Package 'distr6'

August 28, 2019

Title The Complete R6 Probability Distributions Interface

Version 1.1.0

Description An R6 object oriented distributions package. Unified interface for 37 probability distributions and 11 kernels including functionality for multiple scientific types. Additionally functionality for composite distributions and numerical imputation. Design patterns including wrappers and decorators are described in Gamma et al. (1994, ISBN:0-201-63361-2). For quick reference of probability distributions including d/p/q/r functions and results we refer to McLaughlin, M. P. (2001). Additionally Devroye (1986, ISBN:0-387-96305-7) for sampling the Dirichlet distribution, Gentle (2009) <doi:10.1007/978-0-387-98144-4> for sampling the Multivariate Normal distribution and Michael et al. (1976) <doi:10.2307/2683801> for sampling the Wald distribution.

Imports checkmate, R6, R62S3 (>= 1.3.1), GoFKernel, stats, extraDistr, utils, expint, data.table, pracma

Suggests knitr, testthat, devtools, rmarkdown, magrittr, actuar, remotes

License MIT + file LICENSE

LazyData true

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

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

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 6.1.1

Collate 'RSmisc_helpers.R' 'SetInterval_operations.R' 'Distribution.R' 'DistributionDecorator.R' 'DistributionDecorator_CoreStatistics.R' 'DistributionDecorator_ExoticStatistics.R' 'DistributionDecorator_FunctionImputation.R' 'setSymbol.R' 'SetInterval.R' 'SetInterval_Interval.R' 'SetInterval_SpecialSet.R' 'Distribution_Kernel.R' 'Distribution_SDistribution.R' 'ParameterSet.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'
'SDistribution_Arcsine.R' 'SDistribution_Bernoulli.R'
'SDistribution_Beta.R' 'SDistribution_Binomial.R'
'SDistribution_Categorical.R' 'SDistribution_Cauchy.R'
'SDistribution_ChiSquared.R' 'SDistribution_Degenerate.R'
'SDistribution_Dirichlet.R' 'SDistribution_DiscreteUniform.R'
'SDistribution_Empirical.R' 'SDistribution_Exponential.R'
'SDistribution_FDistribution.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_NegBinomal.R' 'SDistribution_Normal.R'
'SDistribution_Pareto.R' 'SDistribution_Poisson.R'
'SDistribution_Rayleigh.R' 'SDistribution_StudentT.R'
'SDistribution_Triangular.R' 'SDistribution_Uniform.R'
'SDistribution_Wald.R' 'SDistribution_Weibull.R'
'SetInterval_Set.R' 'Wrapper.R' 'Wrapper_ArrayDistribution.R'
'Wrapper_Convolution.R' 'Wrapper_HuberizedDistribution.R'
'Wrapper_MixtureDistribution.R' 'Wrapper_ProductDistribution.R'
'Wrapper_Scale.R' 'Wrapper_TruncatedDistribution.R'
'Wrapper_VectorDistribution.R' 'assertions.R'
'decomposeMixture.R' 'decorate.R' 'distr6-deprecated.R'
'distr6.R' 'distr6.news.R' 'distr6_globals.R'
'exkurtosisType.R' 'generalPNorm.R' 'getParameterSet.R'
'listDecorators.R' 'listDistributions.R' 'listKernels.R'
'listSpecialSets.R' 'listWrappers.R'
'makeUniqueDistributions.R' 'measures.R'
'simulateEmpiricalDistribution.R' 'skewType.R' 'zzz.R'

NeedsCompilation no

Author Raphael Sonabend [aut, cre] (<https://orcid.org/0000-0001-9225-4654>),
Franz Kiraly [aut],
Peter Ruckdeschel [ctb] (Author of distr),
Matthias Kohl [ctb] (Author of distr),
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 2019-08-27 23:30:16 UTC

R topics documented:

`distr6-package` *distr6: Object Oriented Distributions in R*

Description

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

Details

The main features of distr6 are:

- Currently implements 36 probability distributions (and 11 Kernels) including all distributions in the R stats package. Each distribution has (where possible) closed form analytic expressions for basic statistical methods.
- Decorators that add further functionality to probability distributions including numeric results for useful modelling functions such as p-norms and k-moments.
- Wrappers for composite distributions including convolutions, truncation, mixture distributions and product distributions.

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

```
vignette("distr6","distr6")
```

And for more advanced usage see the complete tutorials at

https://alan-turing-institute.github.io/distr6/index.html

Author(s)

Maintainer: Raphael Sonabend <raphael.sonabend.15@ucl.ac.uk> (0000-0001-9225-4654)

Authors:

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

Other contributors:

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

- Jordan Deenichin <d.deenichin@gmail.com> [contributor]
- Chengyang Gao <garoc371@gmail.com> [contributor]
- Chloe Zhaoyuan Gu <guzhaoyuan@outlook.com> [contributor]
- Yunjie He <zcakebx@ucl.ac.uk> [contributor]
- Xiaowen Huang <hxw3678@gmail.com> [contributor]
- Shuhan Liu <Shuhan.liu.99@gmail.com> [contributor]
- Runlong Yu <edwinyurl@hotmail.com> [contributor]
- Chijing Zeng <britneyzenguk@gmail.com> [contributor]
- Qian Zhou <zcakqz1@ucl.ac.uk> [contributor]

See Also

Useful links:

- https://alan-turing-institute.github.io/distr6/
- https://github.com/alan-turing-institute/distr6/
- Report bugs at https://github.com/alan-turing-institute/distr6/issues

---

`Arcsine` *Arcsine Distribution Class*

Description

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

Details

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

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

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

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

`cf` and `mgf` are omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results.

