Package ‘distr6’

December 12, 2020

Title The Complete R6 Probability Distributions Interface

Version 1.4.8


Imports checkmate, data.table, R6, R62S3 (>= 1.4.0), set6 (>= 0.2.0), stats, Rcpp

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

License MIT + file LICENSE

LazyData true


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

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.1.1

Collate 'helpers.R' 'distr6Globals.R' 'Distribution.R'
  'DistributionDecorator.R'
  'DistributionDecorator_CoreStatistics.R'
  'DistributionDecorator_ExoticStatistics.R'
  'DistributionDecorator_FunctionImputation.R'
  'Distribution_Kernel.R' 'Distribution_SDistribution.R'
  'Kernel_Cosine.R' 'Kernel_Epanechnikov.R' 'Kernel_Logistic.R'
  'Kernel_Normal.R' 'Kernel_Quartic.R' 'Kernel_Sigmoid.R'
'Kernel_Silverman.R' 'Kernel_Triangular.R' 'Kernel_Tricube.R'
'Kernel_Triweight.R' 'Kernel_UNIFORM.R' 'ParameterSet.R'
'ParameterSetCollection.R' 'RcppExports.R'
'SDistribution_Arcsine.R' 'SDistribution_Bernoulli.R'
'SDistribution_Beta.R' 'SDistribution_BetaNoncentral.R'
'SDistribution_Binomial.R' 'SDistribution_Categorical.R'
'SDistribution_Cauchy.R' 'SDistribution_ChiSquared.R'
'SDistribution_ChiSquaredNoncentral.R'
'SDistribution_Degenerate.R' 'SDistribution.Dirichlet.R'
'SDistribution_DiscreteUniform.R' 'SDistribution_Empirical.R'
'SDistribution_EmpiricalMultivariate.R'
'SDistribution_Erlang.R' 'SDistribution_Exponential.R'
'SDistribution_FDistribution.R'
'SDistribution_FDistributionNoncentral.R'
'SDistribution_Frechet.R' 'SDistribution_Gamma.R'
'SDistribution_Geometric.R' 'SDistribution_Gompertz.R'
'SDistribution_Gumbel.R' 'SDistribution_Hypergeometric.R'
'SDistribution_InverseGamma.R' 'SDistribution_Laplace.R'
'SDistribution_Logarithmic.R' 'SDistribution_Logistic.R'
'SDistribution_Loglogistic.R' 'SDistribution_Lognormal.R'
'SDistribution_Multinomial.R'
'SDistribution_MultivariateNormal.R'
'SDistribution_NegBinomial.R' 'SDistribution_Normal.R'
'SDistribution_Pareto.R' 'SDistribution_Poisson.R'
'SDistribution_Rayleigh.R' 'SDistribution_ShiftedLogistic.R'
'SDistribution_StudentT.R' 'SDistribution_StudentTNoncentral.R'
'SDistribution_Triangular.R' 'SDistribution_Uniform.R'
'SDistribution_Wald.R' 'SDistribution_Weibull.R'
'SDistribution_WeightedDiscrete.R' 'Wrapper.R'
'Wrapper_Convolution.R' 'Wrapper_HuberizedDistribution.R'
'Wrapper_MixtureDistribution.R' 'Wrapper_ProductDistribution.R'
'Wrapper_Scale.R' 'Wrapper_TruncatedDistribution.R'
'Wrapper_VectorDistribution.R' 'assertions.R'
'c.Distribution.R' 'decomposeMixture.R' 'decorate.R'
'distr6-deprecated.R' 'distr6-package.R' 'distr6.news.R'
'distrSimulate.R' 'exkurtosisType.R' 'generalPNorm.R'
'getParameterSet.R' 'helpers_pdq.R' 'helpers_wrappers.R'
'isPdqr.R' 'lines_continuous.R' 'lines_discrete.R' 'lines.R'
'listDecorators.R' 'listDistributions.R' 'listKernels.R'
'listWrappers.R' 'makeUniqueDistributions.R' 'measures.R'
'mixturise.Vector.R' 'plot_continuous.R' 'plot_discrete.R'
'plot_distribution.R' 'plot_multivariate.R'
'plot_vectordistribution.R' 'qgplot.R' 'rep.Distribution.R'
'sets.R' 'simulateEmpiricalDistribution.R' 'skewType.R' 'zzz.R'

LinkingTo Rcpp

NeedsCompilation yes

Author Raphael Sonabend [aut, cre] (<https://orcid.org/0000-0001-9225-4654>),
Franz Kiraly [aut],
R topics documented:

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 <raphael.sonabend.15@ucl.ac.uk>
Repository CRAN
Date/Publication 2020-12-12 15:50:06 UTC

R topics documented:

distr6-package ................................................. 7
Arcsine .......................................................... 8
as.data.table.ParameterSet ................................... 12
as.MixtureDistribution ....................................... 12
as.ParameterSet ............................................... 13
as.ProductDistribution ....................................... 14
as.VectorDistribution ....................................... 14
Bernoulli ..................................................... 14
Beta ............................................................. 19
BetaNoncentral ................................................. 23
Binomial ........................................................ 25
c.Distribution .................................................. 29
Categorical ..................................................... 30
Cauchy ........................................................... 35
cdf ............................................................... 39
cdfAntiDeriv .................................................... 40
cdfPNorm ........................................................ 41
cdfSquared2Norm .............................................. 41
cf ................................................................. 42
ChiSquared ...................................................... 42
ChiSquaredNoncentral ........................................ 46
Convolution ...................................................... 50
CoreStatistics ................................................. 52
correlation ..................................................... 55
Cosine ........................................................... 56
cum Hazard ...................................................... 58
decorate .......................................................... 58
decorators ....................................................... 59
R topics documented:

- Degenerate .................................................. 60
- Dirichlet ..................................................... 64
- DiscreteUniform ............................................. 67
- distr6News .................................................. 71
- Distribution .................................................. 72
- DistributionDecorator ...................................... 81
- DistributionWrapper ....................................... 82
- distrSimulate ............................................... 84
- dmax ........................................................ 85
- dmin ........................................................ 86
- Empirical ..................................................... 87
- EmpiricalMV .................................................. 91
- entropy ....................................................... 94
- Epanechnikov ................................................ 94
- Erlang ......................................................... 96
- exkurtosisType ............................................. 100
- ExoticStatistics ........................................... 101
- Exponential ................................................ 105
- FDistribution .............................................. 110
- FDistributionNoncentral .................................. 114
- Frechet ....................................................... 117
- FunctionImputation ....................................... 121
- Gamma ....................................................... 122
- generalPNorm ............................................... 127
- genExp ....................................................... 128
- Geometric ................................................... 129
- getParameterSupport ...................................... 133
- getParameterValue ........................................ 134
- Gompertz .................................................... 134
- Gumbel ....................................................... 136
- hazard ........................................................ 140
- huberize ..................................................... 141
- HuberizedDistribution .................................... 142
- Hypergeometric ........................................... 144
- inf .......................................................... 147
- InverseGamma ............................................... 148
- iqr .......................................................... 152
- Kernel ......................................................... 152
- kthmoment ................................................... 155
- kurtosis ..................................................... 155
- kurtosisType ................................................. 156
- Laplace ....................................................... 156
- liesInSupport .............................................. 161
- liesInType .................................................. 161
- lines.Distribution ........................................ 162
- listDecorators ............................................. 163
- listDistributions ......................................... 164
- listKernels .................................................. 165
listWrappers ............................................................... 165
Logarithmic ............................................................... 166
Logistic ................................................................... 170
LogisticKernel ........................................................... 174
Loglogistic ............................................................... 176
Lognormal ............................................................... 180
makeUniqueDistributions ............................................... 185
mean.Distribution ...................................................... 186
median.Distribution .................................................... 186
merge.ParameterSet ................................................... 187
mgf ....................................................................... 187
MixtureDistribution .................................................. 188
mixturiseVector ....................................................... 193
mode .................................................................... 194
Multinomial .............................................................. 194
MultivariateNormal .................................................... 198
NegativeBinomial ...................................................... 202
Normal ................................................................. 207
NormalKernel ........................................................... 211
parameters .............................................................. 213
ParameterSet ............................................................ 214
ParameterSetCollection ................................................ 222
Pareto ................................................................. 227
pdf ..................................................................... 232
pdfPNorm ............................................................... 233
pdfSquared2Norm ....................................................... 233
pgf ..................................................................... 234
plot.Distribution ....................................................... 234
plot.VectorDistribution ............................................... 236
Poisson ................................................................. 237
prec ..................................................................... 241
print.ParameterSet ..................................................... 241
ProductDistribution ................................................... 242
properties .............................................................. 247
qqplot ................................................................. 248
quantile.Distribution ................................................ 249
Quartic ................................................................. 250
rand ................................................................... 252
Rayleigh ............................................................... 252
rep.Distribution ........................................................ 256
SDistribution ........................................................... 256
setParameterValue ...................................................... 257
ShiftedLoglogistic ..................................................... 258
Sigmoid ............................................................... 261
Silverman .............................................................. 263
simulateEmpiricalDistribution .................................... 265
skewness ............................................................... 266
skewnessType ........................................................ 266
R topics documented:

skewType ......................................................... 267
stdev ............................................................ 267
strprint .......................................................... 268
StudentT ......................................................... 269
StudentTNoncentral ............................................. 273
summary.Distribution .......................................... 275
sup ............................................................... 276
support ............................................................ 276
survival ........................................................... 277
survivalAntiDeriv ............................................... 278
survivalPNorm ................................................... 278
symmetry .......................................................... 279
testContinuous .................................................. 279
testDiscrete ...................................................... 280
testDistribution .................................................. 281
testDistributionList ............................................. 282
testLeptokurtic ................................................... 283
testMatrixvariate ............................................... 284
testMesokurtic .................................................... 285
testMixture ....................................................... 286
testMultivariate .................................................. 286
testNegativeSkew ................................................ 287
testNoSkew ....................................................... 288
testParameterSet ................................................... 289
testParameterSetCollection .................................... 290
testParameterSetCollectionList ............................... 291
testParameterSetList ............................................. 292
testPlatykurtic .................................................... 293
testPositiveSkew ................................................ 294
testSymmetric ..................................................... 295
testUnivariate .................................................... 295
traits ............................................................... 296
Triangular ......................................................... 297
TriangularKernel .................................................. 302
Tricube ............................................................ 304
Triweight .......................................................... 306
truncate ............................................................ 308
TruncatedDistribution .......................................... 308
type ................................................................. 310
Uniform ........................................................... 311
UniformKernel ..................................................... 315
valueSupport ...................................................... 317
variance ............................................................ 317
variateForm ........................................................ 318
VectorDistribution ............................................... 318
Wald ............................................................... 326
Weibull ............................................................. 330
WeightedDiscrete ................................................. 334
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.

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:
\texttt{vignette("distr6","distr6")}

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

**Author(s)**

**Maintainer**: Raphael Sonabend \(<\texttt{raphael.sonabend.15@ucl.ac.uk}>\) (ORCID)

Authors:

- Franz Kiraly \(<\texttt{f.kiraly@ucl.ac.uk}>\)

Other contributors:

- Peter Ruckdeschel \(<\texttt{peter.ruckdeschel@uni-oldenburg.de}>\) (Author of distr) [contributor]
- Matthias Kohl \(<\texttt{Matthias.Kohl@stamats.de}>\) (Author of distr) [contributor]
- Shen Chen \(<\texttt{seanchen9832@icloud.com}>\) [contributor]
Arcsine

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.

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 \(-\infty < a \leq b < \infty\).

The distribution is supported on \([a, b]\).

Value

Returns an R6 object inheriting from class SDistribution.

Super classes

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

Public fields

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

Methods

Public methods:

- Arcsine$new()
- Arcsine$mean()
- Arcsine$mode()
- Arcsine$variance()
- Arcsine$skewness()
- Arcsine$kurtosis()
- Arcsine$entropy()
- Arcsine$pgf()
- Arcsine$setParameterValue()
- Arcsine$clone()

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

Usage:
Arcsine$new(lower = 0, upper = 1, 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

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

with an integration analogue for continuous distributions.

Usage:
Arcsine$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:
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,

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

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:
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

\texttt{Arcsine}$\texttt{entropy(base = 2, ...)}$

\textit{Arguments:}

\begin{itemize}
  \item base (integer(1))
  \begin{itemize}
    \item Base of the entropy logarithm, default = 2 (Shannon entropy)
  \end{itemize}
  \item ... Unused.
\end{itemize}

\textbf{Method} \texttt{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 \( X \).

\textit{Usage:}

\texttt{Arcsine$pgf(z, ...)}$

\textit{Arguments:}

\begin{itemize}
  \item z (integer(1))
  \begin{itemize}
    \item \( z \) integer to evaluate probability generating function at.
  \end{itemize}
  \item ... Unused.
\end{itemize}

\textbf{Method} \texttt{setParameterValue()}: Sets the value(s) of the given parameter(s).

\textit{Usage:}

\texttt{Arcsine$setParameterValue(..., lst = NULL, error = "warn")}$

\textit{Arguments:}

\begin{itemize}
  \item ... \texttt{ANY}
  \begin{itemize}
    \item Named arguments of parameters to set values for. See examples.
  \end{itemize}
  \item lst (list(1))
  \begin{itemize}
    \item Alternative argument for passing parameters. List names should be parameter names and list values are the new values to set.
  \end{itemize}
  \item error (character(1))
  \begin{itemize}
    \item If "warn" then returns a warning on error, otherwise breaks if "stop".
  \end{itemize}
\end{itemize}

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

\textit{Usage:}

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

\textit{Arguments:}

\begin{itemize}
  \item deep \begin{itemize}
    \item Whether to make a deep clone.
  \end{itemize}
\end{itemize}

\textbf{References}

as.MixtureDistribution

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

---

as.data.table.ParameterSet

Coerce ParameterSet to data.table

Description

Coerces a ParameterSet to a data.table.

Usage

```r
## S3 method for class 'ParameterSet'
as.data.table(x, ...)
```

Arguments

- `x`: ParameterSet
- `...`: Ignored.

Value

A data.table.

---

as.MixtureDistribution

Coercion to Mixture Distribution

Description

Helper functions to quickly convert compatible objects to a MixtureDistribution.

Usage

```r
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.

Description

Coerces objects to ParameterSet.

Usage

\[
\text{as.ParameterSet}(x, \ldots)
\]

## S3 method for class 'data.table'
\[
\text{as.ParameterSet}(x, \ldots)
\]

## S3 method for class 'list'
\[
\text{as.ParameterSet}(x, \ldots)
\]

Arguments

x: object

\ldots: additional arguments

Details

Currently supported coercions are from data tables and lists. Function assumes that the data table columns are the correct inputs to a ParameterSet, see the constructor for details. Similarly for lists, names are taken to be ParameterSet parameters and values taken to be arguments.

Value

An R6 object of class ParameterSet.

See Also

ParameterSet
as.ProductDistribution

_Coercion to Product Distribution_

**Description**

Helper functions to quickly convert compatible objects to a ProductDistribution.

**Usage**

```r
as.ProductDistribution(object)
```

**Arguments**

- `object` MixtureDistribution or VectorDistribution

---

as.VectorDistribution  

_Coercion to Vector Distribution_

**Description**

Helper functions to quickly convert compatible objects to a VectorDistribution.

**Usage**

```r
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,

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

for probability \( p \).

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

Value

Returns an R6 object inheriting from class SDistribution.

Super classes

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

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.

Methods

Public methods:

- \text{Bernoulli}\$new()
- \text{Bernoulli}\$mean()
- \text{Bernoulli}\$mode()
- \text{Bernoulli}\$median()
- \text{Bernoulli}\$variance()
- \text{Bernoulli}\$skewness()
- \text{Bernoulli}\$kurtosis()
- \text{Bernoulli}\$entropy()
- \text{Bernoulli}\$mgf()
- \text{Bernoulli}\$cf()
- \text{Bernoulli}\$pgf()
- \text{Bernoulli}\$setParameterValue()
- \text{Bernoulli}\$clone()

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

Usage:

\text{Bernoulli}\$new(prob = 0.5, qprob = NULL, decorators = NULL)

Arguments:
Bernoulli

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

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

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.
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:
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 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:

Bernoulli$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:

Bernoulli$setParameterValue(..., lst = NULL, error = "warn")

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".

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

Usage:

Bernoulli$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

References

Michael P. McLaughlin.
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.

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

Value

Returns an R6 object inheriting from class SDistribution.

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.
Methods

Public methods:

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

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

Usage:
`Beta$new(shape1 = 1, shape2 = 1, 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:
`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:
`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 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:*

Beta$variance(...)

*Arguments:*

... Unused.

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

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

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:*

Beta$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:*

Beta$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:*

Beta$entropy(base = 2, ...)

*Arguments:*

base (integer(1))

Base of the entropy logarithm, default = 2 (Shannon entropy)
Method \texttt{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 \( X \).

