribution is constructed, where the mode is automatically calculated. Otherwise mode can be set manually. Cannot be changed after construction.
decorators (character())
    Decorators to add to the distribution during construction.

Examples:
Triangular$new(lower = 2, upper = 5, symmetric = TRUE)
Triangular$new(lower = 2, upper = 5, mode = 4, symmetric = FALSE)

# You can view the type of Triangular distribution with $description
Triangular$new(symmetric = TRUE)$description
Triangular$new(symmetric = FALSE)$description

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

$$E_X(X) = \sum p_X(x) \times x$$

with an integration analogue for continuous distributions.

Usage:
Triangular$mean(...)

Arguments:
... Unused.
**Method** `mode()`: The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

*Usage:*

```
Triangular$mode(which = "all")
```

*Arguments:*

- `which` *(character(1) | numeric(1))*
  
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

**Method** `median()`: Returns the median of the distribution. If an analytical expression is available returns distribution median, otherwise if symmetric returns `self$mean`, otherwise returns `self$quantile(0.5)`.

*Usage:*

```
Triangular$median()
```

**Method** `variance()`: The variance of a distribution is defined by the formula

\[
\text{var}_X = E[X^2] - E[X]^2
\]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

*Usage:*

```
Triangular$variance(...)
```

*Arguments:*

- `...` unused.

**Method** `skewness()`: The skewness of a distribution is defined by the third standardised moment,

\[
sk_X = E_X\left[\frac{x - \mu}{\sigma}\right]^3
\]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

*Usage:*

```
Triangular$skewness(...)
```

*Arguments:*

- `...` unused.

**Method** `kurtosis()`: The kurtosis of a distribution is defined by the fourth standardised moment,

\[
k_X = E_X\left[\frac{x - \mu}{\sigma}\right]^4
\]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

*Usage:*

```
Triangular$kurtosis(excess = TRUE, ...)
```
Arguments:
excess (logical(1))
   If TRUE (default) excess kurtosis returned.
   ... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

$$- \sum (f_X) \log(f_X)$$

where $f_X$ is the pdf of distribution X, with an integration analogue for continuous distributions.

Usage:
Triangular$entropy(base = 2, ...)

Arguments:
base (integer(1))
   Base of the entropy logarithm, default = 2 (Shannon entropy)
   ... Unused.

Method mgf(): The moment generating function is defined by

$$mgf_X(t) = E_X[exp(xt)]$$

where X is the distribution and $E_X$ is the expectation of the distribution X.

Usage:
Triangular$mgf(t, ...)

Arguments:
t (integer(1))
   t integer to evaluate function at.
   ... Unused.

Method cf(): The characteristic function is defined by

$$cf_X(t) = E_X[exp(xti)]$$

where X is the distribution and $E_X$ is the expectation of the distribution X.

Usage:
Triangular$cf(t, ...)

Arguments:
t (integer(1))
   t integer to evaluate function at.
   ... Unused.

Method pgf(): The probability generating function is defined by

$$pgf_X(z) = E_X[exp(z^x)]$$

where X is the distribution and $E_X$ is the expectation of the distribution X.

Usage:
Triangular

\text{Triangular}\$pgf(z, \ldots)

\text{Arguments:}
\begin{itemize}
\item \textit{z} (integer(1))
\end{itemize}
\text{z \ integer to evaluate probability generating function at.}

\ldots \ \text{Unused.}

\textbf{Method \texttt{clone()}}: The objects of this class are cloneable with this method.

\textbf{Usage:}
\begin{itemize}
\item \texttt{Triangular}\$clone(deep = \texttt{FALSE})
\end{itemize}

\textbf{Arguments:}
\begin{itemize}
\item \texttt{deep} \ \texttt{Whether to make a deep clone.}
\end{itemize}

\textbf{References}


\textbf{See Also}

Other continuous distributions: \texttt{Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Uniform, Wald, Weibull}

Other univariate distributions: \texttt{Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Uniform, Wald, Weibull, WeightedDiscrete}

\textbf{Examples}

\begin{verbatim}
## Method \texttt{Triangular}\$new

Triangular\$new(lower = 2, upper = 5, symmetric = \texttt{TRUE})
Triangular\$new(lower = 2, upper = 5, mode = 4, symmetric = \texttt{FALSE})

# You can view the type of Triangular distribution with \$description
Triangular\$new(symmetric = \texttt{TRUE})\$description
Triangular\$new(symmetric = \texttt{FALSE})\$description
\end{verbatim}
Description

Mathematical and statistical functions for the Triangular kernel defined by the pdf,

\[ f(x) = 1 - |x| \]

over the support \( x \in (-1, 1) \).

Super classes

\texttt{distr6::Distribution -> distr6::Kernel -> TriangularKernel}

Public fields

- name: Full name of distribution.
- short_name: Short name of distribution for printing.
- description: Brief description of the distribution.

Methods

Public methods:

- \texttt{TriangularKernel$pdfSquared2Norm()}
- \texttt{TriangularKernel$cdfSquared2Norm()}
- \texttt{TriangularKernel$variance()}
- \texttt{TriangularKernel$clone()}

Method pdfSquared2Norm(): The squared 2-norm of the pdf is defined by

\[ \int_a^b (f_X(u))^2 \, du \]

where \( X \) is the Distribution, \( f_X \) is its pdf and \( a, b \) are the distribution support limits.

Usage:

\texttt{TriangularKernel$pdfSquared2Norm(x = 0, upper = Inf)}

Arguments:

- \texttt{x (numeric(1))}
  - Amount to shift the result.
- \texttt{upper (numeric(1))}
  - Upper limit of the integral.

Method cdfSquared2Norm(): The squared 2-norm of the cdf is defined by

\[ \int_a^b (F_X(u))^2 \, du \]

where \( X \) is the Distribution, \( F_X \) is its pdf and \( a, b \) are the distribution support limits.
Usage:
TriangularKernel$cdfSquared2Norm(x = 0, upper = 0)

Arguments:
x (numeric(1))
   Amount to shift the result.
upper (numeric(1))
   Upper limit of the integral.

Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
TriangularKernel$variance(...)

Arguments:
... Unused.

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

Usage:
TriangularKernel$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also
Other kernels: Cosine, Epanechnikov, LogisticKernel, NormalKernel, Quartic, Sigmoid, Silverman, Tricube, Triweight, UniformKernel

Tricube

<table>
<thead>
<tr>
<th>Tricube Kernel</th>
</tr>
</thead>
</table>

Description
Mathematical and statistical functions for the Tricube kernel defined by the pdf,

\[ f(x) = \frac{70}{81}(1 - |x|^3)^3 \]

over the support \( x \in (-1, 1) \).

Details
The quantile function is omitted as no closed form analytic expressions could be found, decorate with FunctionImputation for numeric results.
Super classes

distr6::Distribution -> distr6::Kernel -> Tricube

Public fields

name  Full name of distribution.
short_name  Short name of distribution for printing.
description  Brief description of the distribution.

Methods

Public methods:

- Tricube$pdfSquared2Norm()
- Tricube$cdfSquared2Norm()
- Tricube$variance()
- Tricube$clone()

Method pdfSquared2Norm(): The squared 2-norm of the pdf is defined by

\[ \int_a^b (f_X(u))^2 \, du \]

where X is the Distribution, \( f_X \) is its pdf and \( a, b \) are the distribution support limits.

Usage:

Tricube$pdfSquared2Norm(x = 0, upper = Inf)

Arguments:

x (numeric(1))  Amount to shift the result.
upper (numeric(1))  Upper limit of the integral.