When the Standard Arcsine is constructed (default) then `rbeta` is used for sampling, otherwise via inverse transform

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Arcsine$new(lower = 0, upper = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
lower	integer	lower distribution limit.
upper	integer	upper distribution limit.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Arcsine distribution is parameterised with lower and upper as numerics.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution

variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

listDistributions for all available distributions. rbeta for the Beta distribution sampling function. CoreStatistics for numerical results.

Examples

```
x = Arcsine$new(lower = 2, upper = 5)

# Update parameters
x$setParameterValue(upper = 4, lower = 1)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`as.data.table` *Coerce ParameterSet to data.table*

Description

Coerces a ParameterSet to a data.table.

Usage

```
as.data.table(object)
```

Arguments

`object`	ParameterSet

Value

A data.table.

R6 Usage

$as.data.table()

See Also

`ParameterSet`

as.numeric.Interval *Coerces Interval to Numeric*

Description

If possible coerces an integer-class to numeric.

Details

If the interval is of class 'integer', returns the interval as a numeric. Otherwise does nothing.

This is an R6 method only, no S3 dispatch is available.

Value

If the Interval is of class integer then returns a numeric vector of the interval elements.

R6 Usage

$length()

See Also

Set, length.Interval

as.ParameterSet *Coerce to a ParameterSet*

Description

Coerces objects to ParameterSet.

Usage

```
as.ParameterSet(x,...)

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

## S3 method for class 'list'
as.ParameterSet(x, ...)
```

Arguments

x	object
...	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

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,

$$f(x) = p,\ if\ x = 1$$

$$f(x) = 1 - p,\ if\ x = 0$$

for $p \in [0, 1]$.

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

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Bernoulli$new(prob = 0.5, qprob = NULL, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
prob	numeric	probability of success.
qprob	numeric	probability of failure.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Bernoulli distribution is parameterised with `prob` or `qprob` as a number between 0 and 1. These are related via,

$$qprob = 1 - prob$$

If `qprob` is given then prob is ignored.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`
`support()`	`support`
`symmetry()`	`symmetry`
`sup()`	`sup`
`inf()`	`inf`
`dmax()`	`dmax`
`dmin()`	`dmin`
`skewnessType()`	`skewnessType`
`kurtosisType()`	`kurtosisType`

Statistical Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	`pdf`
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`cdf`
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`quantile.Distribution`
`rand(n, simplify = TRUE)`	`rand`
`mean()`	`mean.Distribution`
`variance()`	`variance`
`stdev()`	`stdev`

prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

listDistributions for all available distributions. Binomial for a generalisation of the Bernoulli distribution.

Examples

```
# Can be parameterised with probability of success or failure
```

```
Bernoulli$new(prob = 0.2)
Bernoulli$new(qprob = 0.3)

x = Bernoulli$new(verbose = TRUE) # Default is with prob = 0.5

# Update parameters
# When any parameter is updated, all others are too!
x$setParameterValue(qprob = 0.3)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`Beta` *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, α, β, is defined by the pdf,

$$f(x) = (x^{\alpha-1}(1-x)^{\beta-1})/B(\alpha, \beta)$$

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

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

`mgf` and `cf` are omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Beta$new(shape1 = 1, shape2 = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`shape1, shape2`	numeric	positive shape parameter.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Beta distribution is parameterised with `shape1` and `shape2` as positive numerics.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`
`support()`	`support`
`symmetry()`	`symmetry`
`sup()`	`sup`
`inf()`	`inf`
`dmax()`	`dmax`
`dmin()`	`dmin`
`skewnessType()`	`skewnessType`
`kurtosisType()`	`kurtosisType`

Statistical Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	`pdf`
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`cdf`
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`quantile.Distribution`
`rand(n, simplify = TRUE)`	`rand`
`mean()`	`mean.Distribution`
`variance()`	`variance`

`stdev()`	`stdev`
`prec()`	`prec`
`cor()`	`cor`
`skewness()`	`skewness`
`kurtosis(excess = TRUE)`	`kurtosis`
`entropy(base = 2)`	`entropy`
`mgf(t)`	`mgf`
`cf(t)`	`cf`
`pgf(z)`	`pgf`
`median()`	`median.Distribution`
`iqr()`	`iqr`

Parameter Methods	**Link**
`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `CoreStatistics` for numerical results.