\textit{Usage:}
\begin{verbatim}
Beta$pgf(z, ...)
\end{verbatim}

\textit{Arguments:}
\begin{itemize}
  \item \( z \) (integer(1))
    \begin{itemize}
      \item \( z \) integer to evaluate probability generating function at.
    \end{itemize}
  \end{itemize}

Method \texttt{setParameterValue()}: Sets the value(s) of the given parameter(s).

\textit{Usage:}
\begin{verbatim}
Beta$setParameterValue(..., lst = NULL, error = "warn")
\end{verbatim}

\textit{Arguments:}
\begin{itemize}
  \item \texttt{... ANY}
    \begin{itemize}
      \item Named arguments of parameters to set values for. See examples.
    \end{itemize}
  \end{itemize}
\begin{itemize}
  \item \texttt{lst} (list(1))
    \begin{itemize}
      \item Alternative argument for passing parameters. List names should be parameter names and list values are the new values to set.
    \end{itemize}
  \end{itemize}
\begin{itemize}
  \item \texttt{error} (character(1))
    \begin{itemize}
      \item If "warn" then returns a warning on error, otherwise breaks if "stop".
    \end{itemize}
  \end{itemize}

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

\textit{Usage:}
\begin{verbatim}
Beta$clone(deep = FALSE)
\end{verbatim}

\textit{Arguments:}
\begin{itemize}
  \item \texttt{deep} Whether to make a deep clone.
\end{itemize}

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,
Noncentral Beta Distribution Class

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,

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

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

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

Value
Returns an R6 object inheriting from class SDistribution.

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.

Methods
Public methods:
- BetaNoncentral$new()
- BetaNoncentral$setParameterValue()
- BetaNoncentral$clone()
Method \texttt{new()}: Creates a new instance of this \texttt{R6} class.

\textit{Usage:}
\begin{verbatim}
BetaNoncentral$new(shape1 = 1, shape2 = 1, location = 0, decorators = NULL)
\end{verbatim}

\textit{Arguments:}
\begin{itemize}
  \item \texttt{shape1} (\texttt{numeric(1)})
    \begin{itemize}
      \item First shape parameter, \texttt{shape1} > 0.
    \end{itemize}
  \item \texttt{shape2} (\texttt{numeric(1)})
    \begin{itemize}
      \item Second shape parameter, \texttt{shape2} > 0.
    \end{itemize}
  \item \texttt{location} (\texttt{numeric(1)})
    \begin{itemize}
      \item Location parameter, defined on the non-negative Reals.
    \end{itemize}
  \item \texttt{decorators} (\texttt{character()})
    \begin{itemize}
      \item Decorators to add to the distribution during construction.
    \end{itemize}
\end{itemize}

Method \texttt{setParameterValue()}: Sets the value(s) of the given parameter(s).

\textit{Usage:}
\begin{verbatim}
BetaNoncentral$setParameterValue(..., lst = NULL, error = "warn")
\end{verbatim}

\textit{Arguments:}
\begin{itemize}
  \item \texttt{... ANY}
    \begin{itemize}
      \item Named arguments of parameters to set values for. See examples.
    \end{itemize}
  \item \texttt{lst} (\texttt{list(1)})
    \begin{itemize}
      \item Alternative argument for passing parameters. List names should be parameter names and
        list values are the new values to set.
    \end{itemize}
  \item \texttt{error} (\texttt{character(1)})
    \begin{itemize}
      \item If "warn" then returns a warning on error, otherwise breaks if "stop".
    \end{itemize}
\end{itemize}

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

\textit{Usage:}
\begin{verbatim}
BetaNoncentral$clone(deep = FALSE)
\end{verbatim}

\textit{Arguments:}
\begin{itemize}
  \item \texttt{deep} Whether to make a deep clone.
\end{itemize}

\textbf{Author(s)}

Jordan Deenichin

\textbf{References}

Michael P. McLaughlin.

\textbf{See Also}

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

Other univariate distributions: Arcsine, Bernoulli, 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

---

**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,

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

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

The distribution is supported on 0, 1, ..., n.

**Value**

Returns an R6 object inheriting from class SDistribution.

**Super classes**

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

**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.
Binomial

Methods

Public methods:

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

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

*Usage:*

```
Binomial$new(size = 10, prob = 0.5, 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.

**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:*

```
Binomial$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:*

```
Binomial$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:
Binomial$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:
Binomial$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:
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, ...)

Method modes(): The number of modes in a distribution is defined by the number of modes,

\[ G \text{ modes} \]

where \( G \) is the number of modes.
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:
Binomial$mgf(t, \ldots)

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:
Binomial(cf(t, \ldots)

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, \ldots)

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:
Binomial$setParameterValue(..., lst = NULL, error = "warn")

Arguments:
Description

Helper function for quickly combining distributions into a VectorDistribution.

Usage

```r
## 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

```r
# 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)
```

Categorical Distribution Class

Description
Mathematical and statistical functions for the Categorical distribution, which is commonly used in classification supervised learning.

Details
The Categorical distribution parameterised with a given support set, \(x_1, ..., x_k\), and respective probabilities, \(p_1, ..., p_k\), is defined by the pmf,

\[
f(x_i) = p_i
\]

for \(p_i, i = 1, ..., k; \sum p_i = 1\).

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

Sampling from this distribution is performed with the `sample` function with the elements given as the support set and the probabilities from the `probs` parameter. The cdf and quantile assumes 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.

Super classes

distr6::Distribution -> distr6::SDistribution -> Categorical

Public fields

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

Methods

Public methods:

• Categorical$new()
• Categorical$mean()
• Categorical$mode()
• Categorical$variance()
• Categorical$skewness()
• Categorical$entropy()
• Categorical$mgf()
• Categorical$cf()
• Categorical$pgf()
• Categorical$setParameterValue()
• Categorical$clone()

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

Usage:
Categorical$new(elements = 1, probs = 1, 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
Categorical

```r
x$setParameterValue(probs = c(0.1, 0.2, 0.7))

# d/p/q/r
x$pdf(c("Bapple", "Carrot", 1, 2))
```

```r
x$cdf("Banana") # Assumes ordered in construction
x$quantile(0.42) # Assumes ordered in construction
x$rand(10)
```

# Statistics
```r
x$mode()
```

summary(x)

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

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

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

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

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,

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

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:
Categorical$skewness(...)

Arguments:
... Unused.

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

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

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:
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

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

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

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

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

Usage:
Categorical$setParameterValue(..., lst = NULL, error = "warn")

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".

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

### 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,

$$f(x) = 1/(\pi\beta(1 + ((x - \alpha)/\beta)^2))$$

for $\alpha \in \mathbb{R}$ and $\beta > 0$.

The distribution is supported on the Reals.

---

**Examples**

```r
# Method `Categorical$new`
# Note probabilities are automatically normalised (if not vectorised)
x <- Categorical$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)
```
Value

Returns an R6 object inheriting from class SDistribution.

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 = 0, scale = 1, 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:**

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,

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

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:**

Cauchy$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:**

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, ...)

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

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

$$pgf_X(z) = E_X[exp(z^2)]$$

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

Usage:
Cauchy$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:
Cauchy$clone(deep = FALSE)

Arguments:
  deep  Whether to make a deep clone.

Author(s)
  Chijing Zeng

References
  Michael P. McLaughlin.

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, Logarithmic, Logistic, Loglogistic, Lognormal, NegativeBinomial, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

cdf

Cumulative Distribution Function

Description
  See Distribution$cdf

Usage
  cdf(object, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE, data = NULL)
Arguments

object (Distribution)

... (numeric())

Points to evaluate the cumulative distribution function of the distribution. 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, $X > x$.

log.p logical(1)

If TRUE returns log-cdf. Default is FALSE.

simplify logical(1)

If TRUE (default) simplifies the pdf 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.

Value

Cdf evaluated at given points as either a numeric if simplify is TRUE or as a data.table::data.table.

cdfAntiDeriv Cumulative Distribution Function Anti-Derivative

Description

The anti-derivative of the cumulative distribution function between given limits or over the full support.

Usage

cdfAntiDeriv(object, lower = NULL, upper = NULL)

Arguments

object Distribution.

lower lower limit for integration, default is infimum.

upper upper limit for integration, default is supremum.

Value

Antiderivative of the cdf evaluated between limits as a numeric.
### cdfPNorm

**Cumulative Distribution Function P-Norm**

**Description**

The p-norm of the cdf evaluated between given limits or over the whole support.

**Usage**

```r
cdfPNorm(object, p = 2, lower = NULL, upper = NULL)
```

**Arguments**

- `object`: Distribution.
- `p`: p-norm to calculate.
- `lower`: lower limit for integration, default is infimum.
- `upper`: upper limit for integration, default is supremum.

**Value**

Given p-norm of cdf evaluated between limits as a numeric.

### cdfSquared2Norm

**Squared Cumulative Distribution Function 2-Norm**

**Description**

The squared 2-norm of the cdf evaluated up to a given limit, possibly shifted.

**Usage**

```r
cdfSquared2Norm(object, x = 0, upper = Inf)
```

**Arguments**

- `object`: Distribution.
- `x`: amount to shift the result.
- `upper`: upper limit of the integral.

**Value**

Squared 2-norm of cdf evaluated between limits as a numeric.
### ChiSquared

#### Characteristic Function

**Description**

Characteristic function of a distribution

**Usage**

\[
\text{cf}(\text{object}, t, \ldots)
\]

**Arguments**

- **object**: Distribution.
- **t**: integer to evaluate characteristic function at.
- **...**: Passed to $\text{genExp}$.

**Value**

Characteristic function evaluated at t as a numeric.

### 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,

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

for \( \nu > 0 \).

The distribution is supported on the Positive Reals.

**Value**

Returns an R6 object inheriting from class SDistribution.

**Super classes**

\[
distr6::Distribution \rightarrow distr6::SDistribution \rightarrow \text{ChiSquared}
\]
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.

Methods

Public methods:

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

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

Usage:
ChiSquared$new(df = 1, 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,

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

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:
ChiSquared$skewness(...)

Arguments:
... Unused.

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

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

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:
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

$$- \sum (fx) \log(fx)$$

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)
... Unused.

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

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

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

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

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

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

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

Usage:
ChiSquared$setParameterValue(..., lst = NULL, error = "warn")
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".

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

Michael P. McLaughlin.

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.
ChiSquaredNoncentral

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

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

for \( \nu \geq 0, \lambda \geq 0 \).
The distribution is supported on the Positive Reals.

Value
Returns an R6 object inheriting from class SDistribution.

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.

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

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

Usage:
ChiSquaredNoncentral$new(df = 1, location = 0, 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

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

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,

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

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:*
ChiSquaredNoncentral$skewness(...)

*Arguments:*
... Unused.

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

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

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:*
ChiSquaredNoncentral$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:
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 setParameterValue(): Sets the value(s) of the given parameter(s).

Usage:
ChiSquaredNoncentral$setParameterValue(..., lst = NULL, error = "warn")

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".

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

Usage:
ChiSquaredNoncentral$clone(deep = FALSE)

Arguments:
deepe 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
```

```r
## 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 substracts.

**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

```r
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)
```

CoreStatistics

Core Statistical Methods Decorator

Description

This decorator adds numeric methods for missing analytic expressions in `Distributions` as well as adding generalised expectation and moments functions.

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 -> 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.
correlation

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

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

correlation Distribution Correlation

Description
Correlation of a distribution.

Usage
correlation(object)
Arguments

object Distribution.

Value

Either '1' if distribution is univariate or the correlation as a numeric or matrix.

Description

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

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

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:
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:

\texttt{Cosine\$cdfSquared2Norm(x = 0, upper = 0)}

Arguments:

\begin{itemize}
\item \texttt{x} (numeric(1))
\hspace{1em} Amount to shift the result.
\item \texttt{upper} (numeric(1))
\hspace{1em} Upper limit of the integral.
\end{itemize}

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.

Usage:

\texttt{Cosine\$variance(...)}

Arguments:

\begin{itemize}
\item \texttt{...} Unused.
\end{itemize}

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

Usage:

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

Arguments:

\begin{itemize}
\item \texttt{deep} Whether to make a deep clone.
\end{itemize}

See Also

Other kernels: \texttt{Epanechnikov}, \texttt{LogisticKernel}, \texttt{NormalKernel}, \texttt{Quartic}, \texttt{Sigmoid}, \texttt{Silverman}, \texttt{TriangularKernel}, \texttt{Tricube}, \texttt{Triweight}, \texttt{UniformKernel}
---

### cumHazard

**Cumulative Hazard Function**

**Description**

See ExoticStatistics$\text{cumHazard}$. 

**Usage**

```r
cumHazard(object, ..., log = FALSE, simplify = TRUE, data = NULL)
```

**Arguments**

- `object` *(Distribution)*
- `...` *(numeric())*
  
  Points to evaluate the probability density function of the distribution. 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 log-cumHazard Default is FALSE.
- `simplify` logical(1)
  
  If TRUE (default) simplifies the pdf 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.

**Value**

Cumulative hazard function as a numeric, natural logarithm returned if log is TRUE.

---

### decorate

**Decorate Distributions**

**Description**

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

**Usage**

```r
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"))
```

---

**decorators**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the decorators added to a distribution.</td>
</tr>
</tbody>
</table>

**Usage**

```r
decorators(object)
```
**Degenerate Arguments**

object Distribution.

**Value**

Character vector of decorators.

**R6 Usage**

$decorators

---

**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,

$$f(x) = 1, \text{ if } x = \mu$$

$$f(x) = 0, \text{ if } x \neq \mu$$

for $\mu \in \mathbb{R}$.

The distribution is supported on $\mu$.

Also known as the Dirac distribution.

**Value**

Returns an R6 object inheriting from class SDistribution.

**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.
Methods

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

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

Usage:
`Degenerate$new(mean = 0, 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

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

with an integration analogue for continuous distributions.

Usage:
`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:
`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,
\[
\text{sk}_X = E_X\left[\frac{x - \mu}{\sigma}\right]^3
\]
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:
Degenerate$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:
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)
... 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:*

\$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:*

\$cf(t, ...)

*Arguments:*

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

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

*Usage:*

\$setParameterValue(..., lst = NULL, error = "warn")

*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".

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

*Usage:*

\$clone(deep = FALSE)

*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,

\[
f(x_1, \ldots, x_k) = \frac{\prod_i \Gamma(\alpha_i) / \Gamma(\sum \alpha_i)}{\prod_i (x_i^{\alpha_i - 1})}
\]

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

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

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

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

### Value

Returns an R6 object inheriting from class SDistribution.

### 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.

**Methods**

**Public methods:**

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

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

*Usage:*

```
Dirichlet$new(params = c(1, 1), 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

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

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 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
See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, 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 \(a, b \in \mathbb{Z}; \ b \geq a\).

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

Value

Returns an R6 object inheriting from class SDistribution.

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.
Methods

Public methods:

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

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

Usage:

`DiscreteUniform$new(lower = 0, upper = 1, 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

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

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:
DiscreteUniform$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:
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)
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:}
\DiscreteUniform$mgf(t, \ldots)$

\textit{Arguments:}
\begin{itemize}
  \item \( t \) (integer(1))
    \begin{itemize}
      \item \( t \) integer to evaluate function at.
    \end{itemize}
  \end{itemize}

... 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:}
\DiscreteUniform$cf(t, \ldots)$

\textit{Arguments:}
\begin{itemize}
  \item \( t \) (integer(1))
    \begin{itemize}
      \item \( t \) integer to evaluate function at.
    \end{itemize}
  \end{itemize}

... Unused.

Method \texttt{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.

\textit{Usage:}
\DiscreteUniform$pgf(z, \ldots)$

\textit{Arguments:}
\begin{itemize}
  \item \( z \) (integer(1))
    \begin{itemize}
      \item \( z \) integer to evaluate probability generating function at.
    \end{itemize}
  \end{itemize}

... Unused.

Method \texttt{setParameterValue()}: Sets the value(s) of the given parameter(s).

\textit{Usage:}
\DiscreteUniform$setParameterValue(..., lst = NULL, error = "warn")$

\textit{Arguments:}
\begin{itemize}
  \item \ldots \ ANY
    \begin{itemize}
      \item Named arguments of parameters to set values for. See examples.
    \end{itemize}
  \end{itemize}

\begin{itemize}
  \item \( lst \) (list(1))
    \begin{itemize}
      \item Alternative argument for passing parameters. List names should be parameter names and list values are the new values to set.
    \end{itemize}
  \end{itemize}
error (character(1))
   If "warn" then returns a warning on error, otherwise breaks if "stop".

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

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

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

**References**
Michael P. McLaughlin.

**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**

*Show distr6 NEWS.md File*

**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()
```

```r
## 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$print()**
- **Distribution$summary()**
• Distribution$parameters()
• Distribution$getParameterValue()
• Distribution$setParameterValue()
• Distribution$pdf()
• Distribution$cdf()
• Distribution$quantile()
• Distribution$rand()
• Distribution$prec()
• Distribution$stdev()
• Distribution$median()
• Distribution$iqr()
• Distribution$correlation()
• Distribution$liesInSupport()
• Distribution$liesInType()
• Distribution$workingSupport()
• Distribution$clone()

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

Usage:
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 type of the distribution.
pdf function(1)
  Probability density function of the distribution. At least one of pdf and cdf must be pro-
vided.
cdf function(1)
  Cumulative distribution function of the distribution. At least one of pdf and cdf must be pro-
vided.
quantile function(1)
  Quantile (inverse-cdf) function of the distribution.
rand function(1)
  Simulation function for drawing random samples from the distribution.
parameters ([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, de-
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.


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 summary.
... ANY  
Unused. Added for consistency.

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

**Usage:**
Distribution$parameters(id = NULL)

**Arguments:**
id character()  
id of parameter value to return.

**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 = NULL, error = "warn")

**Arguments:**
... ANY  
Named arguments of parameters to set values for. See examples.
lst (list())  
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".

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

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

#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:

Distribution$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.

Examples:
    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)))

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:
    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.

Examples:
    b <- Binomial$new()
b$rand(10)
mvn <- MultivariateNormal$new()
mvn$rand(5)

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

Usage:
    Distribution$prec()

Method stdev(): Returns the standard deviation of the distribution as sqrt(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 self$mean, otherwise returns self$quantile(0.5).

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

Arguments:
na.rm (logical(1))
   Ignored, added for consistency.
   ... ANY
   Ignored, added for consistency.

**Method** `iqr()`: Inter-quartile range of the distribution. Estimated as `self$quantile(0.75) - self$quantile(0.25)`.

*Usage*

```r
Distribution$iqr()
```

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

*Usage*

```r
Distribution$correlation()
```

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

*Usage*

```r
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*

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

## Method `Distribution$quantile`

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)))
```
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.
DistributionWrapper

Usage:
DistributionDecorator$new()

Method decorate(): Decorates the given distribution with the methods available in this decor-
ator.

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: Ele-
ments of Reusable Object-Oriented Software.” Addison-Wesley.

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.

Value
Returns error. Abstract classes cannot be constructed directly.

Super class
distr6::Distribution -> DistributionWrapper
Methods

Public methods:

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

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

Usage:

```r
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 ([ParameterSetCollection])
  Optional parameters to add to the internal collection, ignored if distlist is given.