Method cdfSquared2Norm(): The squared 2-norm of the cdf is defined by

\[ \int_a^b (F_X(u))^2 \, du \]

where X is the Distribution, \( F_X \) is its pdf and \( a, b \) are the distribution support limits.

Usage:

Tricube$cdfSquared2Norm(x = 0, upper = 0)

Arguments:

x (numeric(1))  Amount to shift the result.
upper (numeric(1))  Upper limit of the integral.
Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
```
Tricube$variance(...)  
```

Arguments:
... Unused.

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

Usage:
```
Tricube$clone(deep = FALSE)  
```

Arguments:

\[ \text{deep} \] Whether to make a deep clone.

See Also

Other kernels: Cosine, Epanechnikov, LogisticKernel, NormalKernel, Quartic, Sigmoid, Silverman, TriangularKernel, Triweight, UniformKernel

---

**Triweight**

**Triweight Kernel**

**Description**

Mathematical and statistical functions for the Triweight kernel defined by the pdf,

\[ f(x) = \frac{35}{32}(1 - x^2)^3 \]

over the support \( x \in (-1, 1) \).

**Details**

The quantile function is omitted as no closed form analytic expression could be found, decorate with FunctionImputation for numeric results.

**Super classes**

```
\text{distr6::Distribution} -> \text{distr6::Kernel} -> \text{Triweight}  
```

**Public fields**

- name Full name of distribution.
- short_name Short name of distribution for printing.
- description Brief description of the distribution.
Methods

Public methods:

- Triweight$pdfSquared2Norm()
- Triweight$cdfSquared2Norm()
- Triweight$variance()
- Triweight$clone()

Method pdfSquared2Norm(): The squared 2-norm of the pdf is defined by

$$\int_a^b (f_X(u))^2 du$$

where X is the Distribution, $f_X$ is its pdf and a, b are the distribution support limits.

Usage:
Triweight$pdfSquared2Norm(x = 0, upper = Inf)

Arguments:
- x (numeric(1))
  - Amount to shift the result.
- upper (numeric(1))
  - Upper limit of the integral.

Method cdfSquared2Norm(): The squared 2-norm of the cdf is defined by

$$\int_a^b (F_X(u))^2 du$$

where X is the Distribution, $F_X$ is its pdf and a, b are the distribution support limits.

Usage:
Triweight$cdfSquared2Norm(x = 0, upper = 0)

Arguments:
- x (numeric(1))
  - Amount to shift the result.
- upper (numeric(1))
  - Upper limit of the integral.

Method variance(): The variance of a distribution is defined by the formula

$$\text{var}_X = E[X^2] - E[X]^2$$

where $E_X$ is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
Triweight$variance(...)

Arguments:
... Unused.
Method clone(): The objects of this class are cloneable with this method.

Usage:
Triweight$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also
Other kernels: Cosine, Epanechnikov, LogisticKernel, NormalKernel, Quartic, Sigmoid, Silverman, TriangularKernel, Tricube, UniformKernel

---

truncatemethod

Truncate a Distribution

Description
S3 functionality to truncate an R6 distribution.

Usage
truncate(x, lower = NULL, upper = NULL)

Arguments
x Distribution.
lower lower limit for truncation.
upper upper limit for truncation.

See Also
TruncatedDistribution

---

TruncatedDistribution

Distribution Truncation Wrapper

Description
A wrapper for truncating any probability distribution at given limits.
Details

The pdf and cdf of the distribution are required for this wrapper, if unavailable decorate with FunctionImputation first.

Truncates a distribution at lower and upper limits on a left-open interval, using the formulae

\[ f_T(x) = \frac{f_X(x)}{F_X(upper) - F_X(lower)} \]

\[ F_T(x) = \frac{F_X(x) - F_X(lower)}{F_X(upper) - F_X(lower)} \]

where \( f_T/F_T \) is the pdf/cdf of the truncated distribution \( T = \text{Truncate}(X, \text{lower}, \text{upper}) \) and \( f_X/F_X \) is the pdf/cdf of the original distribution. \( T \) is supported on \((-\infty, \infty)\).

Super classes

distr6::Distribution -> distr6::DistributionWrapper -> TruncatedDistribution

Active bindings

properties Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:

- TruncatedDistribution$new()
- TruncatedDistribution$clone()

Method new(): Creates a new instance of this R6 class.

Usage:

TruncatedDistribution$new(distribution, lower = NULL, upper = NULL)

Arguments:

distribution ([Distribution])

  Distribution to wrap.
lower (numeric(1))

  Lower limit to huberize the distribution at. If NULL then the lower bound of the Distribution is used.
upper (numeric(1))

  Upper limit to huberize the distribution at. If NULL then the upper bound of the Distribution is used.

Examples:

TruncatedDistribution$new(
  Binomial$new(prob = 0.5, size = 10),
  lower = 2, upper = 4
)

# alternate constructor
truncated(Binomial$new(), lower = 2, upper = 4)
Method `clone()`: The objects of this class are cloneable with this method.

Usage:
```r
TruncatedDistribution$clone(deep = FALSE)
```

Arguments:
- `deep` Whether to make a deep clone.

See Also

Other wrappers: Convolution, DistributionWrapper, HuberizedDistribution, MixtureDistribution, ProductDistribution, VectorDistribution

Examples

```r
## Method \textasciitilde\texttt{TruncatedDistribution}\$\texttt{new}
## ------------------------------------------
TruncatedDistribution$new(
  Binomial$new(prob = 0.5, size = 10),
  lower = 2, upper = 4
)

# alternate constructor
truncate(Binomial$new(), lower = 2, upper = 4)
```

Uniform Distribution Class

Description

Mathematical and statistical functions for the Uniform distribution, which is commonly used to model continuous events occurring with equal probability, as an uninformed prior in Bayesian modelling, and for inverse transform sampling.

Details

The Uniform distribution parameterised with lower, \(a\), and upper, \(b\), limits is defined by the pdf,

\[
f(x) = \frac{1}{b - a}
\]

for \(-\infty < a < b < \infty\).

Value

Returns an R6 object inheriting from class SDistribution.
Distribution support
The distribution is supported on \([a, b]\).

Default Parameterisation
Unif(lower = 0, upper = 1)

Omitted Methods
N/A

Also known as
N/A

Super classes
```
distr6::Distribution -> distr6::SDistribution -> Uniform
```

Public fields
```
name  Full name of distribution.
short_name  Short name of distribution for printing.
description  Brief description of the distribution.
packages  Packages required to be installed in order to construct the distribution.
```

Active bindings
```
properties  Returns distribution properties, including skewness type and symmetry.
```

Methods

Public methods:
```
• Uniform$new()
• Uniform$mean()
• Uniform$mode()
• Uniform$variance()
• Uniform$skewness()
• Uniform$kurtosis()
• Uniform$entropy()
• Uniform$mgf()
• Uniform$cf()
• Uniform$pgf()
• Uniform$clone()
```

Method new(): Creates a new instance of this R6 class.

Usage:
Uniform$new(lower = NULL, upper = NULL, decorators = NULL)

Arguments:
lower (numeric(1))
  Lower limit of the Distribution, defined on the Reals.
upper (numeric(1))
  Upper limit of the Distribution, defined on the Reals.
decorators (character())
  Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) * x \]

with an integration analogue for continuous distributions.

Usage:
Uniform$mean(...)

Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
Uniform$mode(which = "all")

Arguments:
which (character(1) | numeric(1)
  Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula

\[ var_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance matrix is returned.