Examples

```
x = Beta$new(shape1 = 2, shape2 = 5)
```

```
# Update parameters
x$setParameterValue(shape1 = 1)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`Binomial` *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 $p \epsilon [0, 1]$, 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.

Constructor

Binomial$new(size = 10, prob = 0.5, qprob = NULL, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`size`	numeric	number of trials.
`prob`	numeric	probability of success.
`qprob`	numeric	probability of failure.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Binomial distribution is parameterised with `size` as a whole number, and either `prob` or `qprob` as a number between 0 and 1. These are related via,

$$qprob = 1 - prob$$

If `qprob` is given then `prob` is ignored.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`
`support()`	`support`
`symmetry()`	`symmetry`
`sup()`	`sup`
`inf()`	`inf`
`dmax()`	`dmax`
`dmin()`	`dmin`
`skewnessType()`	`skewnessType`
`kurtosisType()`	`kurtosisType`

Statistical Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	`pdf`
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`cdf`
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`quantile.Distribution`
`rand(n, simplify = TRUE)`	`rand`
`mean()`	`mean.Distribution`
`variance()`	`variance`

stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

listDistributions for all available distributions.

Examples

```
# Can be parameterised with probability of success or failure
```

```
Binomial$new(prob = 0.2)
Binomial$new(qprob = 0.3)

x = Binomial$new() # Default is with prob = 0.5 and size = 10

# Update parameters
# When any parameter is updated, all others are too!
x$setParameterValue(size = 4, qprob = 0.1)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`Categorical` *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 $p_i, i = 1, \ldots, k; \sum p_i = 1$.

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

Only the mode, pdf, cdf, quantile and rand are available for this Distribution, all other methods return `NaN`. 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).

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Categorical$new(..., probs, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
...	ANY	elements in the support Set. See details.
probs	numeric	vector of probabilities. See details.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Categorical distribution is parameterised with a series of elements for the support set and probs determining the probability of each category occurring. The length of the probability list should equal the number of elements. The probability vector is automatically normalised with

$$probs = probs/sum(probs)$$

If no arguments are given, then defaults to one element '1' with probability one.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax

`dmin()`	dmin
`skewnessType()`	skewnessType
`kurtosisType()`	kurtosisType

Statistical Methods	**Link**
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	pdf
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	cdf
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	quantile.Distribution
`rand(n, simplify = TRUE)`	rand
`mean()`	mean.Distribution
`variance()`	variance
`stdev()`	stdev
`prec()`	prec
`cor()`	cor
`skewness()`	skewness
`kurtosis(excess = TRUE)`	kurtosis
`entropy(base = 2)`	entropy
`mgf(t)`	mgf
`cf(t)`	cf
`pgf(z)`	pgf
`median()`	median.Distribution
`iqr()`	iqr

Parameter Methods	**Link**
`parameters(id)`	parameters
`getParameterValue(id, error = "warn")`	getParameterValue
`setParameterValue(..., lst = NULL, error = "warn")`	setParameterValue

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	liesInSupport
`liesInType(x, all = TRUE, bound = FALSE)`	liesInType

Representation Methods	**Link**
`strprint(n = 2)`	strprint
`print(n = 2)`	print
`summary(full = T)`	summary.Distribution
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `sample` for the sampling function.

Examples

```
# Note probabilities are automatically normalised
x = Categorical$new("Bapple","Banana",2,probs=c(0.2,0.4,1))

# Only the probabilities can be changed and must the same length as in construction
x$setParameterValue(probs = c(0.1,0.2,0.7))

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

# Statistics
x$mode()

summary(x)
```

Cauchy *Cauchy Distribution Class*

Description

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

Details

The Cauchy distribution parameterised with location, α, and scale, β, is defined by the pdf,

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

for $\alpha \epsilon R$ and $\beta > 0$.

The distribution is supported on the Reals.

The mean and variance are undefined, hence `NaN` is returned.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Cauchy$new(location = 0, scale = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
location	numeric	location parameter.
scale	numeric	scale parameter.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Cauchy distribution is parameterised with location as a numeric and scale as a positive numeric.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin

skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

listDistributions for all available distributions.

Examples

```
x = Cauchy$new(location = 2, scale = 5)

# Update parameters
x$setParameterValue(scale = 3)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

---

cdf *Cumulative Distribution Function*

Description

Returns the cumulative distribution function for a distribution evaluated at a given point.

Usage

```
cdf(object, x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
```

Arguments

object	Distribution.
x1	vector of numerics to evaluate function at.
...	additional arguments.
lower.tail	logical; if TRUE (default), probabilities are $P(X \leq x)$ otherwise, $P(X > x)$.
log.p	logical; if TRUE, probabilities p are given as log(p).
simplify	if TRUE (default) returns results in simplest form (vector or data.table) otherwise as data.table.

Details

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 without warning using an analytic expression. Otherwise, if the distribution has not been decorated with `FunctionImputation`, `NULL` is returned. To impute the cdf, use `decorate(distribution,FunctionImputation)`, this will provide a numeric calculation for the cdf with warning.

Additional named arguments can be passed, which are required for composite distributions such as `ProductDistribution` and `ArrayDistribution`.

Value

Cumulative distribution function evaluated at given points as either a numeric if `simplify` is TRUE or as a data.table.

R6 Usage

$cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)

See Also

`pdf`, `quantile`, `rand` for other statistical functions. `FunctionImputation`, `decorate` for imputing missing functions.

---

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.

Details

The cdf anti-derivative is defined by

$$acdf(a,b) = \int_a^b F_X(x)dx$$

where X is the distribution, F_X is the cdf of the distribution X and a, b are the limits of integration.

Can only be used after decorating with ExoticStatistics.

Value

Antiderivative of the cdf evaluated between limits as a numeric.

R6 Usage

$cdfAntiDeriv(lower = NULL, upper = NULL)

See Also

ExoticStatistics and decorate

---

cdfPNorm *Cumulative Distribution Function P-Norm*

Description

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

Usage

```
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.