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

Usage:
DistributionWrapper$setParameterValue(..., lst = NULL, error = "warn")

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".

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

```r
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.

See Also

- `rand`

---

### dmax

**Distribution Maximum Accessor**

**Description**

Returns the distribution maximum as the maximum of the support. If the support is not bounded above then maximum is given by

\[ maximum = \text{supremum} - 1.1e - 15 \]

**Usage**

```r
dmax(object)
```

**Arguments**

- `object`: Distribution.
Value

Maximum as a numeric.

R6 Usage

$dmax$

See Also

support, dmin, sup, inf

---

dmin

**Distribution Minimum Accessor**

**Description**

Returns the distribution minimum as the minimum of the support. If the support is not bounded below then minimum is given by

\[
\text{minimum} = \text{infimum} + 1.1e - 15
\]

**Usage**

dmin(object)

**Arguments**

object Distribution.

**Value**

Minimum as a numeric.

**R6 Usage**

$dmin$
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 \( x_i \in \mathbb{R}, i = 1, ..., k \).

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

Sampling from this distribution is performed with the \texttt{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 \texttt{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.

Super classes

distr6::Distribution -> distr6::SDistribution -> Empirical

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{Empirical$new()}
- \texttt{Empirical$mean()}
- \texttt{Empirical$mode()}
- \texttt{Empirical$variance()}
- \texttt{Empirical$skewness()}
Method `new()`: Creates a new instance of this R6 class.

Usage:
`Empirical$new(samples = 1, decorators = 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

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

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

Usage:
Empirical

Empirical$variance(...)

Arguments:
... Unused.

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

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

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:
Empirical$skewness(...)

Arguments:
... Unused.

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

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

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:
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

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

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

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

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

Method cf(): The characteristic function is defined by
\[ cf_X(t) = E_X[exp(\pi t)] \]
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
\[ pgf_X(z) = E_X[exp(z^t)] \]
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")

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".

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,

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

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

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

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).
EmpiricalMV

Value
Returns an R6 object inheriting from class SDistribution.

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 = data.frame(1, 1), 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

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

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

Usage:
EmpiricalMV$variance(...)

Arguments:
... Unused.

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

Usage:
EmpiricalMV$setParameterValue(..., lst = NULL, error = "warn")

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".

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))
```
**entropy**  
*Distribution Entropy*

**Description**

(Information) Entropy of a distribution

**Usage**

```r
entropy(object, base = 2, ...)
```

**Arguments**

- `object` Distribution.
- `base` base of the entropy logarithm, default = 2 (Shannon entropy)
- `...` Passed to `genExp`.

**Value**

Entropy with given base as a numeric.

---

**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**

```
distr6:: Distribution -> distr6:: Kernel -> Epanechnikov
```

**Public fields**

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

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

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

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

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

**Arguments:**
- ... Unused.
**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**

---

**Erlang**

**Erlang Distribution Class**

**Description**
Mathematical and statistical functions for the Erlang distribution, which is commonly used as a special case of the Gamma distribution when the shape parameter is an integer.

**Details**
The Erlang distribution parameterised with shape, \(\alpha\), and rate, \(\beta\), is defined by the pdf,

\[
f(x) = (\beta^{\alpha})(x^{\alpha-1})(\exp(-x\beta))/(\alpha - 1)!
\]

for \(\alpha = 1, 2, 3, \ldots\) and \(\beta > 0\).
The distribution is supported on the Positive Reals.

**Value**
Returns an R6 object inheriting from class SDistribution.

**Super classes**
distr6::Distribution -> distr6::SDistribution -> 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$setParameterValue()`
- `Erlang$clone()`

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

**Usage:**

```
Erlang$new(shape = 1, rate = 1, 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. `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

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

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

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

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

Usage:
Erlang$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:
Erlang$setParameterValue(..., lst = NULL, error = "warn")
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".

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

Usage:
Erlang$clone(deep = FALSE)

Arguments:
  deep  Whether to make a deep clone.

References


See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, 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, ChiSquared, Degenerate, DiscreteUniform, Empirical, 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

---------------------------------------------------------------------

exkurtosisType   Kurtosis Type

---------------------------------------------------------------------

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'.

See Also

kurtosis, skewType

Examples

exkurtosisType(-1)
exkurtosisType(0)
exkurtosisType(1)

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 \( F_X \) 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

\[
s(a, b) = \int_a^b S_X(x) dx
\]

where \( S_X \) 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.
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, \( \lambda \), is defined by the pdf,
\[ f(x) = \lambda e^{-x\lambda} \]
for \( \lambda > 0 \).
The distribution is supported on the Positive Reals.
Exponential

Value
Returns an R6 object inheriting from class SDistribution.

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$pgf()
• Exponential$setParameterValue()
• Exponential$clone()

Method new(): Creates a new instance of this R6 class.
Usage:
Exponential$new(rate = 1, 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

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

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 available 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

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

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

Usage:
Exponential$variance(...)

Arguments:
... Unused.

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

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

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:
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, ...)

*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:*
Exponential$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:
Exponential$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:
Exponential$setParameterValue(..., lst = NULL, error = "warn")

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".

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

Usage:
Exponential$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, 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,

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

for \( \mu, \nu > 0 \).

The distribution is supported on the Positive Reals.

**Value**

Returns an R6 object inheriting from class SDistribution.

**Super classes**

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.
Methods

Public methods:

• `FDistribution$new()`
• `FDistribution$mean()`
• `FDistribution$mode()`
• `FDistribution$variance()`
• `FDistribution$skewness()`
• `FDistribution$kurtosis()`
• `FDistribution$entropy()`
• `FDistribution$mgf()`
• `FDistribution$pgf()`
• `FDistribution$setParameterValue()`
• `FDistribution$clone()`

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

Usage:
`FDistribution$new(df1 = 1, df2 = 1, 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

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

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

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

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

Usage:
FDistribution$variance(...)

Arguments:
... Unused.

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

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

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:
FDistribution$skewness(...)

Arguments:
... Unused.

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

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

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

Usage:
FDistribution$setParameterValue(..., lst = NULL, error = "warn")

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".

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.
FDistributionNoncentral

References


See Also

Other continuous distributions: Arcsine, BetaNoncentral, Beta, Cauchy, ChiSquaredNoncentral, ChiSquared, Dirichlet, Erlang, Exponential, FDistributionNoncentral, 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, FDistributionNoncentral, 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

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,

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

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

The distribution is supported on the Positive Reals.

Value

Returns an R6 object inheriting from class SDistribution.

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.

Methods

Public methods:

- `FDistributionNoncentral$new()`
- `FDistributionNoncentral$mean()`
- `FDistributionNoncentral$variance()`
- `FDistributionNoncentral$setParameterValue()`
- `FDistributionNoncentral$clone()`

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

Usage:

```r
FDistributionNoncentral$new(df1 = 1, df2 = 1, location = 0, 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

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

with an integration analogue for continuous distributions.

Usage:

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

**Usage:**

```r
FDistributionNoncentral$setParameterValue(..., lst = NULL, error = "warn")
```

**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".

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

**Usage:**

```r
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

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,

$$f(x) = \frac{\alpha}{\beta} \left(\frac{x - \gamma}{\beta}\right)^{-1-\alpha} \exp\left(-\frac{x - \gamma}{\beta}\right)^{-\alpha}$$

for $\alpha, \beta \in R^+$ and $\gamma \in R$.

The distribution is supported on $x > \gamma$.

Also known as the Inverse Weibull distribution.

Value

Returns an R6 object inheriting from class SDistribution.

Super classes

distr6::Distribution -> distr6::SDistribution -> 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.

Methods

Public methods:

- Frechet$new()
- Frechet$mean()
- Frechet$mode()
- Frechet$median()
- Frechet$variance()
- Frechet$skewness()
- Frechet$kurtosis()
- Frechet$entropy()
- Frechet$pgf()
• Frechet$setParameterValue()
• Frechet$clone()

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

*Usage:*
Frechet$new(shape = 1, scale = 1, minimum = 0, 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

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

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

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

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,

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

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:
Frechet$skewness(...)

Arguments:
... Unused.

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

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

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:
Frechet$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:
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

$$pg_f_X(z) = E_X [exp(z^x)]$$

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

Usage:
Frechet$setParameterValue(..., lst = NULL, error = "warn")

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".

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

Michael P. McLaughlin.

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
FunctionImputation  

**Imputed Pdf/Cdf/Quantile/Rand Functions Decorator**

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

**See Also**
Other decorators: CoreStatistics, ExoticStatistics

**Examples**

```r
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()
  )
nest(x, "FunctionImputation", n = 1000)

  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)
}
```
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 $\alpha, \beta > 0$.

The distribution is supported on the Positive Reals.

Value

Returns an R6 object inheriting from class SDistribution.

Super classes

distr6::Distribution $\rightarrow$ distr6::SDistribution $\rightarrow$ 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:

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

Method new(): Creates a new instance of this R6 class.
Usage:
Gamma$new(shape = 1, rate = 1, scale = NULL, mean = NULL, decorators = NULL)

Arguments:
shape (numeric(1))
   Shape parameter, defined on the positive Reals.
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.
mean (numeric(1))
   Alternative parameterisation of the distribution, defined on the positive Reals. If given then rate and scale are ignored. Related by mean = shape/rate.
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:
Gamma$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:
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

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

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,

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

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:
Gamma$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:
Gamma$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:
Gamma$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:
Gamma$mgf(t, ...)

Arguments:
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
Gamma$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:
```r
Gamma$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
Gamma$setParameterValue(..., lst = NULL, error = "warn")
```

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".

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

Usage:
```r
Gamma$clone(deep = FALSE)
```

Arguments:
- `deep` Whether to make a deep clone.
generalPNorm

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

Description

Calculate the p-norm of any function between given limits.

Usage

generalPNorm(fun, p, lower, upper, range = 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

\[
\sum_i (x_{i+1} - x_i) \ast |f(x_i)|^p
\]
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
genExp$new()$pdf, 2, 0, 10)
```

---

**Description**

A generalised expectation function for distributions, for arithmetic mean and more complex numeric calculations.

**Usage**

```r
genExp(object, trafo = NULL, cubature = FALSE, ...)
```

**Arguments**

- `object` Distribution.
- `trafo` transformation for expectation calculation, see details.
- `cubature` If TRUE uses `cubature::cubintegrate` for approximation, otherwise `integrate`
- `...` Passed to `cubature::cubintegrate`.

**Value**

The given expectation as a numeric, otherwise NULL.
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,

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

for probability \( p \).

The distribution is supported on the Naturals (zero is included if modelling number of failures before success).

The Geometric distribution is used to either refer to modelling the number of trials or number of failures before the first success.

Value

Returns an R6 object inheriting from class SDistribution.

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

Usage:
```
Geometric$new(prob = 0.5, qprob = NULL, trials = FALSE, 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

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

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

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

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

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.

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.

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, ...)

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:
Geometric$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:
Geometric$setParameterValue(..., lst = NULL, error = "warn")

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".

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

Usage:
Geometric$clone(deep = FALSE)

Arguments:
depth Whether to make a deep clone.
**getParameterSupport**

**Parameter Support Accessor**

**Description**

Returns the support of the given parameter.

**Usage**

```r
getParameterSupport(object, id, error = "warn")
```

**Arguments**

- **object**: Distribution or ParameterSet.
- **id**: character, id of the parameter to return.
- **error**: character, value to pass to stopwarn.

**Value**

An R6 object of class inheriting from `set6::Set`

**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
getParameterValue  \[ \text{Parameter Value Accessor} \]

**Description**

Returns the value of the given parameter.

**Usage**

```r
getParameterValue(object, id, error = "warn")
```

**Arguments**

- **object**: Distribution or ParameterSet.
- **id**: character, id of the parameter to return.
- **error**: character, value to pass to `stopwarn`.

**Value**

The current value of a given parameter as a numeric.

---

**Gompertz**  \[ \text{Gompertz Distribution Class} \]

**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,

\[
f(x) = \alpha \beta \exp(x\beta) \exp(\alpha) \exp(-\exp(x\beta)\alpha)
\]

for \(\alpha, \beta > 0\).

The distribution is supported on the Non-Negative Reals.

**Value**

Returns an R6 object inheriting from class SDistribution.

**Super classes**

```r
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 = 1, scale = 1, 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 \( \beta > 0 \).

The distribution is supported on the Reals.

Value

Returns an R6 object inheriting from class SDistribution.

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 = 0, scale = 1, 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,

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

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.
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

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

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

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

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

**Usage:**
Gumbel\$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:
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
Michael P. McLaughlin.

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

hazard

Hazard Function

Description
See ExoticStatistics$hazard.

Usage
hazard(object, ..., log = FALSE, simplify = TRUE, data = NULL)
huberize

Arguments

object (Distribution).

... (numeric())

Points to evaluate the probability density function of the distribution. 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 log-Hazard Default is FALSE.

simplify logical(1)

If TRUE (default) simplifies the pdf 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.

Value

Hazard function as a numeric, natural logarithm returned if log is TRUE.

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

\[ f_H(x) = \begin{cases} F(x), & if \ x \leq lower \\ f(x), & if lower < x < upper \\ F(x), & if \ x \geq upper \end{cases} \]

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

distr6::Distribution -> distr6::DistributionWrapper -> HuberizedDistribution

Methods

Public methods:

• HuberizedDistribution$new()
• HuberizedDistribution$setParameterValue()
• 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 setParameterValue(): Sets the value(s) of the given parameter(s).
Usage:
HuberizedDistribution$setParameterValue(..., lst = NULL, error = "warn")
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".

Method clone(): The objects of this class are cloneable with this method.
Usage:
HuberizedDistribution$clone(deep = FALSE)
Arguments:
deep Whether to make a deep clone.

See Also
Other wrappers: Convolution, DistributionWrapper, MixtureDistribution, ProductDistribution, TruncatedDistribution, VectorDistribution

Examples

```r
# ------------------------------------------------
## Method \texttt{HuberizedDistribution}\$\texttt{new}
## ------------------------------------------------
HuberizedDistribution$new(
  Binomial$new(prob = 0.5, size = 10),
  lower = 2, upper = 4
)

# alternate constructor
huberize(Binomial$new(), lower = 2, upper = 4)
```
Hypergeometric

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,

\[
f(x) = \frac{C(K, x)C(N - K, n - x)}{C(N, n)}
\]

for \( N = \{0, 1, 2, \ldots, N\} \), \( n, K = \{0, 1, 2, \ldots, N\} \) and \( C(a, b) \) is the combination (or binomial coefficient) function.

The distribution is supported on \( \{\max(0, n + K - N), \ldots, \min(n, K)\} \).

Value

Returns an R6 object inheriting from class SDistribution.

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.

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:*

```r
Hypergeometric$new(
  size = 50,
  successes = 5,
  failures = NULL,
  draws = 10,
  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())
  Decorators to add to the distribution during construction.

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

$$E_X(X) = \sum_{x} p_X(x) * x$$

with an integration analogue for continuous distributions.

*Usage:*

```r
Hypergeometric$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
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

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

where $E_X$ is the expectation of distribution $X$. If the distribution is multivariate the covariance matrix is returned.
Usage:
Hypergeometric$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:
Hypergeometric$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:
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 = NULL, error = "warn")

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".