Usage:
Uniform$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,

\[ sk_X = E_X\left[ \frac{x - \mu^3}{\sigma} \right] \]

where \( E_X \) is the expectation of distribution X, \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.
Usage:
Uniform$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left[ \frac{X - \mu}{\sigma}^4 \right] \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Uniform$kurtosis(excess = TRUE, ...)

Arguments:
excess (logical(1))
   If TRUE (default) excess kurtosis returned.
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

\[ -\sum (f_X) \log(f_X) \]

where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions.

Usage:
Uniform$entropy(base = 2, ...)

Arguments:
base (integer(1))
   Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method mgf(): The moment generating function is defined by

\[ mgf_X(t) = E_X[exp(tX)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:
Uniform$mgf(t, ...)

Arguments:
t (integer(1))
   t integer to evaluate function at.
... Unused.

Method cf(): The characteristic function is defined by

\[ cf_X(t) = E_X[exp(xti)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).
Usage:
Uniform$cf(t, ...)

Arguments:
t (integer(1))
    t integer to evaluate function at.

Method pgf(): The probability generating function is defined by

$$pgf_X(z) = E_X[exp(z^t)]$$

where X is the distribution and $E_X$ is the expectation of the distribution X.

Usage:
Uniform$pgf(z, ...)

Arguments:
z (integer(1))
    z integer to evaluate probability generating function at.

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

Usage:
Uniform$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

Author(s)
Yumi Zhou

References
Michael P. McLaughlin.

See Also
Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Wald, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Wald, Weibull, WeightedDiscrete
Description

Mathematical and statistical functions for the Uniform kernel defined by the pdf,

\[ f(x) = \frac{1}{2} \]

over the support \( x \in (-1, 1) \).

Super classes

\texttt{distr6::Distribution -> distr6::Kernel -> UniformKernel}

Public fields

- \texttt{name}  Full name of distribution.
- \texttt{short_name}  Short name of distribution for printing.
- \texttt{description}  Brief description of the distribution.

Methods

Public methods:

- \texttt{UniformKernel$pdfSquared2Norm()}
- \texttt{UniformKernel$cdfSquared2Norm()}
- \texttt{UniformKernel$variance()}
- \texttt{UniformKernel$clone()}

Method \texttt{pdfSquared2Norm()}: The squared 2-norm of the pdf is defined by

\[ \int_a^b (f_X(u))^2 du \]

where \( X \) is the Distribution, \( f_X \) is its pdf and \( a, b \) are the distribution support limits.

Usage:

\texttt{UniformKernel$pdfSquared2Norm(x = 0, upper = Inf)}

Arguments:

- \texttt{x} (\texttt{numeric(1)})
  Amount to shift the result.
- \texttt{upper} (\texttt{numeric(1)})
  Upper limit of the integral.

Method \texttt{cdfSquared2Norm()}: The squared 2-norm of the cdf is defined by

\[ \int_a^b (F_X(u))^2 du \]

where \( X \) is the Distribution, \( F_X \) is its pdf and \( a, b \) are the distribution support limits.
Usage:
UniformKernel$cdfSquared2Norm(x = 0, upper = 0)

Arguments:
x (numeric(1))
  Amount to shift the result.
upper (numeric(1))
  Upper limit of the integral.

Method variance(): The variance of a distribution is defined by the formula

\[ \text{var}_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:
UniformKernel$variance(...) 

Arguments:
... Unused.

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

Usage:
UniformKernel$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also

Other kernels: Cosine, Epanechnikov, LogisticKernel, NormalKernel, Quartic, Sigmoid, Silverman, TriangularKernel, Tricube, Triweight

---

VectorDistribution  Vectorise Distributions

Description

A wrapper for creating a vector of distributions.

Details

A vector distribution is intened to vectorize distributions more efficiently than storing a list of distributions. To improve speed and reduce memory usage, distributions are only constructed when methods (e.g. d/p/q/r) are called.

Super classes

distr6::Distribution -> distr6::DistributionWrapper -> VectorDistribution
Active bindings

- `modelTable` Returns reference table of wrapped `Distributions`.
- `distlist` Returns list of constructed wrapped `Distributions`.
- `ids` Returns ids of constructed wrapped `Distributions`.

Methods

**Public methods:**

- `VectorDistribution$new()`
- `VectorDistribution$getParameterValue()`
- `VectorDistribution$wrappedModels()`
- `VectorDistribution$strprint()`
- `VectorDistribution$mean()`
- `VectorDistribution$mode()`
- `VectorDistribution$median()`
- `VectorDistribution$variance()`
- `VectorDistribution$skewness()`
- `VectorDistribution$kurtosis()`
- `VectorDistribution$entropy()`
- `VectorDistribution$mgf()`
- `VectorDistribution$cf()`
- `VectorDistribution$pgf()`
- `VectorDistribution$pdf()`
- `VectorDistribution$cdf()`
- `VectorDistribution$quantile()`
- `VectorDistribution$rand()`
- `VectorDistribution$clone()`

**Method** `new()`: Creates a new instance of this `R6` class.

*Usage:*

```r
VectorDistribution$new(
  distlist = NULL,
  distribution = NULL,
  params = NULL,
  shared_params = NULL,
  name = NULL,
  short_name = NULL,
  decorators = NULL,
  vecdist = NULL,
  ids = NULL,
  ...
)
```

*Arguments:*
**VectorDistribution**

- distlist (list())
  - List of Distributions.

- distribution (character(1))
  - Should be supplied with params and optionally shared_params as an alternative to distlist.
  - Much faster implementation when only one class of distribution is being wrapped. distribution is the full name of one of the distributions in listDistributions(), or "Distribution" if constructing custom distributions. See examples in VectorDistribution.

- params (list()|data.frame())
  - Parameters in the individual distributions for use with distribution. Can be supplied as a list, where each element is the list of parameters to set in the distribution, or as an object coercable to data.frame, where each column is a parameter and each row is a distribution.
  - See examples in VectorDistribution.

- shared_params (list())
  - If any parameters are shared when using the distribution constructor, this provides a much faster implementation to list and query them together. See examples in VectorDistribution.

- name (character(1))
  - Optional name of wrapped distribution.

- short_name (character(1))
  - Optional short name/ID of wrapped distribution.

- decorators (character())
  - Decorators to add to the distribution during construction.

**vecdist** VectorDistribution

- Alternative constructor to directly create this object from an object inheriting from VectorDistribution.

- ids (character())
  - Optional ids for wrapped distributions in vector, should be unique and of same length as the number of distributions.

... Unused

**Examples:**

```r

\dontrun{
VectorDistribution$new(
  distribution = "Binomial",
  params = list(
    list(prob = 0.1, size = 2),
    list(prob = 0.6, size = 4),
    list(prob = 0.2, size = 6)
  )
)
}

VectorDistribution$new(
  distribution = "Binomial",
  params = data.table::data.table(prob = c(0.1, 0.6, 0.2), size = c(2, 4, 6))
)

# Alternatively
VectorDistribution$new(

```
list(
  Binomial$new(prob = 0.1, size = 2),
  Binomial$new(prob = 0.6, size = 4),
  Binomial$new(prob = 0.2, size = 6)
)

\textbf{Method} \texttt{getParameterValue}(): Returns the value of the supplied parameter.

\textit{Usage:}
\texttt{VectorDistribution$getParameterValue(id, \ldots)}

\textit{Arguments:}
id character()
  id of parameter value to return.
\ldots Unused

\textbf{Method} \texttt{wrappedModels}(): Returns model(s) wrapped by this wrapper.

\textit{Usage:}
\texttt{VectorDistribution$wrappedModels(model = NULL)}

\textit{Arguments:}
model (character(1))
  id of wrapped Distributions to return. If \texttt{NULL} (default), a list of all wrapped Distributions is returned; if only one Distribution is matched then this is returned, otherwise a list of Distributions.