Details

The p-norm of the cdf is defined by

$$(\int_a^b |F_X|^p d\mu)^{1/p}$$

where X is the distribution, F_X is the cdf and a, b are the limits of integration.

Returns NULL if distribution is not continuous.

Can only be used after decorating with ExoticStatistics.

Value

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

R6 Usage

$cdfPNorm(object, p = 2, lower = NULL, upper = NULL)

See Also

ExoticStatistics and decorate

---

cf *Characteristic Function*

Description

Characteristic function of a distribution

Usage

```
cf(object, t)
```

Arguments

object	Distribution.
t	integer to evaluate characteristic function at.

Details

The characteristic function is defined by

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

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

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

Value

Characteristic function evaluated at t as a numeric.

R6 Usage

$cf(t)

See Also

CoreStatistics and decorate

`ChiSquared` *Chi-Squared Distribution Class*

Description

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

Details

The Chi-Squared distribution parameterised with degrees of freedom, ν, is defined by the pdf,

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

for $\nu > 0$.

The distribution is supported on the Positive Reals.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

ChiSquared$new(df = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`df`	numeric	degrees of freedom.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Chi-Squared distribution is parameterised with `df` as a positive numeric.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	Link
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `Normal` for the Normal distribution.

Examples

```
x = ChiSquared$new(df = 2)

# Update parameters
x$setParameterValue(df = 3)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

class.SetInterval *SetInterval Minimum Accessor*

Description

Returns the SetInterval minimum.

Details

This is an R6 method only, no S3 dispatch is available.

Value

One of 'numeric' or 'integer'.

R6 Usage

$min()

See Also

SetInterval

complement.SetInterval *Symbolic Complement for SetInterval*

Description

Makes a symbolic representation for the complement of sets/intervals.

Usage

```
complement.SetInterval(...)

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

Arguments

x	SetInterval
y	SetInterval
...	SetIntervals to take the complement of.

Details

This does not calculate the complement of the arguments but is just a symbolic representation using unicode.

Value

Returns an R6 object inheriting from SetInterval.

See Also

`product.SetInterval`, `union.SetInterval`, `power.SetInterval`

Examples

```
PosNaturals$new() - Interval$new(-1,10)
complement.SetInterval(Reals$new(), Interval$new(10,Inf))
```

Complex *Set of Complex Numbers*

Description

The mathematical set of complex numbers.

Details

The set of Complex numbers is defined as the set of reals with possibly imaginary components, i.e.

$$Complex = \{a + bi \mid a, b \in R\}$$

where R is the set of reals.

Value

Returns R6 object of class Complex.

Constructor

Complex$new(dim = 1)

Constructor Arguments

Argument	Type	Details
`dim`	numeric	Dimension of the set.

See Also

listSpecialSets

Examples

```
Complex$new()
Complex$new(dim = 2)
```

Convolution *Distribution Convolution Wrapper*

Description

Calculates the convolution of two distribution via numerical calculations.

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.

Constructor

Convolution$new(dist1, dist2, add = TRUE, type = NULL)

Constructor Arguments

Argument	Type	Details
dist1	distribution	First distribution in convolution.
dist2	distribution	Second distribution in convolution.
add	logical	Add or subtract distributions.
type	logical	Type of new distribution, automated if NULL.

See Also

listWrappers

CoreStatistics *Core Statistical Methods for Distributions*

Description

This decorator adds numeric methods for missing analytic expression in distr6 Distribution objects 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. See the 'Added Methods' section below to find details of the methods that are added to the Distribution. Methods already present in the distribution are not overwritten by the decorator.

Use decorate to decorate a Distribution.

All methods in this decorator use numerical approximations and therefore better results may be available from analytic computations.

Value

Returns a decorated R6 object inheriting from class SDistribution with the methods listed below added to the SDistribution methods.

Constructor

CoreStatistics$new(distribution)

Constructor Arguments

Argument	Type	Details
distribution	distribution	Distribution to decorate.

Added Methods

Method	Name	Link
mgf(t)	Moment generating function	mgf
pgf(t)	Probability generating function	pgf

`cf(t)`	Characteristic function	`cf`
`entropy(base = 2)`	(Shannon) Entropy	`entropy`
`skewness()`	Skewness	`skewness`
`kurtosis(excess = TRUE)`	Kurtosis	`kurtosis`
`kthmoment(type = "central")`	Kth Moment	`kthmoment`
`genExp(trafo)`	Generalised Expectation	`genExp`
`mode(which = "all")`	Mode	`mode`
`variance()`	Variance	`variance`
`mean()`	Arithmetic mean	`mean.Distribution`

See Also

`decorate`, `listDecorators`

Examples

```
x = Binomial$new()
decorate(x, CoreStatistics)
x$genExp()

x = Binomial$new(decorators = CoreStatistics)
x$kthmoment(4)
```

`correlation` *Distribution Correlation*

Description

Correlation of a distribution.

Usage

```
correlation(object)
```

Arguments

`object`	Distribution.