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

Usage:
Hypergeometric$clone(deep = FALSE)

Arguments:
depth 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

---

### inf

**Infimum Accessor**

**Description**

Returns the distribution infimum as the infimum of the support.

**Usage**

`inf(object)`

**Arguments**

- `object` Distribution.

**Value**

Infimum as a numeric.

**R6 Usage**

`$inf`
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,

$$f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{-\alpha-1} \exp\left(-\frac{\beta}{x}\right)$$

for $\alpha, \beta > 0$, where $\Gamma$ is the gamma function.
The distribution is supported on the Positive Reals.

Value
Returns an R6 object inheriting from class SDistribution.

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()
• \texttt{InverseGamma}\$\texttt{clone()}

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

*Usage:*

\texttt{InverseGamma}\$\texttt{new(shape = 1, scale = 1, decorators = NULL)}

*Arguments:*

- \texttt{shape (numeric(1))}
  
  Shape parameter, defined on the positive Reals.

- \texttt{scale (numeric(1))}
  
  Scale parameter, defined on the positive Reals.

- \texttt{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:*

\texttt{InverseGamma}\$\texttt{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:*

\texttt{InverseGamma}\$\texttt{mode(which = "all")}

*Arguments:*

- \texttt{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:*

\texttt{InverseGamma}\$\texttt{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:*

InverseGamma$skewness(...) 

*Arguments:*

... Unused.

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

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

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:*

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:*

**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**:

InverseGamma$pgf(z, ...)

**Arguments**:

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

**Methods**

**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**:

InverseGamma$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**:

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
**iqr**

*Distribution Interquartile Range*

---

**Description**

Interquartile range of a distribution

**Usage**

```r
iqr(object)
```

**Arguments**

- `object`: Distribution.

**Value**

Interquartile range of distribution as a numeric.

---

**Kernel**

*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.

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:
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.

Usage:
Kernel$cdfSquared2Norm(x = 0, upper = Inf)

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

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:
Kernel$skewness(...) 

Arguments:
... Unused.

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

Usage:
Kernel$clone(deep = FALSE)

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

### Description

Kth standardised or central moment of a distribution

### Usage

```r
kthmoment(object, k, type = c("central", "standard", "raw"), ...)
```

### Arguments

- **object**: Distribution.
- **k**: the kth moment to calculate
- **type**: one of 'central', 'standard' or 'raw', abbreviations allowed
- **...**: Passed to $genExp.

### Value

If univariate, the given k-moment as a numeric, otherwise NULL.

---

**kurtosis**  

### Description

Kurtosis of a distribution

### Usage

```r
kurtosis(object, excess = TRUE, ...)```

### Arguments

- **object**: Distribution.
- **excess**: logical, if TRUE (default) excess Kurtosis returned.
- **...**: Passed to $genExp.

### Value

Kurtosis as a numeric.
kurtosisType: Type of Kurtosis Accessor - Deprecated

Description

Deprecated. Use $properties$kurtosis.

Usage

kurtosisType(object)

Arguments

object: Distribution.

Value

If the distribution kurtosis is present in properties, returns one of "platykurtic"/"mesokurtic"/"leptokurtic". otherwise returns NULL.

Laplace: 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, μ, and scale, β, is defined by the pdf,

\[ f(x) = \exp\left(-|x - \mu|/\beta\right)/(2\beta) \]

for \( \mu \epsilon R \) and \( \beta > 0 \).

The distribution is supported on the Reals.

Value

Returns an R6 object inheriting from class SDistribution.

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$setParameterValue()`  
- `Laplace$clone()`  

**Method** `new()`:

*Creates a new instance of this R6 class.*

**Usage:**

```
Laplace$new(mean = 0, scale = 1, var = 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.
- `var` (numeric(1))  
  Variance of the distribution, defined on the positive Reals. `var = 2*scale^2`. If `var` is provided 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:**

```
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,

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

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

Usage:
Laplace$setParameterValue(..., lst = NULL, error = "warn")

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".

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

Usage:
Laplace$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, 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
liesInSupport  
Test if Data Lies in Distribution Support

Description
Tests if the given data lies in the support of the Distribution, either tests if all data lies in the support or any of it.

Usage
liesInSupport(object, x, all = TRUE, bound = FALSE)

Arguments
- object: Distribution.
- x: vector of numerics to test.
- all: logical, see details.
- bound: logical, if FALSE (default) uses dmin/dmax otherwise inf/sup.

Value
Either a vector of logicals if all is FALSE otherwise returns TRUE if every element lies in the distribution support or FALSE otherwise.

liesInType  
Test if Data Lies in Distribution Type

Description
Tests if the given data lies in the type of the Distribution, either tests if all data lies in the type or any of it.

Usage
liesInType(object, x, all = TRUE, bound = FALSE)

Arguments
- object: Distribution.
- x: vector of numerics to test.
- all: logical, see details.
- bound: logical, if FALSE (default) uses dmin/dmax otherwise inf/sup.

Value
Either a vector of logicals if all is FALSE otherwise returns TRUE if every element lies in the distribution type or FALSE otherwise.
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
## S3 method for class 'Distribution'
lines(x, fun, npoints = 3000, ...)

Arguments
x   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")
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.

See Also

DistributionDecorator

Examples

listDecorators()
listDecorators(FALSE)
listDistributions  Lists Implemented Distributions

Description

Lists distr6 distributions in a data.table or a character vector, can be filtered by traits, implemented package, and tags.

Usage

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

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 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,

$$f(x) = -\theta^x / x\log(1 - \theta)$$

for $0 < \theta < 1$.
The distribution is supported on 1, 2, 3, ....

Value
Returns an R6 object inheriting from class SDistribution.

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 = 0.5, 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

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

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")
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:
Logarithmic$variance(...)

Arguments:
... Unused.

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

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

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:
Logarithmic$skewness(...)

Arguments:
... Unused.

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

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

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:
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

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

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

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

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

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 $\mu \in \mathbb{R}$ and $s > 0$.
The distribution is supported on the Reals.

Value
Returns an R6 object inheriting from class SDistribution.

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$setParameterValue()
• Logistic$clone()

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

*Usage:*

```r
Logistic$new(mean = 0, scale = 1, 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, $sd = scale \times pi / sqrt(3)$. 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

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

with an integration analogue for continuous distributions.

*Usage:*

```r
Logistic$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
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

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

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

*Usage:*

```r
Logistic$variance(...)
```

*Arguments:*

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:
Logistic$skewness(...)

Arguments:
... Unused.

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

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

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:
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

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

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.
... Unused.

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

Usage:
Logistic$setParameterValue(..., lst = NULL, error = "warn")

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".

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,

\[ f(x) = \left(\exp(x) + 2 + \exp(-x)\right)^{-1} \]

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.

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(...)
Loglogistic

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
Methods

Public methods:

• Loglogistic$new()
• Loglogistic$mean()
• Loglogistic$mode()
• Loglogistic$median()
• Loglogistic$variance()
• Loglogistic$skewness()
• Loglogistic$kurtosis()
• Loglogistic$pgf()
• Loglogistic$setParameterValue()
• Loglogistic$clone()

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

Usage:
Loglogistic$new(scale = 1, shape = 1, 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.

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:
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

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

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

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

**Arguments:**
... Unused.

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

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

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:**
Loglogistic$skewness(...)

**Arguments:**
... Unused.

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

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

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

Usage:
Loglogistic$setParameterValue(..., lst = NULL, error = "warn")

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".

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
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, 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, Logistic, Lognormal, NegativeBinomial,
Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT,
Triangular, Uniform, Wald, Weibull, WeightedDiscrete
Lognormal 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,

\[
\text{exp}\left(-\frac{(\log(x) - \mu)^2}{2\sigma^2}\right) / (x\sigma \sqrt{2\pi})
\]

for \( \mu \in \mathbb{R} \) and \( \sigma > 0 \).

The distribution is supported on the Positive Reals.

Also known as the Log-Gaussian distribution.

Value

Returns an R6 object inheriting from class SDistribution.

Super classes

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

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{Lognormal$new()}
- \texttt{Lognormal$mean()}
- \texttt{Lognormal$mode()}
- \texttt{Lognormal$median()}
- \texttt{Lognormal$variance()}
- \texttt{Lognormal$skewness()}
- \texttt{Lognormal$kurtosis()}
- \texttt{Lognormal$entropy()}
- \texttt{Lognormal$mgf()}

Lognormal

- Lognormal\$pgf()
- Lognormal\$setParameterValue()
- Lognormal\$clone()

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

**Usage:**

```r
Lognormal$new(
  meanlog = 0,
  varlog = 1,
  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.
  
  \[
  sdlog = varlog^2
  \]

  - 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.
  
  \[
  preclog = 1/varlog
  \]

  - 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.
  
  \[
  var = (\exp(var) - 1) * \exp(2 * meanlog + varlog)
  \]

- `sd` (numeric(1))
  Standard deviation of the distribution on the natural scale, defined on the positive Reals.
  
  \[
  sd = var^2
  \]

  - 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.

  \[ prec = 1/var \]

  If given then all other parameters except mean are ignored.

decorators (character(1))
  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

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

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,
\[ sk_X = E_X \left[ \frac{x - \mu^3}{\sigma} \right] \]
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:
Lognormal$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised moment,
\[ k_X = E_X \left[ \frac{x - \mu^4}{\sigma} \right] \]
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:
Lognormal$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:
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 setParameterValue(): Sets the value(s) of the given parameter(s).

Usage:
Lognormal$setParameterValue(..., lst = NULL, error = "warn")

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".

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.

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, MultivariateNormal,
makeUniqueDistributions

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

## ------------------------------------------------
## 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
makeUniqueDistributions(list(Binomial$new(), Binomial$new()))
mean.Distribution  Distribution Mean

Description

Arithmetic mean for the probability distribution.

Usage

```r
## S3 method for class 'Distribution'
mean(x, ...)
```

Arguments

- `x`  Distribution.
- `...`  Passed to $genExp.

Value

Mean as a numeric.

median.Distribution  Median of a Distribution

Description

Median of a distribution assuming quantile is provided.

Usage

```r
## S3 method for class 'Distribution'
median(x, na.rm = NULL, ...)
```

Arguments

- `x`  Distribution.
- `na.rm`  ignored, added for consistency with S3 generic.
- `...`  ignored, added for consistency with S3 generic.

Value

Quantile function evaluated at 0.5 as a numeric.
merge.ParameterSet  

Description

Combine ParameterSets

Usage

```r
## S3 method for class 'ParameterSet'
merge(x, y, ...)
```

Arguments

- `x`: ParameterSet
- `y`: ParameterSet
- `...`: ParameterSets

Value

An R6 object of class ParameterSet.

mgf  

Moment Generating Function

Description

Moment generating function of a distribution

Usage

```r
mgf(object, t, ...)
```

Arguments

- `object`: Distribution.
- `t`: integer to evaluate moment generating function at.
- `...`: Passed to $genExp.

Value

Moment generating function evaluated at `t` as a numeric.
**MixtureDistribution**  
*Mixture Distribution Wrapper*

**Description**

Wrapper used to construct a mixture of two or more distributions.

**Details**

A mixture distribution is defined by

\[
F_P(x) = w_1 F_{X_1}(x) \times \ldots \times w_n F_{X_N}(x)
\]

#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**

\[\text{distr6::Distribution} \rightarrow \text{distr6::DistributionWrapper} \rightarrow \text{distr6::VectorDistribution} \rightarrow \text{MixtureDistribution}\]

**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:*

```r
MixtureDistribution$new(  
  distlist = NULL,  
  weights = "uniform",  
  distribution = NULL,  
  params = NULL,  
  shared_params = NULL,  
  name = NULL,  
  short_name = NULL,  
  decorators = NULL,  
  vecdist = 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.

Method vecdist VectorDistribution

Alternative constructor to directly create this object from an object inheriting from VectorDistribution.

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:**

```r
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:**

```r
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:**

```r
MixtureDistribution$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. @examples m <- Mix-
tureDistribution$new(list(Binomial$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 evalu-
  ate. 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:
MixtureDistribution$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 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.

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$rnd(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.

Examples:

m <- MixtureDistribution$new(distribution = "Normal",
params = data.table::data.table(mean = 1:2), shared_params = list(sd = 1))
m$rnd(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 MixtureDistribution$new
MixtureDistribution$new(list(Binomial$new(prob = 0.5, size = 10), Binomial$new()),
   weights = c(0.2, 0.8))
```

```r
# Method MixtureDistribution$pdf
m <- MixtureDistribution$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)
```

```r
# Method MixtureDistribution$rnd
```
mixturiseVector

m <- MixtureDistribution$new(distribution = "Normal",
params = data.table::data.table(mean = 1:2), shared_params = list(sd = 1))
m$rand(5)

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 \(D_{3N} = m(D_{1N}, D_{2N}, wN)\) where \(m\) is a mixture distribution with weights \(wN\).

Examples
## 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)


Mode of a Distribution

Description

A numeric search for the mode(s) of a distribution.

Usage

mode(object, which = "all")

Arguments

object Distribution.
which which mode of the distribution should be returned, default is all.

Details

If the distribution has multiple modes, all are returned by default. Otherwise the index of the mode
to return can be given or "all" if all should be returned.

If an analytic expression isn’t available, returns error. To impute a numerical expression, use the
CoreStatistics decorator.

Value

The estimated mode as a numeric, either all modes (if multiple) or the ordered mode given in which.

See Also

CoreStatistics and decorate.

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, \( p_1, ..., p_k \), is defined by the pmf,

\[
f(x_1, x_2, ..., x_k) = \frac{n!}{x_1! \cdot x_2! \cdot ... \cdot x_k!} \cdot p_1^{x_1} \cdot p_2^{x_2} \cdot ... \cdot p_k^{x_k}
\]

for \( p_i, i = 1, ..., k; \sum p_i = 1 \) and \( n = 1, 2, ... \).

The distribution is supported on \( \sum x_i = N \).

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

Value

Returns an R6 object inheriting from class SDistribution.

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.

Methods

Public methods:

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

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

Usage:

Multinomial$new(size = 10, probs = c(0.5, 0.5), decorators = NULL)

Arguments:
size (integer(1))
   Number of trials, defined on the positive Naturals.
probs (numeric())
   Vector of probabilities. Automatically normalised by probs = probs/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

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

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,

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

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,

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

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

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

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

$$pgf_X(z) = E_X[exp(zx)]$$

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

Usage:
Multinomial$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

References

Michael P. McLaughlin.

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,

$$f(x_1, \ldots, x_k) = (2\pi)^{-k/2}|\Sigma|^{-1/2}exp\left(-1/2(x - \mu)^T\Sigma^{-1}(x - \mu)\right)$$

for $\mu \in \mathbb{R}^k$ and $\Sigma \in \mathbb{R}^{k \times k}$.

The distribution is supported on the Reals and only when the covariance matrix is positive-definite. cdf and quantile are omitted as no closed form analytic expression could be found, decorate with FunctionImputation for a numerical imputation.

Sampling is performed via the Cholesky decomposition using chol.

Number of variables cannot be changed after construction.
Value

Returns an R6 object inheriting from class SDistribution.

Super classes

distr6::Distribution -> distr6::SDistribution -> MultivariateNormal

Public fields

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

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:
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

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

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

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

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

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

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.
Method setParameterValue(): Sets the value(s) of the given parameter(s).
   Usage:
   MultivariateNormal$setParameterValue(..., lst = NULL, error = "warn")
   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".

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

---

NegativeBinomial

Negative Binomial Distribution Class

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.
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 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).

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.

Note that the size parameter is not currently vectorised in VectorDistributions.

Value

Returns an R6 object inheriting from class SDistribution.

Super classes

`distr6::Distribution -> distr6::SDistribution -> 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.

Methods

Public methods:

- NegativeBinomial$new()
- NegativeBinomial$mean()
- NegativeBinomial$mode()
- NegativeBinomial$variance()
- NegativeBinomial$skewness()
- NegativeBinomial$kurtosis()
- NegativeBinomial$mgf()
- NegativeBinomial$cf()
- NegativeBinomial$pgf()
- NegativeBinomial$setParameterValue()
- NegativeBinomial$clone()

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

*Usage:*

```r
NegativeBinomial$new(
  size = 10,
  prob = 0.5,
  qprob = NULL,
  mean = NULL,
  form = c("fbs", "sbf", "tbf", "tbs"),
  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

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

with an integration analogue for continuous distributions.

*Usage:*

```r
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

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

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

Usage:
NegativeBinomial$variance(\ldots)$

Arguments:
\ldots 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:
NegativeBinomial$skewness(\ldots)$

Arguments:
\ldots 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:
NegativeBinomial$kurtosis(excess = TRUE, \ldots)$

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

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

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

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

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

Usage:
NegativeBinomial$setParameterValue(..., lst = NULL, error = "warn")

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".

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 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,

$$f(x) = \frac{\exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma^2}}$$

for $\mu \in \mathbb{R}$ and $\sigma^2 > 0$.

The distribution is supported on the Reals.

Also known as the Gaussian distribution.

Value

Returns an R6 object inheriting from class SDistribution.

Super classes

```
  distr6::Distribution -> distr6::SDistribution -> 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.