\textbf{Method} \texttt{strprint}(): Printable string representation of the VectorDistribution. Primarily used internally.

\textit{Usage:}
\texttt{VectorDistribution$strprint(n = 10)}

\textit{Arguments:}
n (integer(1))
  Number of distributions to include when printing.

\textbf{Method} \texttt{mean}(): Returns named vector of means from each wrapped Distribution.

\textit{Usage:}
\texttt{VectorDistribution$mean(\ldots)}

\textit{Arguments:}
\ldots Passed to \texttt{CoreStatistics$genExp} if numeric.

\textbf{Method} \texttt{mode}(): Returns named vector of modes from each wrapped Distribution.

\textit{Usage:}
\texttt{VectorDistribution$mode(which = "all")}

\textit{Arguments:}
which (character(1) | numeric(1))
  Ignored if distribution is unimodal. Otherwise “all” returns all modes, otherwise specifies which mode to return.

**Method** `median()`: Returns named vector of medians from each wrapped `Distribution`.

*Usage:*

```
VectorDistribution$median()
```

**Method** `variance()`: Returns named vector of variances from each wrapped `Distribution`.

*Usage:*

```
VectorDistribution$variance(...)
```

*Arguments:*

... Passed to `CoreStatistics$genExp` if numeric.

**Method** `skewness()`: Returns named vector of skewness from each wrapped `Distribution`.

*Usage:*

```
VectorDistribution$skewness(...)
```

*Arguments:*

... Passed to `CoreStatistics$genExp` if numeric.

**Method** `kurtosis()`: Returns named vector of kurtosis from each wrapped `Distribution`.

*Usage:*

```
VectorDistribution$kurtosis(excess = TRUE, ...)
```

*Arguments:*

excess (logical(1))
  If TRUE (default) excess kurtosis returned.
... Passed to `CoreStatistics$genExp` if numeric.

**Method** `entropy()`: Returns named vector of entropy from each wrapped `Distribution`.

*Usage:*

```
VectorDistribution$entropy(base = 2, ...)
```

*Arguments:*

base (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)
... Passed to `CoreStatistics$genExp` if numeric.

**Method** `mgf()`: Returns named vector of mgf from each wrapped `Distribution`.

*Usage:*

```
VectorDistribution$mgf(t, ...)
```

*Arguments:*

t (integer(1))
  t integer to evaluate function at.
... Passed to `CoreStatistics$genExp` if numeric.
Method `cf()`: Returns named vector of cf from each wrapped Distribution.

Usage:
```
VectorDistribution$cf(t, ...)
```

Arguments:
- `t` (integer(1))
  - `t` integer to evaluate function at.
- `...` Passed to `CoreStatistics$genExp` if numeric.

Method `pgf()`: Returns named vector of pgf from each wrapped Distribution.

Usage:
```
VectorDistribution$pgf(z, ...)
```

Arguments:
- `z` (integer(1))
  - `z` integer to evaluate probability generating function at.
- `...` Passed to `CoreStatistics$genExp` if numeric.

Method `pdf()`: Returns named vector of pdfs from each wrapped Distribution.

Usage:
```
VectorDistribution$pdf(..., log = FALSE, simplify = TRUE, data = NULL)
```

Arguments:
- `...` (numeric())
  - Points to evaluate the function at Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.
- `log` (logical(1))
  - If TRUE returns the logarithm of the probabilities. Default is FALSE.
- `simplify` logical(1)
  - If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a `data.table::data.table`.
- `data` array
  - Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

Examples:
```
v <- VectorDistribution$new(
    distribution = "Binomial",
    params = data.frame(size = 9:10, prob = c(0.5,0.6)))
```
```
v$pdf(2)
  # Equivalently
v$pdf(2, 2)
```
```
v$pdf(1:2, 3:4)
  # or as a matrix
```
vd$pdf(data = matrix(1:4, nrow = 2))

# when wrapping multivariate distributions, arrays are required
vd <- VectorDistribution$new(
  distribution = "Multinomial",
  params = list(
    list(size = 5, probs = c(0.1, 0.9)),
    list(size = 8, probs = c(0.3, 0.7))
  )
)

# evaluates Multinom1 and Multinom2 at (1, 4)
vd$pdf(1, 4)

# evaluates Multinom1 at (1, 4) and Multinom2 at (5, 3)
vd$pdf(data = array(c(1,4,5,3), dim = c(1,2,2)))

# and the same across many samples
vd$pdf(data = array(c(1,2,4,3,5,1,3,7), dim = c(2,2,2)))

**Method cdf()**: Returns named vector of cdfs from each wrapped Distribution. Same usage as $pdf.

**Usage**: VectorDistribution$cdf(
  ..., 
  lower.tail = TRUE, 
  log.p = FALSE, 
  simplify = TRUE, 
  data = NULL 
)

**Arguments**:

... (numeric())
  Points to evaluate the function at Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.

lower.tail (logical(1))
  If TRUE (default), probabilities are $X \leq x$, otherwise, $P(X > x)$.

log.p (logical(1))
  If TRUE returns the logarithm of the probabilities. Default is FALSE.

simplify logical(1)
  If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.

data array
  Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.
Method quantile(): Returns named vector of quantiles from each wrapped Distribution. Same usage as $cdf.

Usage:
VectorDistribution$quantile(
    ...,
    lower.tail = TRUE,
    log.p = FALSE,
    simplify = TRUE,
    data = NULL
)

Arguments:
... (numeric())
Points to evaluate the function at Arguments do not need to be named. The length of each argument corresponds to the number of points to evaluate, the number of arguments corresponds to the number of variables in the distribution. See examples.
lower.tail (logical(1))
If TRUE (default), probabilities are $X \leq x$, otherwise, $P(X > x)$.
log.p (logical(1))
If TRUE returns the logarithm of the probabilities. Default is FALSE.
simplify logical(1)
If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.
data array
Alternative method to specify points to evaluate. If univariate then rows correspond with number of points to evaluate and columns correspond with number of variables to evaluate. In the special case of VectorDistributions of multivariate distributions, then the third dimension corresponds to the distribution in the vector to evaluate.

Method rand(): Returns data.table::data.table of draws from each wrapped Distribution.

Usage:
VectorDistribution$rand(n, simplify = TRUE)

Arguments:
n (numeric(1))
Number of points to simulate from the distribution. If length greater than 1, then n <- length(n),
simplify logical(1)
If TRUE (default) simplifies the return if possible to a numeric, otherwise returns a data.table::data.table.