Details

In terms of covariance, the correlation of a distribution is defined by the equation,

$$\rho_{XY} = \sigma_{XY}/\sigma_X\sigma_Y$$

where σ_{XY} is the covariance of X and Y and σ_X, σ_Y and the respective standard deviations of X and Y.

If the distribution is univariate then returns 1.

Calculates correlation analytically from variance. If an analytic expression for variance isn't available, returns error. To impute a numeric expression, use the `CoreStatistics` decorator.

Value

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

R6 Usage

$correlation()

`Cosine` *Cosine Kernel*

Description

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

$$f(x) = (\pi/4)cos(x\pi/2)$$

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

Value

Returns an R6 object inheriting from class Kernel.

Constructor

Cosine$new(decorators = NULL)

Constructor Arguments

Argument	Type	Details
`decorators`	Decorator	decorators to add functionality.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	Link
prec()	prec
stdev()	stdev
mode()	mode
mean()	mean.Distribution
median()	median.Distribution
iqr()	iqr
correlation()	correlation

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

---

cumHazard *Cumulative Hazard Function*

Description

The cumulative hazard function of a probability distribution is the anti-derivative of the hazard function.

Usage

```
cumHazard(object, x1, log = FALSE)
```

Arguments

object	Distribution.
x1	Point to evaluate the cumulative hazard function at.
log	logical, if TRUE then the (natural) logarithm of the cumulative hazard function is returned.

Details

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.

Can only be used after decorating with `ExoticStatistics`.

Value

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

R6 Usage

$cumHazard(x1, log = FALSE)

See Also

`ExoticStatistics` and `decorate`

`decorate` *Decorate Distributions*

Description

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

Usage

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

Arguments

`distribution`	distribution to decorate
`decorators`	list of 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.

The `decorators` parameter should either be a list of decorator classes (i.e. not as strings) or a single decorator class; see examples.

Value

Returns a decorated R6 object inheriting from class SDistribution with the methods listed from one of the available decorators added to the SDistribution methods.

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.

Examples

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

E <- Exponential$new()
decorate(E, list(CoreStatistics, ExoticStatistics))
```

---

decorators *Decorators Accessor*

Description

Returns the decorators added to a distribution.

Usage

```
decorators(object)
```

Arguments

`object`	Distribution.

Value

Character vector of decorators.

R6 Usage

$decorators()

See Also

`decorate`

`Degenerate` *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, μ is defined by the pmf,

$$f(x) = 1,\ if\ x = \mu$$

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

for $\mu \epsilon R$.

The distribution is supported on μ.

Also known as the Dirac distribution.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Degenerate$new(mean = 0, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`mean`	numeric	location parameter.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Degenerate distribution is parameterised with `mean` as a numeric.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.

package The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue

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

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions.

Examples

```
x = Degenerate$new(mean = 4)

# Update parameters
x$setParameterValue(mean = 2.56)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

dimension.SetInterval *SetInterval Dimension Accessor*

Description

Returns the SetInterval dimension.

Details

This is an R6 method only, no S3 dispatch is available.

Value

Dimension as an integer.

R6 Usage

$dimension()

See Also

SetInterval

Dirichlet *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, $\alpha_1, ..., \alpha_k$, is defined by the pdf,

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

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

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

mgf and cf are omitted as no closed form analytic expression could be found, decorate with CoreStatistics for numerical results. 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.

Constructor

Dirichlet$new(params = c(1, 1), decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`params`	numeric	vector of concentration parameters.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Dirichlet distribution is parameterised with `params` as a vector of positive numerics. The parameter `K` is automatically calculated by counting the length of the params vector, once constructed this cannot be changed.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`
`support()`	`support`
`symmetry()`	`symmetry`
`sup()`	`sup`
`inf()`	`inf`
`dmax()`	`dmax`
`dmin()`	`dmin`

skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

Devroye, Luc (1986). Non-Uniform Random Variate Generation. Springer-Verlag. ISBN 0-387-96305-7.

See Also

`listDistributions` for all available distributions. `Beta` for the Beta distribution. `CoreStatistics` for numerical results. `FunctionImputation` to numerically impute d/p/q/r.

Examples

```
# Different parameterisations
x <- Dirichlet$new(params = c(2,5,6))

# Update parameters
x$setParameterValue(params = c(3, 2, 3))
# 'K' parameter is automatically calculated
x$parameters()
## Not run:
# This errors as less than three parameters supplied
x$setParameterValue(params = c(1, 2))

## End(Not run)

# d/p/q/r
# Note the difference from R stats
x$pdf(0.1, 0.4, 0.5)
# This allows vectorisation:
x$pdf(c(0.3, 0.2), c(0.6, 0.9), c(0.9,0.1))
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

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,

$$f(x) = 1/(b - a + 1)$$

for $a, b \in Z;\ b \geq a$.

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

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

DiscreteUniform$new(lower = 0, upper = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`lower`	integer	lower distribution limit.
`upper`	integer	upper distribution limit.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Discrete Uniform distribution is parameterised with `lower` and `upper` as whole numbers.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`

type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	**Link**
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**

`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `Uniform` for the (continuous) Uniform distribution.

Examples

```
x <- DiscreteUniform$new(lower = -10, upper = 5)

# Update parameters
x$setParameterValue(lower = 2, upper = 7)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`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

```
distr6News()
```

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.