References


See Also

Other discrete distributions: Bernoulli, Binomial, Categorical, Degenerate, DiscreteUniform, EmpiricalMV, Empirical, 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, ShiftedLogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete
Methods

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

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

Usage:
Normal$new(mean = 0, var = 1, 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(var). If provided then var ignored.
prec (numeric(1))
Precision of the distribution, defined on the positive Reals. prec = 1/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

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

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

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

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

Usage:
Normal$variance(...)

Arguments:
... Unused.

Method skewness(): The skewness of a distribution is defined by the third standardised mo-
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.

Usage:
Normal$skewness(...)

Arguments:
... Unused.

Method kurtosis(): The kurtosis of a distribution is defined by the fourth standardised mo-
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.

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

$$pgf_X(z) = E_X[exp(zt)]$$

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.
... Unused.

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

Usage:
Normal$setParameterValue(..., lst = NULL, error = "warn")
NormalKernel

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".

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

Usage:
Normal$clone(deep = FALSE)

Arguments:
depth 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, 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, Logistic, Loglogistic, Lognormal, NegativeBinomial, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

NormalKernel

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

distr6::Distribution -> distr6::Kernel -> NormalKernel

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:

• NormalKernel$new()
• NormalKernel$pdfSquared2Norm()
• NormalKernel$variance()
• NormalKernel$clone()

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

Usage:
NormalKernel$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:
NormalKernel$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:*

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*

---

### parameters

#### Parameters Accessor

**Description**

Returns some or all the parameters in a distribution.

**Usage**

parameters(object, id = NULL)

**Arguments**

- **object**: Distribution or ParameterSet.
- **id**: character, see details.

**Value**

An R6 object of class ParameterSet or a data.table.
ParameterSets are passed to the `Distribution` constructor when creating a custom probability distribution that takes parameters.

### Active bindings
- `deps` Returns ParameterSet dependencies table.
- `checks` Returns ParameterSet assertions table.
- `trasfos` Returns ParameterSet transformations table.
- `length` Number of parameters in ParameterSet.

### Methods

#### Public methods:
- `ParameterSet$new()`
- `ParameterSet$print()`
- `ParameterSet$parameters()`
- `ParameterSet$getParameterSupport()`
- `ParameterSet$getParameterValue()`
- `ParameterSet$setParameterValue()`
- `ParameterSet$merge()`
- `ParameterSet$addDeps()`
- `ParameterSet$addChecks()`
- `ParameterSet$addTrafos()`
- `ParameterSet$values()`
- `ParameterSet$clone()`

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

**Usage:**
```r
ParameterSet$new(  
  id,  
  value,  
  support,  
  settable = TRUE,  
  updateFunc = NULL,  
  description = NULL  
)
```

**Arguments:**
id (character(1)|list())
   id of the parameter(s) to construct, should be unique.
value (ANY|list())
   Value of parameter(s) to set.
support ([set6::Set]|list())
   Support of parameter(s) to set
settable (character(1)|list())
   Logical flag indicating if the parameter(s) can be updated after construction.
updateFunc (list())
   Deprecated, please use $addDeps instead.
description (character(1)|list())
   Optional description for the parameter(s).

Details: Every argument can either be given as the type listed or as a list of that type. If arguments are provided as a list, then each argument must be of the same length, with values as NULL where appropriate. See examples for more.

Examples:
id <- list("prob", "size")
value <- list(0.2, 5)
support <- list(set6::Interval$new(0, 1), set6::PosNaturals$new())
description <- list("Probability of success", NULL)
ParameterSet$new(id = id,
   value = value,
   support = support,
   description = description
)

ParameterSet$new(id = "prob",
   value = 0.2,
   support = set6::Interval$new(0, 1),
   description = "Probability of success"
)

Method print(): Prints the ParameterSet.

Usage:
ParameterSet$print(hide_cols = c("settable"), ...)

Arguments:
hide_cols (character())
   Names of columns in the ParameterSet to hide whilst printing.
...
   ANY
   Additional arguments, currently unused.

Method parameters(): Returns the full parameter details for the supplied parameter, or returns self if id is NULL.

Usage:
ParameterSet$parameters(id = NULL)
Arguments:
id character()
   id of parameter to return.

Method getParameterSupport(): Returns the support of the supplied parameter.

Usage:
ParameterSet$getParameterSupport(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".

Returns: A set6::Set object.

Examples:
ps <- ParameterSet$new(id = "prob",
   value = 0.2,
   support = set6::Interval$new(0, 1),
   settable = TRUE,
   description = "Probability of success"
)
ps$getParameterSupport("prob")

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

Usage:
ParameterSet$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".

Examples:
ps <- ParameterSet$new(id = "prob",
   value = 0.2,
   support = set6::Interval$new(0, 1),
   settable = TRUE,
   description = "Probability of success"
)
ps$getParameterValue("prob")

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

Usage:
ParameterSet$setParameterValue(
   ...,
   lst = NULL,
   error = "warn",
   .suppressCheck = 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".
.suppressCheck (logical(1))
  Should be set internally only.

Examples:
id <- list("rate")
value <- list(1)
support <- list(set6::PosReals$new())
ps <- ParameterSet$new(
  id, value, support
)
ps$setParameterValue(rate = 2)
ps$getParameterValue("rate")

Method merge(): Merges multiple parameter sets.
Usage:
ParameterSet$merge(y, ...)
Arguments:
y ([ParameterSet])
... ([ParameterSet]s)
Examples:
\dontrun{
  ps1 <- ParameterSet$new(id = c("prob", "qprob"),
    value = c(0.2, 0.8),
    support = list(set6::Interval$new(0, 1), set6::Interval$new(0, 1))
  )
  ps1$addChecks(function(self) self$getParameterValue("qprob") > 0)
  ps1$addDeps("prob", "qprob", function(self)
    list(qprob = 1 - self$getParameterValue("prob")))
  ps2 <- ParameterSet$new(id = "size",
    value = 10,
    support = set6::Interval$new(0, 10, class = "integer"),
  )
  ps2$addTrafos("size", function(x, self) x + 1)
  ps1$merge(ps2)
  ps1$print()
}

Method addDeps(): Add parameter dependencies for automatic updating.
Usage:
ParameterSet$addDeps(x, y, fun)

Arguments:
x (character(1))
   id of parameter that updates y.
y (character())
   id of parameter(s) that is/are updated by x.
fun (function(1))
   Function used to update y, must include self in formal arguments and should return a named list with names identical to, and in the same order, as y.

Examples:
\dontrun{
  ps <- ParameterSet$new(
    id = list("a", "b", "c"),
    value = list(2, 3, 1/2),
    support = list(set6::Reals$new(), set6::Reals$new(), set6::Reals$new())
  )
  ps$addDeps("a", c("b", "c"),
    function(self) {
      list(b = self$getParameterValue("a") + 1,
           c = 1/self$getParameterValue("a"))
    }
  )
}

Method addChecks(): Add parameter checks for automatic assertions. Note checks are made after any transformations.

Usage:
ParameterSet$addChecks(fun)

Arguments:
fun (function(1))
   Function used to check ParameterSet, must include self in formal arguments and result in a logical.

Examples:
\dontrun{
  id <- list("lower", "upper")
  value <- list(1, 3)
  support <- list(set6::PosReals$new(), set6::PosReals$new())
  ps <- ParameterSet$new(
    id, value, support
  )
  ps$addChecks(function(self)
    self$getParameterValue("lower") < self$getParameterValue("upper")
  )
}

Method addTrafos(): Transformations to apply to parameter before setting. Note transformations are made before checks. NOTE: If a transformation for a parameter already exists then this will be overwritten.
**Usage:**
ParameterSet$addTrafos(x, fun, dt = NULL)

**Arguments:**
- `x` (character(1))
  - id of parameter to be transformed. Only one trafo function per parameter allowed - though multiple transformations can be encoded within this.
- `fun` (function(1))
  - Function used to transform `x`, must include `x`, `self` in formal arguments and `x` in body where `x` is the value of the parameter to check. See first example.
- `dt` ([data.table::data.table])
  - Alternate method to directly construct data.table of transformations to add. See second example.

**Examples:**
```r
\dontrun{
  ps <- ParameterSet$new(
    "probs", list(c(1, 1)), set6::Interval$new(0,1)^2
  )
  ps$addTrafos("probs", function(x, self) return(x / sum(x)))
  ps$trafos
  ps$setParameterValue(probs = c(1, 2))
  ps$getParameterValue("probs")

  # Alternate method (better with more parameters)
  ps <- ParameterSet$new(
    "probs", list(c(1, 1)), set6::Interval$new(0,1)^2
  )
  ps$addTrafos(dt = data.table::data.table(
    x = "probs",
    fun = function(x, self) return(x / sum(x))
  ))
}
```

**Method** values(): Returns parameter set values as a named list.

**Usage:**
ParameterSet$values(settable = TRUE)

**Arguments:**
- `settable` (logical(1))
  - If TRUE (default) only returns values of settable parameters, otherwise returns all.

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

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

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

```r
# Method `ParameterSet$new`

id <- list("prob", "size")
value <- list(0.2, 5)
support <- list(set6::Interval$new(0, 1), set6::PosNaturals$new())
description <- list("Probability of success", NULL)
ParameterSet$new(id = id,
    value = value,
    support = support,
    description = description)

ParameterSet$new(id = "prob",
    value = 0.2,
    support = set6::Interval$new(0, 1),
    description = "Probability of success")

# Method `ParameterSet$getParameterSupport`

ps <- ParameterSet$new(id = "prob",
    value = 0.2,
    support = set6::Interval$new(0, 1),
    settable = TRUE,
    description = "Probability of success")
ps$getParameterSupport("prob")

# Method `ParameterSet$getParameterValue`

ps <- ParameterSet$new(id = "prob",
    value = 0.2,
    support = set6::Interval$new(0, 1),
    settable = TRUE,
    description = "Probability of success")
ps$getParameterValue("prob")

# Method `ParameterSet$setParameterValue`

id <- list("rate")
value <- list(1)
```
support <- list(set6::PosReals$new())

ps <- ParameterSet$new(
  id, value, support
)

ps$setParameterValue(rate = 2)

ps$getParameterValue("rate")

## Method `ParameterSet$merge`

## Not run:

ps1 <- ParameterSet$new(id = c("prob", "qprob"),
  value = c(0.2, 0.8),
  support = list(set6::Interval$new(0, 1), set6::Interval$new(0, 1))
)

ps1$addChecks(function(self) self$getParameterValue("x") > 0)

ps1$addDeps("prob", "qprob", function(self)
  list(qprob = 1 - self$getParameterValue("prob")))

ps2 <- ParameterSet$new(id = "size",
  value = 10,
  support = set6::Interval$new(0, 10, class = "integer"),
)

ps2$addTrafos("size", function(x, self) x + 1)

ps1$merge(ps2)

ps1$print()

## Method `ParameterSet$addDeps`

## Not run:

id <- list("lower", "upper")

ps$addDeps("a", c("b", "c"),
  function(self) {
    list(b = self$getParameterValue("a") + 1,
      c = 1/self$getParameterValue("a"))
  })

## Method `ParameterSet$addChecks`
value <- list(1, 3)
support <- list(set6::PosReals$new(), set6::PosReals$new())
ps <- ParameterSet$new(
  id, value, support
)
ps$addChecks(function(self)
  self$getParameterValue("lower") < self$getParameterValue("upper"))

## End(Not run)

## Method `ParameterSet$addTrafos`

## Not run:
ps <- ParameterSet$new(
  "probs", list(c(1, 1)), set6::Interval$new(0,1)^2
)
ps$addTrafos("probs", function(x, self) return(x / sum(x)))
ps$trafos
ps$setParameterValue(probs = c(1, 2))
ps$getParameterValue("probs")

# Alternate method (better with more parameters)
ps <- ParameterSet$new(
  "probs", list(c(1, 1)), set6::Interval$new(0,1)^2
)
ps$addTrafos(dt = data.table::data.table(
  x = "probs",
  fun = function(x, self) return(x / sum(x))
)
))

## End(Not run)

---

ParameterSetCollection

Parameter Set Collections for Wrapped Distributions

Description

ParameterSetCollection is used to combine multiple ParameterSets in wrapped distributions. Generally only need to be constructed internally.

Super class

distr6::ParameterSet -> ParameterSetCollection

Active bindings

deps Returns ParameterSet dependencies table.
parameterSets Returns ParameterSets in collection.
Methods

Public methods:

• `ParameterSetCollection$new()`  
• `ParameterSetCollection$print()`  
• `ParameterSetCollection$parameters()`  
• `ParameterSetCollection$getParameterValue()`  
• `ParameterSetCollection$getParameterSupport()`  
• `ParameterSetCollection$setParameterValue()`  
• `ParameterSetCollection$merge()`  
• `ParameterSetCollection$addDeps()`  
• `ParameterSetCollection$values()`  
• `ParameterSetCollection$clone()`

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

Usage:
`ParameterSetCollection$new(..., lst = NULL, .checks = NULL, .supports = NULL)`

Arguments:
... ([ParameterSet])

Alternative constructor by supplying a named list of `ParameterSets`.

Examples:

b = Binomial$new()
g = Geometric$new()

`ParameterSetCollection$new(Binom1 = b$parameters(),
Binom2 = b$parameters(),
Geom = g$parameters())`

`ParameterSetCollection$new(lst = list(Binom1 = b$parameters(),
Binom2 = b$parameters(),
Geom = g$parameters()))`

Method `print()`: Prints the `ParameterSetCollection`.

Usage:
`ParameterSetCollection$print(hide_cols = c("settable"), ...)`

Arguments:

hide_cols (character())

Names of columns in the `ParameterSet` to hide whilst printing.

Arguments: ... ANY

Additional arguments, currently unused.
**Method** `parameters()`: Returns the full parameter details for the supplied parameter, or returns `self` if `id` is `NULL` or unmatched.

*Usage:*

```
ParameterSetCollection$parameters(id = NULL)
```

*Arguments:*

- `id` character()
  - id of parameter to return.

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

*Usage:*

```
ParameterSetCollection$getParameterValue(id, error = "warn")
```

*Arguments:*

- `id` (character(1)) To return the parameter for a specific distribution, use the parameter ID with the distribution name prefix, otherwise to return the parameter for all distributions omit the prefix. See examples.
- `error` (character(1))
  - If "warn" then returns a warning on error, otherwise breaks if "stop".