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

Usage:
VectorDistribution$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also
Other wrappers: Convolution, DistributionWrapper, HuberizedDistribution, MixtureDistribution, ProductDistribution, TruncatedDistribution
## Examples

```r
## Method `VectorDistribution$new`

## Not run:
VectorDistribution$new(
  distribution = "Binomial",
  params = list(
    list(prob = 0.1, size = 2),
    list(prob = 0.6, size = 4),
    list(prob = 0.2, size = 6)
  )
)

VectorDistribution$new(
  distribution = "Binomial",
  params = data.table::data.table(prob = c(0.1, 0.6, 0.2), size = c(2, 4, 6))
)

# Alternatively
VectorDistribution$new(
  list(
    Binomial$new(prob = 0.1, size = 2),
    Binomial$new(prob = 0.6, size = 4),
    Binomial$new(prob = 0.2, size = 6)
  )
)

## End(Not run)

## Method `VectorDistribution$pdf`

vd <- VectorDistribution$new(
  distribution = "Binomial",
  params = data.frame(size = 9:10, prob = c(0.5, 0.6)))

vd$pdf(2)
# Equivalently
vd$pdf(2, 2)

vd$pdf(1:2, 3:4)
# or as a matrix
vd$pdf(data = matrix(1:4, nrow = 2))

# when wrapping multivariate distributions, arrays are required
vd <- VectorDistribution$new(
  distribution = "Multinomial",
  params = list(
    list(prob = c(0.1, 0.6, 0.2), size = c(2, 4, 6)))
)
list(size = 5, probs = c(0.1, 0.9)),
list(size = 8, probs = c(0.3, 0.7))
)

# evaluates Multinom1 and Multinom2 at (1, 4)
vd$pdf(1, 4)

# evaluates Multinom1 at (1, 4) and Multinom2 at (5, 3)
vd$pdf(data = array(c(1,4,5,3), dim = c(1,2,2)))

# and the same across many samples
vd$pdf(data = array(c(1,2,4,3,5,1,3,7), dim = c(2,2,2)))

---

**Wald**

**Wald Distribution Class**

### Description

Mathematical and statistical functions for the Wald distribution, which is commonly used for modelling the first passage time for Brownian motion.

### Details

The Wald distribution parameterised with mean, \( \mu \), and shape, \( \lambda \), is defined by the pdf,

\[
f(x) = (\lambda/(2x^3\pi))^{1/2} \exp((-\lambda(x - \mu)^2)/(2\mu^2x))
\]

for \( \lambda > 0 \) and \( \mu > 0 \).

Sampling is performed as per Michael, Schucany, Haas (1976).

### Value

Returns an R6 object inheriting from class SDistribution.

### Distribution support

The distribution is supported on the Positive Reals.

### Default Parameterisation

Wald(mean = 1, shape = 1)

### Omitted Methods

quantile is omitted as no closed form analytic expression could be found, decorate with FunctionImputation for a numerical imputation.
Also known as

Also known as the Inverse Normal distribution.

Super classes

distr6::Distribution -> distr6::SDistribution -> Wald

Public fields

name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.
packages Packages required to be installed in order to construct the distribution.

Methods

Public methods:

• Wald$new()
• Wald$mean()
• Wald$mode()
• Wald$variance()
• Wald$skewness()
• Wald$kurtosis()
• Wald$mgf()
• Wald$cf()
• Wald$pgf()
• Wald$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
Wald$new(mean = NULL, shape = NULL, decorators = NULL)

Arguments:
mean (numeric(1))
  Mean of the distribution, location parameter, defined on the positive Reals.
shape (numeric(1))
  Shape parameter, defined on the positive Reals.
decorators (character())
  Decorators to add to the distribution during construction.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) * x \]

with an integration analogue for continuous distributions.

Usage:
Wald$mean(...)  
Arguments:  
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:  
Wald$mode(which = "all")  
Arguments:  
which (character(1) | numeric(1))  
Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

Method variance(): The variance of a distribution is defined by the formula
\[ \text{var}_X = E[X^2] - E[X]^2 \]
where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

Usage:  
Wald$variance(...)  
Arguments:  
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised moment,
\[ \text{sk}_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3 \]
where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

Usage:  
Wald$skewness(...)  
Arguments:  
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,
\[ k_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^4 \]
where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:  
Wald$kurtosis(excess = TRUE, ...)  
Arguments:
excess (logical(1))
  If TRUE (default) excess kurtosis returned.
... Unused.

**Method mgf()**: The moment generating function is defined by

\[ mgf_X(t) = E_X[exp(zt)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

*Usage:*
Wald$mgf(t, ...)

*Arguments:*
- \( t \) (integer(1))
  - \( t \) integer to evaluate function at.
... Unused.

**Method cf()**: The characteristic function is defined by

\[ cf_X(t) = E_X[exp(tzi)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

*Usage:*
Wald$cf(t, ...)

*Arguments:*
- \( t \) (integer(1))
  - \( t \) integer to evaluate function at.
... Unused.

**Method pgf()**: The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(zt)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

*Usage:*
Wald$pgf(z, ...)

*Arguments:*
- \( z \) (integer(1))
  - \( z \) integer to evaluate probability generating function at.
... Unused.

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

*Usage:*
Wald$clone(deep = FALSE)

*Arguments:*
- \( deep \) Whether to make a deep clone.
References


See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Weibull

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Weibull, WeightedDiscrete

---

**Weibull**

**Weibull Distribution Class**

**Description**

Mathematical and statistical functions for the Weibull distribution, which is commonly used in survival analysis as it satisfies both PH and AFT requirements.

**Details**

The Weibull distribution parameterised with shape, $\alpha$, and scale, $\beta$, is defined by the pdf,

$$ f(x) = \frac{\alpha}{\beta}(x/\beta)^{\alpha-1}exp(-x/\beta)^\alpha $$

for $\alpha, \beta > 0$.

**Value**

Returns an R6 object inheriting from class SDistribution.

**Distribution support**

The distribution is supported on the Positive Reals.

**Default Parameterisation**

Weibull(shape = 1, scale = 1)
Omitted Methods
N/A

Also known as
N/A

Super classes
\texttt{distr6::Distribution} -> \texttt{distr6::SDistribution} -> \texttt{Weibull}

Public fields

- \texttt{name} Full name of distribution.
- \texttt{short\_name} Short name of distribution for printing.
- \texttt{description} Brief description of the distribution.
- \texttt{packages} Packages required to be installed in order to construct the distribution.

Methods

Public methods:

- \texttt{Weibull$new()}
- \texttt{Weibull$mean()}
- \texttt{Weibull$mode()}
- \texttt{Weibull$median()}
- \texttt{Weibull$variance()}
- \texttt{Weibull$skewness()}
- \texttt{Weibull$kurtosis()}
- \texttt{Weibull$entropy()}
- \texttt{Weibull$pgf()}
- \texttt{Weibull$clone()}

Method \texttt{new()}: Creates a new instance of this \texttt{R6} class.

Usage:
\texttt{Weibull$new(shape = NULL, scale = NULL, altscale = NULL, decorators = NULL)}

Arguments:

- \texttt{shape} (\texttt{numeric(1)})
  Shape parameter, defined on the positive Reals.
- \texttt{scale} (\texttt{numeric(1)})
  Scale parameter, defined on the positive Reals.
- \texttt{altscale} (\texttt{numeric(1)})
  Alternative scale parameter, if given then scale is ignored. \texttt{altscale = scale^shape}.
- \texttt{decorators} (\texttt{character()})
  Decorators to add to the distribution during construction.
**Method** mean(): The arithmetic mean of a (discrete) probability distribution \( X \) is the expectation

\[
E_X(X) = \sum p_X(x) * x
\]

with an integration analogue for continuous distributions.

*Usage:*

```r
Weibull$mean(...)
```

*Arguments:*

... Unused.

**Method** mode(): The mode of a probability distribution is the point at which the pdf is a local maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

*Usage:*

```r
Weibull$mode(which = "all")
```

*Arguments:*

`which` (character(1) | numeric(1))
- Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies which mode to return.

**Method** median(): Returns the median of the distribution. If an analytical expression is available returns distribution median, otherwise if symmetric returns \( \text{self$mean} \), otherwise returns \( \text{self$quantile(0.5)} \).

*Usage:*

```r
Weibull$median()
```

**Method** variance(): The variance of a distribution is defined by the formula

\[
var_X = E[X^2] - E[X]^2
\]

where \( E_X \) is the expectation of distribution \( X \). If the distribution is multivariate the covariance matrix is returned.

*Usage:*