Constructor

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

Constructor Arguments

Argument	Type	Details
name	character	Full name of distribution.
short_name	character	Short name to identify distribution.
type	SetInterval	Scientific type.
support	SetInterval	Distribution support. See Details.
symmetric	logical	Is distribution symmetric?
pdf	function	See Details.
cdf	function	See Details.
quantile	function	See Details.
rand	function	See Details.
parameters	ParameterSet	See Details.
decorators	list	R6 decorators to add in construction.
valueSupport	character	continuous, discrete, mixture. See Details.
variateForm	character	univariate, multivariate, matrixvariate. See Details.
description	character	short description of distribution.

Constructor Details

The most basic Distribution object consists of a name and one of pdf/cdf.

If supplied, `type` and `support` should be given as an R6 SetInterval object. If neither are supplied then the set of Reals is taken to be the type and the dimension is the number of formal arguments in the pdf/cdf. If only `type` is supplied then this is taken to also be the support.

By default, missing `pdf`, `cdf`, `quantile` and `rand` are not automatically imputed. Use the `FunctionImputation` decorator to generate these.

See `ParameterSet` for more details on construction of a ParameterSet.

`decorators` is an optional list of decorators (R6 environments not strings) to decorate the Distribution in construction. Decorators can also be added after construction. See `decorate` for more details.

`valueSupport` should be one of continuous/discrete/mixture if supplied. `variateForm` should be one of univariate/multivariate/matrixvariate if supplied. If not given these are automatically filled from `type` and `support`.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`
`support()`	`support`
`symmetry()`	`symmetry`
`sup()`	`sup`
`inf()`	`inf`
`dmax()`	`dmax`
`dmin()`	`dmin`
`skewnessType()`	`skewnessType`
`kurtosisType()`	`kurtosisType`

d/p/q/r Methods	Link

pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	**Link**
prec()	prec
stdev()	stdev
median()	median.Distribution
iqr()	iqr
correlation()	correlation

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

See Also

See SetInterval and SpecialSet for details on Sets and Intervals. See ParameterSet for parameter details. See decorate for Decorator details.

DistributionDecorator *Abstract DistributionDecorator Class*

Description

The abstract parent class to 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.

Abstract classes cannot be implemented directly. Use the `decorate` function to decorate distributions.

Value

Returns error. Abstract classes cannot be constructed directly.

References

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

See Also

`decorate` and `listDecorators`

DistributionWrapper *Abstract DistributionWrapper Class*

Description

The abstract parent class to wrappers.

Details

Wrapping is the process of adapting the interface of a class into another (Gamma et al. 1994). After wrapping, the parameters of a distribution are prefixed with the distribution name to ensure uniqueness of parameter IDs.

Abstract classes cannot be implemented directly. Use the `listWrappers` function to see constructable wrappers.

Value

Returns error. Abstract classes cannot be constructed directly.

Public Methods

Accessor Methods	Link
wrappedModels(model = NULL)	wrappedModels
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	Link
prec()	prec
stdev()	stdev
median()	median.Distribution
iqr()	iqr
cor()	cor

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

See Also

`listWrappers`

`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

$$maximum = supremum - 1.1e - 15$$

Usage

```
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

$$minimum = infimum + 1.1e - 15$$

Usage

```
dmin(object)
```

Arguments

`object`	Distribution.

Value

Minimum as a numeric.

R6 Usage

$dmin()

See Also

support, dmax, sup, inf

elements *Set Elements Accessor*

Description

Returns the elements in a Set.

Details

This is an R6 method only, no S3 dispatch is available.

Value

Elements in the set.

R6 Usage

$elements()

See Also

Set

Empirical *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 $x_i \epsilon R, i = 1, ..., k$.

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

skewness, kurtosis, entropy, mgf, cf are omitted as no closed form analytic expression could be found, decorate with CoreStatistics for numerical results.

Sampling from this distribution is performed with the sample function with the elements given as the support set and uniform probabilities. 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.

Constructor

Empirical$new(samples, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
samples	numeric	vector of observed samples.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Empirical distribution is parameterised with a vector of elements for the support set.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	**Link**
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `sample` for the sampling function. `CoreStatistics` for numerical results.

Examples

```
x = Empirical$new(stats::runif(1000)*10)

# 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)
```

Empty *Empty Set*

Description

The mathematical Empty, or Null, set.

Details

The Empty set is the set defined by having no elements,

$$Empty = \{\}$$

Value

Returns R6 object of class Empty.

Constructor

Empty$new()

See Also

`listSpecialSets`

Examples

```
Empty$new()
```

`entropy` *Distribution Entropy*

Description

(Information) Entropy of a distribution

Usage

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

Arguments

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

Details

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. The base of the logarithm of the equation determines the type of entropy computed. By default we use base 2 to compute entropy in 'Shannons' or 'bits'.

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

Value

Entropy with given base as a numeric.

R6 Usage

$entropy(base = 2)

See Also

`CoreStatistics` and `decorate`

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.

Value

Returns an R6 object inheriting from class Kernel.

Constructor

Epanechnikov$new(decorators = NULL)

Constructor Arguments

Argument	Type	Details
decorators	Decorator	decorators to add functionality.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link

decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	**Link**
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	**Link**
prec()	prec
stdev()	stdev
mode()	mode
mean()	mean.Distribution
median()	median.Distribution
iqr()	iqr
correlation()	correlation

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
`strprint(n = 2)`	strprint
`print(n = 2)`	print
`summary(full = T)`	summary.Distribution
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

`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)
```

`ExoticStatistics` *Exotic Statistical Methods for Distributions*

Description

This decorator adds methods for more complex statistical methods including p-norms, survival and hazard functions and anti-derivatives.