*Examples:*

```r
psc <- ParameterSetCollection$new(Binom1 = Binomial$new()$parameters(),
                                 Binom2 = Binomial$new()$parameters(),
                                 Geom = Geometric$new()$parameters())
psc$getParameterValue("Binom1_prob")
psc$getParameterValue("prob")
```

**Method** `getParameterSupport()`: Returns the support of the supplied parameter.

*Usage:*

```
ParameterSetCollection$getParameterSupport(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".

*Returns: A* `set6::Set` object.

*Examples:*

```r
b <- Binomial$new()
g <- Geometric$new()
psc <- ParameterSetCollection$new(Binom1 = b$parameters(),
                                 Binom2 = b$parameters(),
                                 Geom = g$parameters())
psc$getParameterSupport("Binom1_prob")
```

**Method** `setParameterValue()`: Sets the value(s) of the given parameter(s). Because of R6 reference semantics this also updates the `ParameterSet` of the wrapped distribution, and vice versa. See examples.
Usage:
ParameterSetCollection$setParameterValue(..., lst = NULL, error = "warn")

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".

Examples:
b <- Binomial$new()
g <- Geometric$new()
pse <- ParameterSetCollection$new(Binom1 = b$parameters(),
   Binom2 = b$parameters(),
   Geom = g$parameters())
pse$getParameterValue("Binom1_prob")
b$getParameterValue("prob")
pse$setParameterValue(Binom1_prob = 0.4)
   # both updated
pse$getParameterValue("Binom1_prob")
b$getParameterValue("prob")
g$setParameterValue(prob = 0.1)
   # both updated
pse$getParameterValue("Geom_prob")
g$getParameterValue("prob")

Method merge(): Merges other ParameterSetCollections into self.
Usage:
ParameterSetCollection$merge(..., lst = NULL)

Arguments:
... ([ParameterSetCollection]s)
lst (list(1))
   Alternative argument for passing parameters. List names should be parameter names and
   list values are the new values to set.
'lst' (list())
   Alternative method of passing a list of ParameterSetCollections.

Examples:
b <- Binomial$new()
g <- Geometric$new()
pse <- ParameterSetCollection$new(Binom = b$parameters())
pse2 <- ParameterSetCollection$new(Geom = g$parameters())
pse$merge(pse2)$parameters()
**Method** `addDeps()`: Dependencies should be added to internal `ParameterSets`.

*Usage:*

```r
ParameterSetCollection$addDeps(...)
```

*Arguments:*

... ANY

Ignored.

**Method** `values()`: Returns parameter set values as a named list.

*Usage:*

```r
ParameterSetCollection$values(settable = TRUE)
```

*Arguments:*

settable (logical(1))

If TRUE (default) only returns values of settable parameters, otherwise returns all.

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

*Usage:*

```r
ParameterSetCollection$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

**Examples**

```r
## Method 'ParameterSetCollection$new'
## ---------------------------

b = Binomial$new()
g = Geometric$new()
ParameterSetCollection$new(Binom1 = b$parameters(),
                          Binom2 = b$parameters(),
                          Geom = g$parameters())

ParameterSetCollection$new(lst = list(Binom1 = b$parameters(),
                          Binom2 = b$parameters(),
                          Geom = g$parameters()))

## Method 'ParameterSetCollection$getParameterValue'
## -----------------------------------------------

psc <- ParameterSetCollection$new(Binom1 = Binomial$new()$parameters(),
                                  Binom2 = Binomial$new()$parameters(),
                                  Geom = Geometric$new()$parameters())

psc$getParameterValue("Binom1_prob")
psc$getParameterValue("prob")
```

```r
## Method 'ParameterSetCollection$getParameterValue'
## -----------------------------------------------

## Method 'ParameterSetCollection$getParameterValue'
## -----------------------------------------------
```
## Method `ParameterSetCollection$getParameterSupport`
```
# ------------------------------------------------
b <- Binomial$new()
g <- Geometric$new()
psc <- ParameterSetCollection$new(Binom1 = b$parameters(),
                                Binom2 = b$parameters(),
                                Geom = g$parameters())
psc$getParameterSupport("Binom1_prob")
# ------------------------------------------------
```

## Method `ParameterSetCollection$setParameterValue`
```
# ------------------------------------------------
b <- Binomial$new()
g <- Geometric$new()
psc <- ParameterSetCollection$new(Binom1 = b$parameters(),
                                Binom2 = b$parameters(),
                                Geom = g$parameters())
psc$getParameterValue("Binom1_prob")
b$getParameterValue("prob")
psc$setParameterValue(Binom1_prob = 0.4)
# both updated
psc$getParameterValue("Binom1_prob")
b$getParameterValue("prob")
g$setParameterValue(prob = 0.1)
# both updated
psc$getParameterValue("Geom_prob")
g$getParameterValue("prob")
# ------------------------------------------------
```

## Method `ParameterSetCollection$merge`
```
# ------------------------------------------------
b <- Binomial$new()
g <- Geometric$new()
psc2 <- ParameterSetCollection$new(Geom = g$parameters())
pcur <- ParameterSetCollection$new(Binom = b$parameters())
pcur$merge(psc2)$parameters()
```

---

**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 $\alpha, \beta > 0$.

The distribution is supported on $[\beta, \infty)$.

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.

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.

Methods

Public methods:

- Pareto$new()
- Pareto$mean()
- Pareto$mode()
- Pareto$median()
- Pareto$variance()
- Pareto$skewness()
- Pareto$kurtosis()
- Pareto$entropy()
- Pareto$mgf()
- Pareto$pgf()
- Pareto$setParameterValue()
- Pareto$clone()

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

Usage:
Pareto$new(shape = 1, scale = 1, 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

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

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,

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

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:
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:
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:
`Pareto$pgf(z, ...)`

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

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

Usage:
`Pareto$setParameterValue(..., lst = NULL, error = "warn")`

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".

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: Arccosine, 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, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete

---

**pdf**

*Probability Density Function*

**Description**

See `Distribution$pdf`

**Usage**

```r
pdf(object, ..., log = FALSE, simplify = TRUE, data = NULL)
```

**Arguments**

- `object` *(Distribution)*)
- `...` *(numeric())*) Points to evaluate the probability density function of the distribution. 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 log-pdf. Default is FALSE.
- `simplify` *logical(1)*) If TRUE (default) simplifies the pdf 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.

**Value**

Pdf evaluated at given points as either a numeric if `simplify` is TRUE or as a `data.table::data.table`. 
pdfPNorm

Description

The p-norm of the pdf evaluated between given limits or over the whole support.

Usage

pdfPNorm(object, p = 2, lower = NULL, upper = NULL)

Arguments

object  Distribution.
p         p-norm to calculate.
lower    lower limit for integration, default is infimum.
upper    upper limit for integration, default is supremum.

See Also

ExoticStatistics and decorate

pdfSquared2Norm

Description

The squared 2-norm of the pdf evaluated up to a given limit, possibly shifted.

Usage

pdfSquared2Norm(object, x = 0, upper = Inf)

Arguments

object  Distribution.
x         amount to shift the result.
upper    upper limit of the integral.

Value

Squared 2-norm of pdf evaluated between limits as a numeric.
## pgf

**Probability Generating Function**

### Description

Probability generating function of a distribution

### Usage

```r
pgf(object, z, ...)
```

### Arguments

- `object`: Distribution.
- `z`: integer to evaluate characteristic function at.
- `...`: Passed to `genExp`.

### Value

Probability generating function evaluated at `z` as a numeric if distribution is discrete, otherwise NaN.

---

## 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.

### 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

## 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",
      ...)
plot.VectorDistribution

Plotting Distribution Functions for a VectorDistribution

Description

Helper function to more easily plot distributions inside a VectorDistribution.

Usage

## 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
## Not run:
# Plot pdf of Normal distribution
v <- VectorDistribution$new(list(Normal$new(), Normal$new(mean = 2)))
plot(v)
plot(v, fun = "surv")
plot(v, fun = "quantile", ylim = c(-4, 4), col = c("blue", "purple"))

## End(Not run)
```

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,

$$f(x) = \frac{\lambda^x \cdot exp(-\lambda)}{x!}$$

for $\lambda > 0$.

The distribution is supported on the Naturals.

#### Value
Returns an R6 object inheriting from class SDistribution.

#### Super classes
```
distr6::Distribution -> distr6::SDistribution -> 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 = 1, 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

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

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

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

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,

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

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

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

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

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

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

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

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

Method \text{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:
Poisson$\text{pgf}(z, \ldots)$

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

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

Usage:
Poisson$\text{clone}(\text{deep} = \text{FALSE})$

Arguments:
deepl Whether to make a deep clone.

References

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

Other univariate distributions: \text{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, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Triangular, Uniform, Wald, Weibull, WeightedDiscrete}
prec

**Precision of a Distribution**

**Description**

Precision of a distribution assuming variance is provided.

**Usage**

```r
prec(object)
```

**Arguments**

- `object`: Distribution.

**Value**

Reciprocal of variance as a numeric.

---

print.ParameterSet

**Print a ParameterSet**

**Description**

Prints a ParameterSet as a data.table with strprint variants of R6 classes.

**Usage**

```r
## S3 method for class 'ParameterSet'
print(x, hide_cols = c("settable"), ...)
```

**Arguments**

- `x`: ParameterSet
- `hide_cols`: string, if given the data.table is filtered to hide these columns
- `...`: ignored, added for S3 consistency
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`  

**Details**

A product distribution is defined by

\[
F_P(X_1 = x_1, ..., X_N = x_N) = F_{X_1}(x_1) \times \ldots \times F_{X_N}(x_N)
\]

#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
)

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.

Examples:
\dontrun{
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))
)
)

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

Usage:
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:
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 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:
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:**

```r
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 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
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:**
ProductDistribution$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.

terms 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:

ProductDistribution$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

See Also

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

Examples

```r
# Not run:
ProductDistribution$new(list(Binomial$new(prob = 0.5, size = 10), Normal$new(mean = 15)))
```
## properties

Returns the properties of the distribution.

### Usage

```
properties(object)
```
Arguments

object  Distribution.

Value

List of distribution properties.

R6 Usage

R6::$properties

---

\texttt{qqplot}  \quad \textit{Quantile-Quantile Plots for \texttt{distr}6 Objects}

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

\texttt{qqplot(x, y, npoints = 3000, idline = TRUE, plot = TRUE, ...)}

Arguments

- \texttt{x}  \quad \texttt{distr}6 object or numeric vector.
- \texttt{y}  \quad \texttt{distr}6 object or numeric vector.
- \texttt{npoints}  \quad \texttt{number of evaluation points}.
- \texttt{idline}  \quad \texttt{logical}; if \texttt{TRUE} (default), the line $y = x$ is plotted.
- \texttt{plot}  \quad \texttt{logical}; if \texttt{TRUE} (default), figures are displayed in the plot window; otherwise a \texttt{data.table::data.table} of points and calculated values is returned.
- \texttt{...}  \quad \texttt{graphical parameters}.

Details

If \texttt{x} or \texttt{y} are given as numeric vectors then they are first passed to the \texttt{Empirical} distribution. The \texttt{Empirical} distribution is a discrete distribution so quantiles are equivalent to the the Type 1 method in \texttt{quantile}.

Author(s)

Chijing Zeng

See Also

\texttt{plot.Distribution} for plotting a \texttt{distr}6 object.
Examples

```r
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

Inverse Cumulative Distribution Function

Usage

```r
## S3 method for class 'Distribution'
quantile(
    x,
    ..., 
    lower.tail = TRUE,
    log.p = FALSE,
    simplify = TRUE,
    data = NULL
)
```

Arguments

- `x` : (Distribution)
  Points to evaluate the quantile function of the distribution. 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, $X > x$.
- `log.p` : logical(1)
  If TRUE returns log-cdf. Default is FALSE.
- `simplify` : logical(1)
  If TRUE (default) simplifies the pdf 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.
Quartic

Value
Quantile evaluated at given points as either a numeric if simplify is TRUE or as a data.table::data.table.

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:
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
**Rand**

*Random Simulation Function*

**Description**

See Distribution$rand

**Usage**

`rand(object, n, simplify = TRUE)`

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>(Distribution)</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>(numeric(1))</td>
<td>Number of points to simulate from the distribution. If length greater than 1, then n &lt;- length(n).</td>
</tr>
<tr>
<td>simplify</td>
<td>logical(1)</td>
<td>If TRUE (default) simplifies the pdf if possible to a numeric, otherwise returns a data.table::data.table.</td>
</tr>
</tbody>
</table>

**Value**

Simulations as either a numeric if simplify is TRUE or as a data.table::data.table.

---

**Rayleigh**

*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 $\alpha > 0$.

The distribution is supported on $[0, \infty)$.

**Value**

Returns an R6 object inheriting from class SDistribution.
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 = 1, 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

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

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

... 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:
based (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:
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.

---

**setParameterValue**  
*Parameter Value Setter*

**Description**

Sets the value of the given parameter.

**Usage**

```r
setParameterValue(object, ..., lst = NULL, error = "warn")
```
Arguments

- object: Distribution or ParameterSet.
- ...: named parameters and values to update, see details.
- lst: optional list, see details.
- error: character, value to pass to stopwarn.

Value

An R6 object of class ParameterSet.

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:

$$f(x) = \left(\frac{\beta}{\alpha}\right) \left(\frac{x - \gamma}{\alpha}\right)^{\beta-1} \left(1 + \left(\frac{x - \gamma}{\alpha}\right)^{\beta}\right)^{-2}$$

for $\alpha, \beta > 0$ and $\gamma \geq 0$.

The distribution is supported on the non-negative Reals.

Value

Returns an R6 object inheriting from class SDistribution.

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.
Methods

Public methods:

• ShiftedLoglogistic$new()
• ShiftedLoglogistic$mean()
• ShiftedLoglogistic$mode()
• ShiftedLoglogistic$median()
• ShiftedLoglogistic$variance()
• ShiftedLoglogistic$pgf()
• ShiftedLoglogistic$setParameterValue()
• ShiftedLoglogistic$clone()

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

Usage:
ShiftedLoglogistic$new(
  scale = 1,
  shape = 1,
  location = 0,
  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

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

with an integration analogue for continuous distributions.

Usage:
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 available 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
\[
\text{var}_X = E[X^2] - E[X]^2
\]
where \(E_X\) is the expectation of distribution \(X\). If the distribution is multivariate the covariance
matrix is returned.
Usage:
ShiftedLoglogistic$variance(...)
Arguments:
... 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:
ShiftedLoglogistic$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:
ShiftedLoglogistic$setParameterValue(..., lst = NULL, error = "warn")
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".

**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**


**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

---

**Description**

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

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

over the support \( x \in R \).

**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

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

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

Usage:

```
Sigmoid$variance(...)  
```

Arguments:

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

**Usage:**

```r
Sigmoid$clone(deep = FALSE)
```

**Arguments:**

depth: Whether to make a deep clone.

---

**See Also**

Other kernels: Cosine, Epanechnikov, LogisticKernel, NormalKernel, Quartic, Silverman, TriangularKernel, Tricube, Triweight, UniformKernel

---

**Silverman**

**Silverman Kernel**

**Description**

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

\[
f(x) = \exp(-|x|/\sqrt{2})/2 \ast \sin(|x|/\sqrt{2} + \pi/4)\]

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 -> Silverman

**Public fields**

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

**Methods**

**Public methods:**

- `Silverman$new()`
- `Silverman$pdfSquared2Norm()`
- `Silverman$cdfSquared2Norm()`
- `Silverman$variance()`
- `Silverman$clone()`
Method new(): Creates a new instance of this R6 class.

Usage:
Silverman$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:
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.
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
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.

See Also
Other kernels: Cosine, Epanechnikov, LogisticKernel, NormalKernel, Quartic, Sigmoid, TriangularKernel, Tricube, Triweight, UniformKernel
skewness

**Description**

Skewness of a distribution

**Usage**

skewness(object, ...)

**Arguments**

- object: Distribution.
- ...: Passed to $genExp$.

**Value**

Skewness as a numeric.

---

skewnessType

**Description**

Deprecated. Use $properties$skewness.

**Usage**

skewnessType(object)

**Arguments**

- object: Distribution.

**Value**

If the distribution skewness is present in properties, returns one of "negative skew", "no skew", "positive skew", otherwise returns NULL.
Skewness Type

Description

Gets the type of skewness

Usage

skewType(skew)

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`.

See Also

skewness, exkurtosisType

Examples

skewType(1)
skewType(0)
skewType(-1)

Stdev

Standard Deviation of a Distribution

Description

Standard deviation of a distribution assuming variance is provided.

Usage

stdev(object)
Arguments

object     Distribution.

Value

Square-root of variance as a numeric.

strprint     String Representation of Print

Description

Parsable string to be supplied to print, data.frame, etc.

Usage

strprint(object, n = 2)

Arguments

object     R6 object
n          Number of parameters to display before & after ellipsis

Details

strprint is a suggested method that should be included in all R6 classes to be passed to methods such as cat, summary and print. Additionally can be used to easily parse R6 objects into data-frames, see examples.

Value

String representation of the distribution.

Examples

Triangular$new()$strprint()
Triangular$new()$strprint(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,

\[
f(x) = \frac{\Gamma((\nu + 1)/2)}{(\sqrt{\nu\pi})\Gamma(\nu/2)} \ast \left(1 + \frac{x^2}{\nu}\right)^{-\left(\nu + 1\right)/2}
\]

for \( \nu > 0 \).

The distribution is supported on the Reals.

**Value**

Returns an R6 object inheriting from class SDistribution.

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

Usage:
StudentT$new(df = 1, 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:
StudentT$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:
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:
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:
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:
StudentT$\text{cf}(t, \ldots)

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:
StudentT$\text{pgf}(z, \ldots)

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:
StudentT$\text{clone}(\text{deep} = \text{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
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,

\[
f(x) = \frac{\nu^{\nu/2} \exp\left(-\left(\nu \lambda^2\right)/(2(x^2+\nu))\right)/\left(\sqrt{\pi} \Gamma\left(\nu/2\right)2^{(\nu-1)/2}(x^2+\nu)^{\left(\nu+1\right)/2}\right)}{\int_0^\infty y^{\nu} \exp\left(-\left(1/2(y-x\lambda/\sqrt{x^2+\nu})^2\right)\right) dy}
\]

for \( \nu > 0 \), \( \lambda \in \mathbb{R} \).

The distribution is supported on the Reals.

Value

Returns an R6 object inheriting from class SDistribution.

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 = 1, location = 0, 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

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

with an integration analogue for continuous distributions.

**Usage:**
- `StudentTNoncentral$mean(...)`

**Arguments:**
- ... Unused.

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

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

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

**Usage:**
- `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

summary.Distribution  Distribution Summary

Description

Summary method for distribution objects (and all child classes).