```r
Weibull$variance(...)
```

*Arguments:*

... Unused.

**Method** skewness(): The skewness of a distribution is defined by the third standardised moment,

\[
sk_X = E_X \left[ \frac{x - \mu}{\sigma} \right]^3
\]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.

*Usage:*

```r
Weibull$skewness(...)
```

*Arguments:*

... Unused.
Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,

\[ k_X = E_X \left( \frac{x - \mu}{\sigma} \right)^4 \]

where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

Usage:
Weibull$kurtosis(excess = TRUE, ...)

Arguments:

excess (logical(1))
  If TRUE (default) excess kurtosis returned.
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by

\[ -\sum (f_X) \log(f_X) \]

where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions.

Usage:
Weibull$entropy(base = 2, ...)

Arguments:

base (integer(1))
  Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method pgf(): The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(z^X)] \]

where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \).

Usage:
Weibull$pgf(z, ...)

Arguments:

z (integer(1))
  z integer to evaluate probability generating function at.
... Unused.

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

Usage:
Weibull$clone(deep = FALSE)

Arguments:

deep  Whether to make a deep clone.
WeightedDiscrete

References


See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, WeightedDiscrete

---

WeightedDiscrete Distribution Class

Description

Mathematical and statistical functions for the WeightedDiscrete distribution, which is commonly used in empirical estimators such as Kaplan-Meier.

Details

The WeightedDiscrete distribution is defined by the pmf,

\[ f(x_i) = p_i \]

for \( p_i, i = 1, \ldots, k; \sum p_i = 1 \).

Sampling from this distribution is performed with the sample function with the elements given as the x values and the pdf as the probabilities. The cdf and quantile assume that the elements are supplied in an indexed order (otherwise the results are meaningless). The number of points in the distribution cannot be changed after construction.

Value

Returns an R6 object inheriting from class SDistribution.

Distribution support

The distribution is supported on \( x_1, \ldots, x_k \).
WeightedDiscrete

Default Parameterisation
WeightDisc(x = 1, pdf = 1)

Omitted Methods
N/A

Also known as
N/A

Super classes
distr6::Distribution -> distr6::SDistribution -> WeightedDiscrete

Public fields
name Full name of distribution.
short_name Short name of distribution for printing.
description Brief description of the distribution.

Active bindings
properties Returns distribution properties, including skewness type and symmetry.

Methods

Public methods:
• WeightedDiscrete$new()
• WeightedDiscrete$strprint()
• WeightedDiscrete$mean()
• WeightedDiscrete$mode()
• WeightedDiscrete$variance()
• WeightedDiscrete$skewness()
• WeightedDiscrete$kurtosis()
• WeightedDiscrete$entropy()
• WeightedDiscrete$mgf()
• WeightedDiscrete$cf()
• WeightedDiscrete$pgf()
• WeightedDiscrete$clone()

Method new(): Creates a new instance of this R6 class.

Usage:
WeightedDiscrete$new(x = NULL, pdf = NULL, cdf = NULL, decorators = NULL)

Arguments:
WeightedDiscrete

x numeric()
  Data samples, must be ordered in ascending order.

pdf numeric()
  Probability mass function for corresponding samples, should be same length x. If cdf is not
given then calculated as cumsum(pdf).

cdf numeric()
  Cumulative distribution function for corresponding samples, should be same length x. If
given then pdf is ignored and calculated as difference of cdfs.

decorators (character())
  Decorators to add to the distribution during construction.


Usage:
WeightedDiscrete$strprint(n = 2)

Arguments:
  n (integer(1))
    Ignored.

Method mean(): The arithmetic mean of a (discrete) probability distribution X is the expectation

\[ E_X(X) = \sum p_X(x) \cdot x \]

with an integration analogue for continuous distributions. If distribution is improper (F(Inf) != 1,
then E_X(x) = Inf).

Usage:
WeightedDiscrete$mean(...)  
Arguments:
... Unused.

Method mode(): The mode of a probability distribution is the point at which the pdf is a local
maximum, a distribution can be unimodal (one maximum) or multimodal (several maxima).

Usage:
WeightedDiscrete$mode(which = "all")

Arguments:
  which (character(1) | numeric(1))
    Ignored if distribution is unimodal. Otherwise "all" returns all modes, otherwise specifies
    which mode to return.

Method variance(): The variance of a distribution is defined by the formula

\[ var_X = E[X^2] - E[X]^2 \]

where \( E_X \) is the expectation of distribution X. If the distribution is multivariate the covariance
matrix is returned. If distribution is improper (F(Inf) != 1, then var_X(x) = Inf).

Usage:
Method skewness(): The skewness of a distribution is defined by the third standardised moment,
\[ sk_X = E_X \left( \frac{x - \mu}{\sigma} \right)^3 \]
where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. If distribution is improper (\( F(\text{Inf}) \neq 1 \)), then \( sk_X(x) = \text{Inf} \).

Usage:
WeightedDiscrete$skewness(...)
Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,
\[ k_X = E_X \left( \frac{x - \mu}{\sigma} \right)^4 \]
where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3. If distribution is improper (\( F(\text{Inf}) \neq 1 \)), then \( k_X(x) = \text{Inf} \).

Usage:
WeightedDiscrete$kurtosis(excess = TRUE, ...)
Arguments:
excess (logical(1))
If TRUE (default) excess kurtosis returned.
... Unused.

Method entropy(): The entropy of a (discrete) distribution is defined by
\[ - \sum (f_X) \log(f_X) \]
where \( f_X \) is the pdf of distribution \( X \), with an integration analogue for continuous distributions. If distribution is improper then entropy is \( \text{Inf} \).

Usage:
WeightedDiscrete$entropy(base = 2, ...)
Arguments:
base (integer(1))
Base of the entropy logarithm, default = 2 (Shannon entropy)
... Unused.

Method mgf(): The moment generating function is defined by
\[ mgf_X(t) = E_X[exp(xt)] \]
where \( X \) is the distribution and \( E_X \) is the expectation of the distribution \( X \). If distribution is improper (\( F(\text{Inf}) \neq 1 \)), then \( mgf_X(x) = \text{Inf} \).
Usage:
WeightedDiscrete$mgf(t, ...)

Arguments:
t (integer(1))
    t integer to evaluate function at.
... Unused.

Method cf(): The characteristic function is defined by

\[ cf_X(t) = E_X[exp(xti)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X. If distribution is improper (\( F(Inf) \neq 1 \), then \( cf_X(x) = Inf \)).

Usage:
WeightedDiscrete$cf(t, ...)

Arguments:
t (integer(1))
    t integer to evaluate function at.
... Unused.

Method pgf(): The probability generating function is defined by

\[ pgf_X(z) = E_X[exp(z^x)] \]

where X is the distribution and \( E_X \) is the expectation of the distribution X. If distribution is improper (\( F(Inf) \neq 1 \), then \( pgf_X(x) = Inf \)).

Usage:
WeightedDiscrete$pgf(z, ...)

Arguments:
z (integer(1))
    z integer to evaluate probability generating function at.
... Unused.

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

Usage:
WeightedDiscrete$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References
See Also

Other discrete distributions: Bernoulli, Binomial, Categorical, Degenerate, DiscreteUniform, EmpiricalMV, Empirical, Geometric, Hypergeometric, Logarithmic, Multinomial, NegativeBinomial

Other univariate distributions: Arcsine, Bernoulli, BetaNoncentral, Beta, Binomial, Categorical, Cauchy, ChiSquaredNoncentral, ChiSquared, Degenerate, DiscreteUniform, Empirical, Erlang, Exponential, FDistributionNoncentral, FDistribution, Frechet, Gamma, Geometric, Gompertz, Gumbel, Hypergeometric, InverseGamma, Laplace, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull

Examples