Details

Decorator objects add functionality to the given Distribution object by copying methods in the decorator environment to the chosen Distribution environment. See the 'Added Methods' section below to find details of the methods that are added to the Distribution. Methods already present in the distribution are not overwritten by the decorator.

Use `decorate` to decorate a Distribution.

Methods in this decorator may use numerical approximations and therefore better results may be available from analytic computations.

Value

Returns a decorated R6 object inheriting from class SDistribution with the methods listed below added to the SDistribution methods.

Constructor

ExoticStatistics$new(distribution)

Constructor Arguments

Argument	Type	Details
`distribution`	distribution	Distribution to decorate.

Added Methods

Method	Name	Link
`survival(x1, log = FALSE)`	Survival function	`survival`
`hazard(x1, log = FALSE)`	Hazard function	`hazard`
`cumHazard(x1, log = FALSE)`	Cumulative hazard function	`cumHazard`
`cdfAntiDeriv(lower = NULL, upper = NULL))`	Anti-derivative of cdf	`cdfAntiDeriv`
`survivalAntiDeriv(lower = NULL, upper = NULL)`	Anti-derivative of survival function	`survivalAntiDeriv`
`cdfPNorm(p = 2, lower = NULL, upper = NULL)`	P-norm of cdf	`cdfPNorm`
`pdfPNorm(p = 2, lower = NULL, upper = NULL)`	P-norm of pdf	`pdfPNorm`
`survivalPNorm(p = 2, lower = NULL, upper = NULL)`	P-norm of survival function	`survivalPNorm`

See Also

decorate, listDecorators

Examples

```
x = Exponential$new()
decorate(x, ExoticStatistics)
x$survival(1)

x = Exponential$new(decorators = ExoticStatistics)
x$survival(4)
```

Exponential *Exponential Distribution Class*

Description

Mathematical and statistical functions for the Exponential distribution, which is commonly used to model inter-arrival times in a Poisson process and has the memoryless property.

Details

The Exponential distribution parameterised with rate, λ, is defined by the pdf,

$$f(x) = \lambda exp(-x\lambda)$$

for $\lambda > 0$.

The distribution is supported on the Positive Reals.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Exponential$new(rate = NULL, scale = NULL, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
rate	numeric	arrival rate.
scale	numeric	scale parameter.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Exponential distribution is parameterised with `rate` or `scale` as positive numerics. These are related via,

$$scale = 1/rate$$

If `scale` is given then `rate` is ignored.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`
`support()`	`support`
`symmetry()`	`symmetry`
`sup()`	`sup`
`inf()`	`inf`
`dmax()`	`dmax`
`dmin()`	`dmin`
`skewnessType()`	`skewnessType`
`kurtosisType()`	`kurtosisType`

Statistical Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	`pdf`
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`cdf`
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`quantile.Distribution`
`rand(n, simplify = TRUE)`	`rand`
`mean()`	`mean.Distribution`
`variance()`	`variance`
`stdev()`	`stdev`

`prec()`	`prec`
`cor()`	`cor`
`skewness()`	`skewness`
`kurtosis(excess = TRUE)`	`kurtosis`
`entropy(base = 2)`	`entropy`
`mgf(t)`	`mgf`
`cf(t)`	`cf`
`pgf(z)`	`pgf`
`median()`	`median.Distribution`
`iqr()`	`iqr`

Parameter Methods	**Link**
`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions.

Examples

```
Exponential$new(rate = 4)
Exponential$new(scale = 3)
```

```
x = Exponential$new(verbose = TRUE) # Default is rate = 1

# Update parameters
# When any parameter is updated, all others are too!
x$setParameterValue(scale = 2)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`ExtendedReals` *Set of Extended Reals*

Description

The mathematical set of extended real numbers.

Details

The set of Extended Reals is defined as the union of the set of reals with +-Infinity, i.e.

$$ExtendedReals = R \cup \{-\infty, \infty\}$$

where R is the set of reals.

Value

Returns R6 object of class ExtendedReals.

Constructor

ExtendedReals$new(dim = 1)

Constructor Arguments

Argument	Type	Details
`dim`	numeric	Dimension of the set.

See Also

listSpecialSets

Examples