Usage

```r
## S3 method for class 'Distribution'
summary(object, full = TRUE, ...)  # S3 method for class 'Distribution'

Arguments

- `object`: Distribution.
- `full`: logical; if TRUE (default), gives an extended summary, otherwise brief.
- `...`: additional arguments.

Value

Printed summary of the distribution.

R6 Usage

```r
$summary(full = TRUE)
```

See Also

Distribution
sup \quad \textit{Supremum Accessor}

Description

Returns the distribution supremum as the supremum of the support.

Usage

\texttt{sup(object)}

Arguments

\texttt{object} \quad \text{Distribution.}

Value

Supremum as a numeric.

R6 Usage

\texttt{$\$sup$}

support \quad \textit{Support Accessor - Deprecated}

Description

Deprecated. Use $\$properties$\$support$

Usage

\texttt{support(object)}

Arguments

\texttt{object} \quad \text{Distribution.}

Details

The support of a probability distribution is defined as the interval where the pmf/pdf is greater than zero,

$$Supp(X) = \{x \in R : f_X(x) > 0\}$$

where $f_X$ is the pmf if distribution $X$ is discrete, otherwise the pdf.
survival

Value

An R6 object of class set6::Set.

R6 Usage

$support

---

survival Survival Function

Description

See ExoticStatistics$survival.

Usage

survival(object, ..., log = FALSE, simplify = TRUE, data = NULL)

Arguments

object (Distribution).

... (numeric())

Points to evaluate the probability density function of the distribution. 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 log-Hazard Default is FALSE.

simplify logical(1)

If TRUE (default) simplifies the pdf 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.

Value

Survival function as a numeric, natural logarithm returned if log is TRUE.
survivalAntiDeriv  Survival Function Anti-Derivative

Description
The anti-derivative of the survival function between given limits or over the full support.

Usage
survivalAntiDeriv(object, lower = NULL, upper = NULL)

Arguments
object  Distribution.
lower  lower limit for integration, default is infimum.
upper  upper limit for integration, default is supremum.

Value
Antiderivative of the survival function evaluated between limits as a numeric.

survivalPNorm  Survival Function P-Norm

Description
The p-norm of the survival function evaluated between given limits or over the whole support.

Usage
survivalPNorm(object, p = 2, lower = NULL, upper = NULL)

Arguments
object  Distribution.
p  p-norm to calculate.
lower  lower limit for integration, default is infimum.
upper  upper limit for integration, default is supremum.

Value
Given p-norm of survival function evaluated between limits as a numeric.
symmetry

Symmetry Accessor - Deprecated

Description

Deprecated. Use $properties$symmetry.

Usage

symmetry(object)

Arguments

object Distribution.

Value

One of "symmetric" or "asymmetric".

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

testDiscrete(Binomial$new()) # FALSE

description

Validation checks to test if Distribution is discrete.

Usage

testDiscrete(object, errmsg = paste(object$short_name, "is not discrete"))
checkDiscrete(object, errmsg = paste(object$short_name, "is not discrete"))
assertDiscrete(object, errmsg = paste(object$short_name, "is not discrete"))

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Distribution</td>
</tr>
<tr>
<td>errmsg</td>
<td>custom error message to return if assert/check fails</td>
</tr>
</tbody>
</table>

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

testDiscrete(Binomial$new()) # FALSE
testDistribution

Description

Validation checks to test if a given object is a Distribution.

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 assert/check/test/DistributionList

Description

Validation checks to test if a given object is a list of Distributions.

Usage

testDistributionList(
  object,
  errmsg = "One or more items in the list are not Distributions"
)

checkDistributionList(
  object,
  errmsg = "One or more items in the list are not 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
**Description**

Validation checks to test if Distribution is leptokurtic.

**Usage**

```r
testLeptokurtic(
  object,
  errormsg = paste(object$short_name, "is not leptokurtic")
)
```

```r
checkLeptokurtic(
  object,
  errormsg = paste(object$short_name, "is not leptokurtic")
)
```

```r
assertLeptokurtic(
  object,
  errormsg = paste(object$short_name, "is not leptokurtic")
)
```

**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
testLeptokurtic(Binomial$new())
```
Description

Validation checks to test if Distribution is matrixvariate.

Usage

testMatrixvariate(
  object,
  errormsg = paste(object$short_name, "is not matrixvariate")
)

checkMatrixvariate(
  object,
  errormsg = paste(object$short_name, "is not matrixvariate")
)

assertMatrixvariate(
  object,
  errormsg = paste(object$short_name, "is not matrixvariate")
)

Arguments

object Distribution

erormsg 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())
**testMixture**

Description

Validation checks to test if Distribution is mixture.

Usage

```r
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

```r
testMixture(Binomial$new()) # FALSE
```

**testMultivariate**

Description

Validation checks to test if Distribution is multivariate.

Usage

```r
testMultivariate(object, errormsg = paste(object$short_name, "is not multivariate"))
checkMultivariate(object, errormsg = paste(object$short_name, "is not multivariate"))
```

Examples

```r
testMultivariate(Binomial$new()) # FALSE
```
assertMultivariate(
    object,
    errormsg = paste(object$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

described testNegativeSkew

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())
Description

Validation checks to test if a given object is a ParameterSet.

Usage

testParameterSet(
  object,
  errmsg = paste(object, "is not an R6 ParameterSet object")
)

checkParameterSet(
  object,
  errmsg = paste(object, "is not an R6 ParameterSet object")
)

assertParameterSet(
  object,
  errmsg = paste(object, "is not an R6 ParameterSet object")
)

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

testParameterSet(5)  # FALSE
testParameterSet(Binomial$new()$parameters())  # TRUE
testParameterSetCollection

**Description**

Validation checks to test if a given object is a ParameterSetCollection.

**Usage**

```r
testParameterSetCollection(
  object,
  errmsg = paste(object, "is not an R6 ParameterSetCollection object")
)
```

```r```

```r
checkParameterSetCollection(
  object,
  errmsg = paste(object, "is not an R6 ParameterSetCollection object")
)
```

```r```

```r
assertParameterSetCollection(
  object,
  errmsg = paste(object, "is not an R6 ParameterSetCollection object")
)
```

**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**

```r
# FALSE
testParameterSetCollection(5)
# TRUE
testParameterSetCollection(ParameterSetCollection$new(Binom = Binomial$new()$parameters()))
```
testParameterSetCollectionList

assert/check/test/ParameterSetCollectionList

Description

Validation checks to test if a given object is a list of ParameterSetCollections.

Usage

```
testParameterSetCollectionList( 
  object, 
  errmsg = "One or more items in the list are not ParameterSetCollections"
)
```

```
checkParameterSetCollectionList( 
  object, 
  errmsg = "One or more items in the list are not ParameterSetCollections"
)
```

```
assertParameterSetCollectionList( 
  object, 
  errmsg = "One or more items in the list are not ParameterSetCollections"
)
```

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

```
testParameterSetCollectionList(list(Binomial$new(), 5)) # FALSE  
testParameterSetCollectionList(list(ParameterSetCollection$new( 
  Binom = Binomial$new()$parameters() 
))) # TRUE
```
Description

Validation checks to test if a given object is a list of ParameterSets.

Usage

```r
testParameterSetList(
  object,
  errmsg = "One or more items in the list are not ParameterSets"
)
```

```r
checkParameterSetList(
  object,
  errmsg = "One or more items in the list are not ParameterSets"
)
```

```r
assertParameterSetList(
  object,
  errmsg = "One or more items in the list are not ParameterSets"
)
```

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

```r
testParameterSetList(list(Binomial$new(), 5)) # FALSE
testParameterSetList(list(Binomial$new(), Exponential$new())) # TRUE
```
Description

Validation checks to test if Distribution is platykurtic.

Usage

testPlatykurtic(
  object,
  errmsg = paste(object$short_name, "is not platykurtic")
)

checkPlatykurtic(
  object,
  errmsg = paste(object$short_name, "is not platykurtic")
)

assertPlatykurtic(
  object,
  errmsg = paste(object$short_name, "is not platykurtic")
)

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

testPlatykurtic(Binomial$new())
Description

Validation checks to test if Distribution is positive skew.

Usage

```r
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

```r
testPositiveSkew(Binomial$new())
```
testSymmetric

Description
Validation checks to test if Distribution is symmetric.

Usage

```r
testSymmetric(object, errmsg = paste(object$short_name, "is not symmetric"))
```

```r
checkSymmetric(object, errmsg = paste(object$short_name, "is not symmetric"))
```

```r
assertSymmetric(
  object,
  errmsg = paste(object$short_name, "is not symmetric")
)
```

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

```r
testSymmetric(Binomial$new()) # FALSE
```

testUnivariate

Description
Validation checks to test if Distribution is univariate.
traits

Usage

testUnivariate(
  object,
  errormsg = paste(object$short_name, "is not univariate")
)

checkUnivariate(
  object,
  errormsg = paste(object$short_name, "is not univariate")
)

assertUnivariate(
  object,
  errormsg = paste(object$short_name, "is not univariate")
)

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

testUnivariate(Binomial$new()) # TRUE

---

traits Traits Accessor

Description

Returns the traits of the distribution.

Usage

traits(object)

Arguments

object Distribution.

Value

List of traits.
**Triangular Distribution Class**

**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,

\[
\begin{align*}
  f(x) &= 0, x < a \\
  f(x) &= 2(x - a)/((b - a)(c - a)), a \leq x < c \\
  f(x) &= 2/(b - a), x = c \\
  f(x) &= 2(b - x)/((b - a)(b - c)), c < x \leq b \\
  f(x) &= 0, x > b \text{ for } a, b, c \in \mathbb{R}, a \leq c \leq b.
\end{align*}
\]

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

**Value**

Returns an R6 object inheriting from class SDistribution.

**Super classes**

```
distr6::Distribution -> distr6::SDistribution -> Triangular
```

**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:**

- `Triangular$new()`
- `Triangular$mean()`
- `Triangular$mode()`
• Triangular$median()
• Triangular$variance()
• Triangular$skewness()
• Triangular$kurtosis()
• Triangular$entropy()
• Triangular$mgf()
• Triangular$cf()
• Triangular$pgf()
• Triangular$setParameterValue()
• Triangular$clone()

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

Usage:
Triangular$new(
  lower = 0,
  upper = 1,
  mode = (lower + upper)/2,
  symmetric = FALSE,
  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, symmetric = FALSE)
Triangular$new(lower = 2, upper = 5, mode = 4)

# You can view the type of Triangular distribution with $description
Triangular$new(lower = 2, upper = 5, symmetric = TRUE)$description
Triangular$new(lower = 2, upper = 5, symmetric = FALSE)$description

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:
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,
\[ \text{sk}_X = E_X\left[\frac{x - \mu}{\sigma}\right]^3 \]
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:
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.
Triangular

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

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

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

**Usage:**

```
Triangular$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:**

```
Triangular$setParameterValue(..., lst = NULL, error = "warn")
```

**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".

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

**Usage:**

```
Triangular$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, F DistributionNoncentral, F Distribution, Frechet, Gamma, Gompertz, Gumbel, InverseGamma, Laplace, Logistic, Loglogistic, Lognormal, MultivariateNormal, Normal, Pareto, Poisson, Rayleigh, ShiftedLoglogistic, StudentTNoncentral, StudentT, Uniform, Wald, Weibull

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

```r
## Method `Triangular$new`

Triangular$new(lower = 2, upper = 5, symmetric = TRUE)
Triangular$new(lower = 2, upper = 5, symmetric = FALSE)
Triangular$new(lower = 2, upper = 5, mode = 4)

# You can view the type of Triangular distribution with $description
Triangular$new(lower = 2, upper = 5, symmetric = TRUE)$description
Triangular$new(lower = 2, upper = 5, symmetric = FALSE)$description
```

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

```r
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:

- TriangularKernel$pdfSquared2Norm()
- TriangularKernel$cdfSquared2Norm()
- TriangularKernel$variance()
- TriangularKernel$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:**
TriangularKernel$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:**
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

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

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
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
\texttt{distr6::Distribution} -> \texttt{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:
- \texttt{Tricube$pdfSquared2Norm()}
- \texttt{Tricube$cdfSquared2Norm()}
- \texttt{Tricube$variance()}
- \texttt{Tricube$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{Tricube$pdfSquared2Norm(x = 0, upper = Inf)}

Arguments:
x \ (\texttt{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:
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

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

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:
- 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

distr6::Distribution -> distr6::Kernel -> 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
Truncate a Distribution

Description

S3 functionality to truncate an R6 distribution.

Usage

```r
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 \( T = \text{Truncate}(X, \text{lower, upper}) \) and \( f_X, F_X \) is the pdf/cdf of the original distribution. \( T \) is supported on (].

Super classes

`distr6::Distribution -> distr6::DistributionWrapper -> TruncatedDistribution`
Methods

Public methods:
- `TruncatedDistribution$new()`
- `TruncatedDistribution$setParameterValue()`
- `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:
```r
TruncatedDistribution$new(
  Binomial$new(prob = 0.5, size = 10),
  lower = 2, upper = 4
)
```

# alternate constructor
truncate(Binomial$new(), lower = 2, upper = 4)

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

Usage:
`TruncatedDistribution$setParameterValue(..., lst = NULL, error = "warn")`

Arguments:
- `...` 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".

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

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

Arguments:
- `deep` Whether to make a deep clone.
See Also

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

Examples

```r
# alternate constructor
truncation(Binomial$new(), lower = 2, upper = 4)
```

---

**type**  
*Type Accessor - Deprecated*

### Description

Deprecated. Use `$traits$type`

### Usage

`type(object)`

#### Arguments

- `object` Distribution.

#### Value

An R6 object of class `set6::Set`.

#### R6 Usage

 `$type`
**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 \(-\infty < a < b < \infty\).

The distribution is supported on \([a,b]\).

**Value**
Returns an R6 object inheriting from class SDistribution.

**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.

**Methods**

**Public methods:**
- `Uniform$new()`
- `Uniform$mean()`
- `Uniform$mode()`
- `Uniform$variance()`
- `Uniform$skewness()`
- `Uniform$kurtosis()`
- `Uniform$entropy()`
- `Uniform$mgf()`
- `Uniform$cf()`
• Uniform$pgf()
• Uniform$setParameterValue()
• Uniform$clone()

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

Usage:
Uniform$new(lower = 0, upper = 1, 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,
\[ sk_X = E_X \left[ \frac{x - \mu}{\sigma}^3 \right] \]

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:
`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
\[ mgf_X(t) = E_X[exp(zt)] \]

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

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

Arguments:
Method \texttt{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:
\texttt{Uniform(cf(t, ...))}

Arguments:
\begin{itemize}
  \item \texttt{t} (integer(1))
    \item integer to evaluate function at.
  \item \ldots Unused.
\end{itemize}

Method \texttt{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:
\texttt{Uniform(pgf(z, ...))}

Arguments:
\begin{itemize}
  \item \texttt{z} (integer(1))
    \item integer to evaluate probability generating function at.
  \item \ldots Unused.
\end{itemize}

Method \texttt{setParameterValue()}: Sets the value(s) of the given parameter(s).

Usage:
\texttt{Uniform(setParameterValue(..., lst = NULL, error = "warn"))}

Arguments:
\begin{itemize}
  \item \ldots ANY
    \item Named arguments of parameters to set values for. See examples.
  \item \texttt{lst} (list(1))
    \item Alternative argument for passing parameters. List names should be parameter names and list values are the new values to set.
  \item \texttt{error} (character(1))
    \item If "warn" then returns a warning on error, otherwise breaks if "stop".
\end{itemize}

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

Usage:
\texttt{Uniform(clone(deep = FALSE))}

Arguments:
\begin{itemize}
  \item \texttt{deep} Whether to make a deep clone.
Uniform Kernel

Author(s)
Yumi Zhou

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, 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

---

Uniform Kernel

**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**

distr6::Distribution -> distr6::Kernel -> UniformKernel

**Public fields**

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

Public methods:
- UniformKernel$pdfSquared2Norm()
- UniformKernel$cdfSquared2Norm()
- UniformKernel$variance()
- UniformKernel$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:
UniformKernel$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:
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

---

### valueSupport

<table>
<thead>
<tr>
<th>valueSupport</th>
<th>Value Support Accessor - Deprecated</th>
</tr>
</thead>
</table>

Description
Deprecated. Use `$traits$valueSupport`

Usage
`valueSupport(object)`

Arguments
- `object` Distribution.

Value
One of "discrete"/"continuous"/"mixture".

---

### variance

<table>
<thead>
<tr>
<th>variance</th>
<th>Distribution Variance</th>
</tr>
</thead>
</table>

Description
The variance or covariance of a distribution, either calculated analytically if or estimated numerically.

Usage
`variance(object, ...)`

Arguments
- `object` Distribution.
- `...` Passed to `$genExp`. 
Value

Variance as a numeric.

---

variateForm  Variate Form Accessor - Deprecated

Description

Deprecated. Use $traits$variateForm

Usage

variateForm(object)

Arguments

object  Distribution.

Value

One of "univariate"/"multivariate"/"matrixvariate".

---

VectorDistribution  Vectorise Distributions

Description

A wrapper for creating a vector of distributions.

Details

A vector distribution is intended 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.
Methods

Public methods:

• VectorDistribution$new()
• VectorDistribution$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:

VectorDistribution$new(
  distlist = NULL,
  distribution = NULL,
  params = NULL,
  shared_params = NULL,
  name = NULL,
  short_name = NULL,
  decorators = NULL,
  vecdist = 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 VectorDistri-
bution.

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 Vec-

torDistribution.

... ANY

Named arguments of parameters to set values for. See examples.