```r
x <- WeightedDiscrete$new(x = 1:3, pdf = c(1 / 5, 3 / 5, 1 / 5))
WeightedDiscrete$new(x = 1:3, cdf = c(1 / 5, 4 / 5, 1)) # equivalently

# d/p/q/r
x$pdf(1:5)
x$cdf(1:5) # Assumes ordered in construction
x$quantile(0.42) # Assumes ordered in construction
x$rand(10)

# Statistics
x$mean()
x$variance()

summary(x)
```

[.VectorDistribution  Extract one or more Distributions from a VectorDistribution

Description

Once a VectorDistribution has been constructed, use [ to extract one or more Distributions from inside it.

Usage

```r
## S3 method for class 'VectorDistribution'
vecdist[i]
```

Arguments

vecdist  VectorDistribution from which to extract Distributions.
i  indices specifying distributions to extract or ids of wrapped distributions.
Examples

v <- VectorDistribution$new(distribution = "Binom", params = data.frame(size = 1:2, prob = 0.5))
v[1]
v["Binom1"]
Index

* continuous distributions
  Arcsine, 7
  Beta, 18
  BetaNoncentral, 22
  Cauchy, 34
  ChiSquared, 39
  ChiSquaredNoncentral, 43
  Dirichlet, 59
  Erlang, 92
  Exponential, 101
  FDistribution, 106
  FDistributionNoncentral, 110
  Frechet, 113
  Gamma, 118
  Gompertz, 129
  Gumbel, 131
  InverseGamma, 142
  Laplace, 148
  Logistic, 161
  Loglogistic, 167
  Lognormal, 171
  MultivariateNormal, 188
  Normal, 197
  Pareto, 203
  Poisson, 210
  Rayleigh, 223
  ShiftedLoglogistic, 228
  StudentT, 237
  StudentTNoncentral, 241
  Triangular, 258
  Uniform, 271
  Wald, 286
  Weibull, 290

* decorators
  CoreStatistics, 48
  ExoticStatistics, 97
  FunctionImputation, 117

* discrete distributions
  Bernoulli, 13
  Binomial, 24
  Categorical, 29
  Degenerate, 55
  DiscreteUniform, 62
  Empirical, 82
  EmpiricalMV, 87
  Geometric, 124
  Hypergeometric, 138
  Logarithmic, 157
  Multinomial, 183
  NegativeBinomial, 192
  WeightedDiscrete, 294

* kernels
  Cosine, 52
  Epanechnikov, 90
  LogisticKernel, 165
  NormalKernel, 201
  Quartic, 221
  Sigmoid, 232
  Silverman, 234
  TriangularKernel, 264
  Tricube, 265
  Triweight, 267
  UniformKernel, 276

* multivariate distributions
  Dirichlet, 59
  EmpiricalMV, 87
  Multinomial, 183
  MultivariateNormal, 188

* univariate distributions
  Arcsine, 7
  Bernoulli, 13
  Beta, 18
  BetaNoncentral, 22
  Binomial, 24
  Categorical, 29
  Cauchy, 34
  ChiSquared, 39
  ChiSquaredNoncentral, 43
Degenerate, 55
DiscreteUniform, 62
Empirical, 82
Erlang, 92
Exponential, 101
FDistribution, 106
FDistributionNoncentral, 110
Frechet, 113
Gamma, 118
Geometric, 124
Gompertz, 129
Gumbel, 131
Hypergeometric, 138
InverseGamma, 142
Laplace, 148
Logarithmic, 157
Logistic, 161
Loglogistic, 167
Lognormal, 171
NegativeBinomial, 192
Normal, 197
Pareto, 203
Poisson, 210
Rayleigh, 223
ShiftedLoglogistic, 228
StudentT, 237
StudentTNoncentral, 241
Triangular, 258
Uniform, 271
Wald, 286
Weibull, 290
WeightedDiscrete, 294

* wrappers
  Convolution, 47
  DistributionWrapper, 78
  HuberizedDistribution, 136
  MixtureDistribution, 177
  ProductDistribution, 214
  TruncatedDistribution, 269
  VectorDistribution, 277
  .Distribution (ProductDistribution), 214
  .Distribution (Convolution), 47
  .Distribution (Convolution), 47
  [.VectorDistribution, 299

array, 72, 73, 99, 100, 179–181, 217, 218, 282–284
as.Distribution, 11
as.MixtureDistribution, 12
as.ProductDistribution, 12
as.VectorDistribution, 13
assertContinuous (testContinuous), 244
assertDiscrete (testDiscrete), 245
assertDistribution (testDistribution), 245
assertDistributionList (testDistributionList), 246
assertLeptokurtic (testLeptokurtic), 247
assertMatrixvariate (testMatrixvariate), 248
assertMesokurtic (testMesokurtic), 249
assertMixture (testMixture), 250
assertMultivariate (testMultivariate), 250
assertNegativeSkew (testNegativeSkew), 251
assertNoSkew (testNoSkew), 252
assertParameterSet (testParameterSet), 253
assertParameterSetList (testParameterSetList), 254
assertPositiveSkew (testPositiveSkew), 255
assertPlatykurtic (testPlatykurtic), 255
assertSymmetric (testSymmetric), 257
assertUnivariate (testUnivariate), 257

BetaNoncentral, 10, 11, 17, 21, 22, 28, 34, 38, 43, 46, 58, 62, 67, 87, 96, 105,
c.Distribution, 28
checkContinuous (testContinuous), 244
checkDiscrete (testDiscrete), 245
checkDistribution (testDistribution), 245
checkDistributionList (testDistributionList), 246
checkLeptokurtic (testLeptokurtic), 247
checkMatrixvariate (testMatrixvariate), 248
checkMesokurtic (testMesokurtic), 249
checkMixture (testMixture), 250
checkMultivariate (testMultivariate), 250
checkNegativeSkew (testNegativeSkew), 251
checkNoSkew (testNoSkew), 252
checkParameterSet (testParameterSet), 253
checkParameterSetList (testParameterSetList), 254
checkPlatykurtic (testPlatykurtic), 255
checkPositiveSkew (testPositiveSkew), 256
checkSymmetric (testSymmetric), 257
checkUnivariate (testUnivariate), 257
ChiSquared, 10, 11, 17, 21, 24, 28, 34, 38, 39, 46, 58, 67, 72, 87, 90, 96, 105, 106, 110, 112, 116, 123, 128, 131, 135, 142, 146, 152, 161, 165, 171, 176, 192, 197, 201, 207, 212, 226, 232, 241, 244, 263, 275, 290, 294, 299
chol, 188
Convolution, 47, 80, 137, 181, 219, 271, 284
CoreStatistics, 48, 101, 118, 280–282
Cosine, 52, 92, 167, 203, 222, 233, 235, 265, 267, 269, 277
cubature::cubintegrate, 51
data.table::data.table, 72–74, 99, 100, 179–181, 217, 218, 220, 282–284
data.table::data.table(), 208
decorate, 54, 77
Delta (Degenerate), 55
Dirac (Degenerate), 55
distr6(distr6-package), 6
distr6-package, 6
distr6::Distribution, 8, 14, 18, 22, 25, 30, 35, 39, 44, 47, 52, 55, 59, 63, 78, 83, 88, 90, 93, 102, 106, 111, 113, 119,
INDEX


Fisk (Logistic), 167


FunctionImputation, 51, 59, 71–74, 101, 117, 136, 184, 188, 208, 270, 286


Gaussian (Normal), 197
generalPNorm, 123


exkurtosisType, 96

ExoticStatistics, 51, 97, 118
INDEX

197, 201, 207, 214, 226, 232, 241, 
244, 263, 275, 290, 294, 299

graphics::layout(), 208

graphics::par(), 208

Gumbel, 10, 11, 17, 21, 24, 28, 34, 38, 43, 46, 
58, 62, 67, 87, 96, 105, 106, 110, 
112, 116, 123, 128, 131, 131, 142, 
146, 152, 161, 165, 171, 176, 192, 
197, 201, 207, 214, 226, 232, 241, 
244, 263, 275, 290, 294, 299

huberize, 136

HuberizedDistribution, 48, 80, 136, 136, 
181, 219, 271, 284

Hypergeometric, 11, 17, 21, 24, 28, 34, 38, 
43, 46, 58, 62, 67, 87, 96, 105, 106, 110, 
112, 116, 123, 128, 131, 135, 138, 
146, 152, 161, 165, 171, 176, 187, 
197, 201, 207, 214, 226, 232, 241, 
244, 263, 275, 290, 294, 299

integrate, 51

InverseGamma, 10, 11, 17, 21, 24, 28, 34, 38, 
43, 46, 58, 62, 67, 87, 96, 105, 106, 110, 
112, 116, 123, 128, 131, 135, 138, 
142, 142, 152, 161, 165, 171, 176, 
192, 197, 201, 207, 214, 226, 232, 
241, 244, 263, 275, 290, 294, 299

InverseGaussian (Wald), 286

InverseNormal (Wald), 286

InverseWeibull (Frechet), 113

Kernel, 146, 156

Laplace, 10, 11, 17, 21, 24, 28, 34, 38, 43, 46, 
58, 62, 67, 87, 96, 105, 106, 110, 
112, 116, 123, 128, 131, 135, 142, 
146, 148, 161, 165, 171, 176, 192, 
197, 201, 207, 214, 226, 232, 241, 
244, 263, 275, 290, 294, 299

length.VectorDistribution, 153

lines.Distribution, 153, 208

listDecorators, 77, 154

listDecorators(), 54

listDistributions, 155

listDistributions(), 81, 178, 215, 279

listKernels, 156

listWrappers, 78, 156

Logarithmic, 11, 17, 21, 24, 28, 34, 38, 43, 
46, 58, 67, 87, 96, 105, 106, 110, 112,
116, 123, 128, 131, 135, 142, 146, 
152, 157, 165, 171, 176, 187, 197, 
201, 207, 214, 226, 232, 241, 244, 
263, 275, 290, 294, 299

Loggaussian (Lognormal), 171

Logistic, 10, 11, 17, 21, 24, 28, 34, 38, 43, 
46, 58, 62, 67, 87, 96, 105, 106, 110, 
112, 116, 123, 128, 131, 135, 142, 
146, 152, 161, 165, 171, 176, 192, 
197, 201, 207, 214, 226, 232, 241, 
244, 263, 275, 290, 294, 299

LogisticKernel, 53, 92, 165, 203, 222, 233, 
235, 265, 267, 269, 277

Loglogistic, 10, 11, 17, 21, 24, 28, 34, 38, 
43, 46, 58, 62, 67, 87, 96, 105, 106, 110, 
112, 116, 123, 128, 131, 135, 138, 
142, 146, 152, 161, 165, 167, 176, 
192, 197, 201, 207, 214, 226, 228, 
232, 241, 244, 263, 275, 290, 294, 
299

Lognormal, 10, 11, 17, 21, 24, 28, 34, 38, 43, 
46, 58, 62, 67, 87, 96, 105, 106, 110, 
112, 116, 123, 128, 131, 135, 142, 
146, 152, 161, 165, 167, 176, 
197, 201, 207, 214, 226, 232, 241, 
244, 263, 275, 290, 294, 299

makeUniqueDistributions, 176

matrix, 11

MixtureDistribution, 12, 13, 48, 80, 137, 
177, 182, 219, 227, 284

mixtureVector, 182

Multinomial, 17, 28, 34, 58, 62, 67, 87, 90, 
128, 142, 161, 183, 192, 197, 299

MultivariateNormal, 10, 21, 24, 38, 43, 46, 
62, 90, 96, 105, 110, 112, 116, 123, 
131, 135, 146, 152, 165, 171, 176, 
187, 188, 201, 207, 214, 226, 232, 
241, 244, 263, 275, 290, 294, 299

NegativeBinomial, 11, 17, 21, 24, 28, 34, 38, 
43, 46, 58, 67, 87, 90, 96, 106, 110, 
112, 116, 123, 128, 131, 135, 142, 
146, 152, 161, 165, 171, 176, 187, 
192, 201, 207, 214, 226, 232, 241, 
244, 263, 275, 290, 294, 299

Normal, 10, 11, 17, 21, 24, 28, 34, 38, 43, 46, 
58, 62, 67, 87, 96, 105, 106, 110, 
112, 116, 123, 128, 131, 135, 142,
par. 208
ParameterSet, 253, 254
Pareto, 10, 11, 17, 21, 24, 28, 34, 38, 43, 46,
58, 62, 67, 87, 96, 105, 106, 110,
112, 116, 123, 128, 131, 135, 142,
146, 152, 161, 165, 171, 176, 192,
197, 201, 203, 214, 226, 232, 241,
244, 263, 275, 290, 294, 299
plot.Distribution, 153, 154, 207, 209, 210, 220
plot.VectorDistribution, 209
Poisson, 10, 11, 17, 21, 24, 28, 34, 38, 43, 46,
58, 62, 67, 87, 96, 105, 106, 110,
112, 116, 123, 128, 131, 135, 142,
146, 152, 161, 165, 171, 176, 192,
197, 201, 207, 210, 226, 232, 241,
244, 263, 275, 290, 294, 299
pracma::gammaz(), 134
ProductDistribution, 12, 13, 48, 80, 137,
181, 214, 227, 271, 284
qqplot, 220
quantile, 220
Quartic, 53, 92, 167, 203, 221, 233, 235, 265,
267, 269, 277
R6, 8, 14, 19, 23, 25, 30, 35, 40, 44, 47, 56, 60,
64, 69, 77–79, 83, 88, 93, 102, 107,
111, 114, 120, 125, 130, 132, 137,
139, 143, 146, 149, 158, 162, 166,
168, 172, 177, 184, 189, 194, 198,
202, 204, 211, 215, 224, 228, 230,
233, 234, 238, 242, 260, 270, 272,
278, 287, 291, 295
Rayleigh, 10, 11, 17, 21, 24, 28, 34, 38, 43,
46, 58, 62, 67, 87, 96, 105, 106, 110,
112, 116, 123, 128, 131, 135, 142,
146, 152, 161, 165, 171, 176, 192,
197, 201, 207, 214, 223, 232, 241,
244, 263, 275, 290, 294, 299
rep.Distribution, 227
sample, 29, 82, 87, 294
INDEX

paramSetList, 252
paramSet, 253
paramSetList, 254
Platykurtic, 255
dPositiveSkew, 256
dSymmetric, 257
Univariate, 257
TriangularKernel, 53, 92, 167, 203, 222, 233, 235, 264, 267, 269, 277
Tricube, 53, 92, 167, 203, 222, 233, 235, 265, 265, 269, 277
Triweight, 53, 92, 167, 203, 222, 233, 235, 265, 267, 267, 277
trunc, 269
TruncatedDistribution, 48, 80, 137, 181, 219, 269, 269, 284
UniformKernel, 53, 92, 167, 203, 222, 233, 235, 265, 267, 269, 276


INDEX