Examples:

\dontref{
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)
  )
)
}

Method wrappedModels(): Returns model(s) wrapped by this wrapper.
Usage:
VectorDistribution\$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 strprint(): Printable string representation of the VectorDistribution. Primarily used internally.

Usage:
VectorDistribution\$strprint(n = 10)

Arguments:
  n (integer(1))
    Number of distributions to include when printing.

Method mean(): Returns named vector of means from each wrapped Distribution.

Usage:
VectorDistribution\$mean(...)

Arguments:
... Passed to CoreStatistics\$genExp if numeric.

Method mode(): Returns named vector of modes from each wrapped Distribution.

Usage:
VectorDistribution\$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 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:**

```r
df <- VectorDistribution$new(
  distribution = "Binomial",
  params = data.frame(size = 9:10, prob = c(0.5,0.6)))

vd$pdf(2)
# Equivalently
vd$pdf(2, 2)

d$v$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.
vector_distribution

simplify logical()
  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()
  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$new(
  distribution = "Binomial",
  params = data.table::data.table(prob = c(0.1, 0.6, 0.2), size = c(2, 4, 6))
)```
\begin{verbatim}
# Alternatively
VectorDistribution$new(
  list(
    Binomial$new(prob = 0.1, size = 2),
    Binomial$new(prob = 0.6, size = 4),
    Binomial$new(prob = 0.2, size = 6)
  )
)

### 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(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)))
\end{verbatim}
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,

\[
f(x) = \frac{\lambda}{2x^3 \pi} \left( \frac{-\lambda(x - \mu)^2}{(2\mu^2 x)} \right)^{1/2} \exp \left( \frac{-\lambda(x - \mu)^2}{(2\mu^2 x)} \right)
\]

for \( \lambda > 0 \) and \( \mu > 0 \).

The distribution is supported on the Positive Reals.

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

Also known as the Inverse Normal distribution.

Sampling is performed as per Michael, Schucany, Haas (1976).

Value

Returns an R6 object inheriting from class SDistribution.

Super classes

distr6::Distribution \rightarrow distr6::SDistribution \rightarrow 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 = 1, shape = 1, 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

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

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,

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

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,

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

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:
Wald$kurtosis(excess = TRUE, ...)

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

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

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

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

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

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

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

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

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 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,

\[
f(x) = \left(\frac{\alpha}{\beta}\right)(x/\beta)^{\alpha-1} \exp\left(-\frac{x}{\beta}\right)^{\alpha}
\]

for \( \alpha, \beta > 0 \).

The distribution is supported on the Positive Reals.

Value

Returns an R6 object inheriting from class SDistribution.

Super classes

distr6::Distribution -> distr6::SDistribution -> Weibull

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:

- Weibull$new()
- Weibull$mean()
- Weibull$mode()
- Weibull$median()
- Weibull$variance()
- Weibull$skewness()
- Weibull$kurtosis()
- Weibull$entropy()
- Weibull$pgf()
- Weibull$setParameterValue()
- Weibull$clone()

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

Usage:

Weibull$new(shape = 1, scale = 1, altscale = NULL, decorators = NULL)

Arguments:

- shape (numeric(1))
  Shape parameter, defined on the positive Reals.
scale (numeric(1))
   Scale parameter, defined on the positive Reals.
altscale (numeric(1))
   Alternative scale parameter, if given then scale is ignored. altscale = scale^shape.
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:*

`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:*

`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 self$mean, otherwise returns self$quantile(0.5).

*Usage:*

`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:*

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

Usage:
Weibull$setParameterValue(..., lst = NULL, error = "warn")
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".

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.

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, 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 \( p_i, i = 1, \ldots, k; \sum p_i = 1 \).

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

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.

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.

Methods

Public methods:

- WeightedDiscrete$new()
- WeightedDiscrete$strprint()
- WeightedDiscrete$mean()
- WeightedDiscrete$mode()
- WeightedDiscrete$variance()
- WeightedDiscrete$skewness()
- WeightedDiscrete$kurtosis()
- WeightedDiscrete$entropy()
- WeightedDiscrete$mgf()
- WeightedDiscrete$cf()
- WeightedDiscrete$pgf()
- WeightedDiscrete$setParameterValue()
- WeightedDiscrete$clone()

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

Usage:
WeightedDiscrete$new(
  data = NULL,
  x = 1,
  pdf = 1,
  cdf = NULL,
  decorators = NULL
)

Arguments:

- **data**: ([data.frame])
  - Deprecated. Use x, pdf, cdf.
  - **x**: numeric()
    - Data samples.
  - **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.

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

Usage:

```
WeightedDiscrete$strprint(n = 2)
```

Arguments:

- **n**: (integer(1))
  - Ignored.

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

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

with an integration analogue for continuous distributions.

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

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

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

*Usage:*

`WeightedDiscrete$variance(...)`

*Arguments:*

... Unused.

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

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

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:*

`WeightedDiscrete$skewness(...)`

*Arguments:*

... Unused.

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

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

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:*

`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.

*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 \( X \).

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 \( X \).

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

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

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

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

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

Usage:
WeightedDiscrete$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:
WeightedDiscrete$setParameterValue(..., lst = NULL, error = "warn")

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".

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

Michael P. McLaughlin.

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

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)
workingSupport  

**Approximate Finite Support**

**Description**

If the distribution has an infinite support then this function calculates the approximate finite limits by finding the largest small number for which \( \text{cdf} = 0 \) and the smallest large number for which \( \text{cdf} = 1 \).

**Usage**

```r
workingSupport(object)
```

**Arguments**

- `object`  
  Distribution.

**Value**

- `set6` object.

---

wrappedModels  

**Gets Internally Wrapped Models**

**Description**

Returns either a list of all the wrapped models or the models named by parameters.

**Usage**

```r
wrappedModels(object, model = NULL)
```

**Arguments**

- `object`  
  Distribution.
- `model`  
  character, see details.

**Value**

If `model` is `NULL` then returns list of models that are wrapped by the wrapper. Otherwise returns `model` given in `model`. 
### [.ParameterSet](#)

**Extract one or more parameters from a ParameterSet**

**Description**

Used to extract one or more parameters from a constructed ParameterSet or ParameterSetCollection.

**Usage**

```r
## S3 method for class 'ParameterSet'
ps[ids, prefix = NULL, ...]
```

**Arguments**

- `ps`  
  ParameterSet  
  ParameterSet from which to extract parameters.

- `ids` (character())
  ids of parameters to extract, if id ends with _ then all parameters starting with ids_ are extracted and the prefix is ignored, prefix can be left NULL. See examples.

- `prefix` (character(1))
  An optional prefix to remove from ids after extraction, assumes _ follows the prefix name, i.e. prefix_ids.

- `...`  
  ANY  
  Ignored, added for consistency.

**Examples**

```r
ps <- VectorDistribution$new(
  distribution = "Binomial",
  params = data.table::data.table(prob = c(0.1, 0.6, 0.2), size = c(2, 4, 6))
)$parameters()

ps["Binom1_prob"] # extracts just Binom1_prob
ps["Binom1_prob", prefix = "Binom1"] # extracts Binom1_prob and removes prefix
ps["Binom1_"] # extracts all Binom1 parameters and removes prefix
```

### [.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.
Index

* continuous distributions
  Arcsine, 8
  Beta, 19
  BetaNoncentral, 23
  Cauchy, 35
  ChiSquared, 42
  ChiSquaredNoncentral, 46
  Dirichlet, 64
  Erlang, 96
  Exponential, 105
  FDistribution, 110
  FDistributionNoncentral, 114
  Frechet, 117
  Gamma, 122
  Gompertz, 134
  Gumbel, 136
  InverseGamma, 148
  Laplace, 156
  Logistic, 170
  Loglogistic, 176
  Lognormal, 180
  MultivariateNormal, 198
  Normal, 207
  Pareto, 227
  Poisson, 237
  Rayleigh, 252
  ShiftedLoglogistic, 258
  StudentT, 269
  StudentTNoncentral, 273
  Triangular, 297
  Uniform, 311
  Wald, 326
  Weibull, 330

* decorators
  CoreStatistics, 52
  ExoticStatistics, 101
  FunctionImputation, 121

* discrete distributions
  Bernoulli, 14
  Binomial, 25
  Categorical, 30
  Degenerate, 60
  DiscreteUniform, 67
  Empirical, 87
  EmpiricalMV, 91
  Geometric, 129
  Hypergeometric, 144
  Logarithmic, 166
  Multinomial, 194
  NegativeBinomial, 202
  WeightedDiscrete, 334

* kernels
  Cosine, 56
  Epanechnikov, 94
  LogisticKernel, 174
  NormalKernel, 211
  Quartic, 250
  Sigmoid, 261
  Silverman, 263
  TriangularKernel, 302
  Tricube, 304
  Triweight, 306
  UniformKernel, 315

* multivariate distributions
  Dirichlet, 64
  EmpiricalMV, 91
  Multinomial, 194
  MultivariateNormal, 198

* univariate distributions
  Arcsine, 8
  Bernoulli, 14
  Beta, 19
  BetaNoncentral, 23
  Binomial, 25
  Categorical, 30
  Cauchy, 35
  ChiSquared, 42
  ChiSquaredNoncentral, 46
Degenerate, 60
DiscreteUniform, 67
Empirical, 87
Erlang, 96
Exponential, 105
FDistribution, 110
FDistributionNoncentral, 114
Frechet, 117
Gamma, 122
Geometric, 129
Gompertz, 134
Gumbel, 136
Hypergeometric, 144
InverseGamma, 148
Laplace, 156
Logarithmic, 166
Logistic, 170
Loglogistic, 176
Lognormal, 180
NegativeBinomial, 202
Normal, 207
Pareto, 227
Poisson, 237
Rayleigh, 252
ShiftedLoglogistic, 258
StudentT, 269
StudentTNoncentral, 273
Triangular, 297
Uniform, 311
Wald, 326
Weibull, 330
WeightedDiscrete, 334

* wrappers
Convolution, 50
DistributionWrapper, 82
HuberizedDistribution, 142
MixtureDistribution, 188
ProductDistribution, 242
TruncatedDistribution, 308
VectorDistribution, 318
.Distribution (ProductDistribution), 242
+ .Distribution (Convolution), 50
- .Distribution (Convolution), 50
[.ParameterSet, 341
[.VectorDistribution, 341

array, 40, 58, 76–78, 103, 104, 141, 190, 191, 232, 244–246, 249, 277, 323–325
as.data.table.ParameterSet, 12
as.MixtureDistribution, 12
as.ParameterSet, 13
as.ProductDistribution, 14
as.VectorDistribution, 14
assertContinuous (testContinuous), 279
assertDiscrete (testDiscrete), 280
assertDistribution (testDistribution), 281
assertDistributionList (testDistributionList), 282
assertLeptokurtic (testLeptokurtic), 283
assertMatrixvariate (testMatrixvariate), 284
assertMesokurtic (testMesokurtic), 285
assertMixture (testMixture), 286
assertMultivariate (testMultivariate), 286
assertNegativeSkew (testNegativeSkew), 287
assertNoSkew (testNoSkew), 288
assertParameterSet (testParameterSet), 289
assertParameterSetCollection (testParameterSetCollection), 290
assertParameterSetCollectionList (testParameterSetCollectionList), 291
assertParameterSetList (testParameterSetList), 292
assertPlatykurtic (testPlatykurtic), 293
assertPositiveSkew (testPositiveSkew), 294
assertSymmetric (testSymmetric), 295
assertUnivariate (testUnivariate), 295

Bernoulli, 12, 14, 22, 25, 29, 35, 39, 46, 50, 64, 71, 91, 93, 100, 110, 114, 116, 120, 127, 133, 136, 140, 147, 151, 160, 169, 174, 179, 185, 198, 207,
data.table::data.table(), 235
decorate, 58, 81, 194, 233
decorators, 59
Delta (Degenerate), 60
Dirac (Degenerate), 60
Dirichlet, 12, 22, 24, 39, 46, 50, 64, 93, 100, 109, 114, 116, 120, 127, 136, 140, 151, 160, 174, 179, 184, 198, 202, 211, 231, 240, 255, 261, 272, 275, 301, 315, 330, 334
distr6 (distr6-package), 7
distr6::package, 7
distr6::DistributionDecorator, 52, 101, 121
distr6::DistributionWrapper, 51, 142, 188, 242, 308, 318
distr6::Kernel, 56, 94, 174, 212, 250, 261, 263, 302, 304, 306, 315
distr6::ParameterSet, 222
distr6::SDistribution, 8, 15, 19, 23, 25, 31, 36, 42, 47, 60, 64, 67, 82, 97, 92, 96, 106, 110, 114, 117, 123, 129, 134, 136, 144, 148, 156, 166, 170, 176, 180, 195, 199, 203, 207, 228, 237, 253, 258, 269, 273, 297, 311, 327,
INDEX

255, 261, 272, 275, 301, 315, 330, 334, 339


Fisk(Loglogistic), 176

FunctionImputation, 55, 64, 76–78, 105, 121, 142, 195, 198, 235, 308, 327


Gaussian (Normal), 207
generalPNorm, 127
genExp, 128


getParameterSupport, 133
getParameterValue, 134


graphics::layout(), 235
graphics::par(), 235


hazard, 140
huberize, 141
HuberizedDistribution, 51, 84, 141, 142, 192, 246, 310, 325


inf, 86, 147
integrate, 54, 128

InverseGaussian (Wald), 326
InverseNormal (Wald), 326
InverseWeibull (Frechet), 117
iqr, 152

Kernel, 152, 165
kthMoment, 155
kurtosis, 101, 155
kurtosisType, 156


liesInSupport, 161
liesInType, 161
lines.Distribution, 162, 235
listDecorators, 81, 163
listDecorators(), 59
listDistributions, 164
listDistributions(), 189, 243, 319
listKernels, 165
listWrappers, 82, 165
Logarithmic, 12, 19, 22–25, 29, 35, 39, 46, 50, 64, 71, 91, 93, 100, 110, 114,


NormalKernel, 57, 96, 176, 211, 263, 265, 303, 305, 307, 317

par, 235
parameters, 213
ParameterSet, 13, 214, 215, 222–224, 226, 289, 292, 341
ParameterSetCollection, 222, 223, 225, 290, 291, 341

pdf, 232
dpdfPNorm, 233
dpdfSquared2Norm, 233
gf, 234
plot.Distribution, 162, 234, 236, 248
plot.VectorDistribution, 236

pracma::gammap(), 139
prec, 241
print.ParameterSet, 241
ProductDistribution, 13, 14, 51, 84, 143, 192, 242, 256, 310, 325
properties, 247
qqplot, 248
quantile, 248
quantile.Distribution, 249
Quartic, 57, 96, 176, 213, 250, 263, 265, 303, 305, 307, 317
R6, 9, 15, 20, 24, 26, 31, 36, 43, 47, 51, 61, 65, 68, 73, 81, 83, 88, 92, 97, 106, 111,
INDEX

115, 118, 123, 130, 135, 137, 142, 145, 149, 153, 157, 167, 171, 175, 177, 181, 188, 195, 199, 204, 208, 212, 214, 223, 228, 238, 242, 253, 257, 259, 262, 264, 270, 273, 298, 309, 312, 319, 328, 331, 335
rand, 85, 252, 265
rep.Distribution, 256
sample, 30, 87, 91, 335
SDistribution, 164, 256
set.seed, 85
set.seed(), 265
set6::Set, 133, 216, 224, 277, 310
setParameterValue, 257
Sigmoid, 57, 96, 176, 213, 251, 261, 265, 303, 305, 307, 317
Silverman, 57, 96, 176, 213, 251, 263, 263, 303, 305, 307, 317
simulateEmpiricalDistribution, 87, 91, 265
skewness, 266, 267
skewnessType, 266
skewType, 101, 267
stdev, 267
strprint, 268
summary.Distribution, 275
sup, 86, 276
support, 86, 276
survival, 76, 277
survivalAntiDeriv, 278
survivalPNorm, 278
SymmetricTriangular (Triangular), 297
symmetry, 279
testContinuous, 279
testDiscrete, 280
testDistribution, 281
testDistributionList, 282
testLeptokurtic, 283
testMatrixvariate, 284
testMesokurtic, 285
testMixture, 286
testMultivariate, 286
testNegativeSkew, 287
testNoSkew, 288
testParameterSet, 289
testParameterSetCollection, 290
testParameterSetCollectionList, 291
testParameterSetList, 292
testPlatykurtic, 293
testPositiveSkew, 294
testSymmetric, 295
testUnimivariate, 295
traits, 296
TriangularKernel, 57, 96, 176, 213, 251, 263, 265, 303, 305, 307, 317
Tricube, 57, 96, 176, 213, 251, 263, 265, 303, 304, 307, 317
Triweight, 57, 96, 176, 213, 251, 263, 265, 303, 305, 306, 317
truncate, 308
TruncatedDistribution, 51, 84, 143, 192, 246, 308, 308, 325
type, 310
UniformKernel, 57, 96, 176, 213, 251, 263, 265, 303, 305, 307, 315
valueSupport, 317
variance, 317
variateForm, 318
workingSupport, 340
wrappedModels, 340