```
ExtendedReals$new()
ExtendedReals$new(dim = 2)
```

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, μ, ν, is defined by the pdf,

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

for $\mu, \nu > 0$.

The distribution is supported on the Positive Reals.

`cf` is omitted as no closed form analytic expression could be found, decorate with CoreStatistics for numerical results.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

FDistribution$new(df1 = 1, df2 = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`df1, df2`	numeric	degrees of freedom.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The 'F' distribution is parameterised with `df1` and `df2` as positive numerics.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	decorators
`traits()`	traits
`valueSupport()`	valueSupport
`variateForm()`	variateForm
`type()`	type
`properties()`	properties
`support()`	support
`symmetry()`	symmetry
`sup()`	sup
`inf()`	inf
`dmax()`	dmax
`dmin()`	dmin
`skewnessType()`	skewnessType
`kurtosisType()`	kurtosisType

Statistical Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	pdf
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	cdf
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	quantile.Distribution
`rand(n, simplify = TRUE)`	rand
`mean()`	mean.Distribution
`variance()`	variance
`stdev()`	stdev
`prec()`	prec
`cor()`	cor
`skewness()`	skewness
`kurtosis(excess = TRUE)`	kurtosis
`entropy(base = 2)`	entropy
`mgf(t)`	mgf

`cf(t)`	`cf`
`pgf(z)`	`pgf`
`median()`	`median.Distribution`
`iqr()`	`iqr`

Parameter Methods	**Link**
`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `Normal` and `ChiSquared` for the Normal and Chi-Squared distributions. `CoreStatistics` for numerical results.

Examples

```
x <- FDistribution$new(df1 = 1, df2 = 3)

# Update parameters
x$setParameterValue(df2 = 10)
x$parameters()

# d/p/q/r
```

```
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`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, α, scale, β, and minimum, γ, is defined by the pdf,

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

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

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

`mgf` and `cf` are omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results.

Also known as the Inverse Weibull distribution.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Frechet$new(shape = 1, scale = 1, minimum = 0, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`shape`	numeric	shape parameter.
`scale`	numeric	scale parameter.
`minimum`	numeric	location parameter.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Frechet distribution is parameterised with `shape`, `scale` as positive numerics and `minimum` as a numeric.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	decorators
`traits()`	traits
`valueSupport()`	valueSupport
`variateForm()`	variateForm
`type()`	type
`properties()`	properties
`support()`	support
`symmetry()`	symmetry
`sup()`	sup
`inf()`	inf
`dmax()`	dmax
`dmin()`	dmin
`skewnessType()`	skewnessType
`kurtosisType()`	kurtosisType

Statistical Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	pdf
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	cdf
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	quantile.Distribution
`rand(n, simplify = TRUE)`	rand
`mean()`	mean.Distribution
`variance()`	variance
`stdev()`	stdev
`prec()`	prec
`cor()`	cor
`skewness()`	skewness

kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

listDistributions for all available distributions. Gumbel and Weibull for other special cases of the generalized extreme value distribution. CoreStatistics for numerical results.

Examples

```
x = Frechet$new(shape = 2, scale = 3, minimum = 6)

# Update parameters
x$setParameterValue(shape = 3)
```

```
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`FunctionImputation` *Imputed Pdf/Cdf/Quantile/Rand Functions*

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.

Details

Decorator objects add functionality to the given Distribution object by copying methods in the decorator environment to the chosen Distribution environment. See the 'Added Methods' section below to find details of the methods that are added to the Distribution. Methods already present in the distribution are not overwritten by the decorator.

Use `decorate` to decorate a Distribution.

All methods in this decorator use numerical approximations and therefore better results may be available from analytic computations.

Value

Returns a decorated R6 object inheriting from class SDistribution with d/p/q/r numerically imputed if previously missing.

Constructor

FunctionImputation$new(distribution)

Constructor Arguments

Argument	Type	Details
`distribution`	distribution	Distribution to decorate.

Added Methods

Method	Name	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	Density/mass function	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	Distribution function	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	Quantile function	quantile.Distribu
rand(n, simplify = TRUE)	Simulation function	rand

See Also

decorate, listDecorators

Examples

```
x = Distribution$new("Test", pdf = function(x) 1/(4-1),
support = Interval$new(1,4),
type = Reals$new())
decorate(x, FunctionImputation)
plot(x$pdf(0:5))
plot(x$cdf(0:5))

x = Distribution$new("Test", pdf = function(x) 1/(4-1),
decorators = ExoticStatistics)
x$cdf(1)
```

Gamma *Gamma Distribution Class*

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, α, and rate, β, is defined by the pdf,

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

for $\alpha, \beta > 0$.

The distribution is supported on the Positive Reals.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Gamma$new(shape = 1, rate = 1, scale = NULL, mean = NULL, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
shape	numeric	shape parameter.
rate	numeric	inverse scale parameter.
scale	numeric	scale parameter.
mean	numeric	alternate scale parameter.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Gamma distribution is parameterised with shape and either rate, scale or mean, all as positive numerics. These are related via,

$$scale = 1/rate$$

$$mean = shape/rate$$

If mean is given then rate and scale are ignored. If scale is given then rate is ignored.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type

properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	**Link**
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**
strprint(n = 2)	strprint

`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions.

Examples

```
Gamma$new(shape = 1, rate = 2)
Gamma$new(shape = 1, scale = 4)
Gamma$new(shape = 1, mean = 0.5)

# Default is shape = 1, rate = 1
x = Gamma$new(verbose = TRUE)

# Update parameters
# When any parameter is updated, all others are too!
x$setParameterValue(scale = 2)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`generalPNorm` *Generalised P-Norm*

Description

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

Usage

```
generalPNorm(fun, p, lower, upper)
```

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

Details

The p-norm of a function f is given by,

$$(\int_S |f|^p d\mu)^{1/p}$$

where S is the function support.

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

```
generalPNorm(Exponential$new()$pdf,2,0,10)
```

---

genExp *Generalised Expectation of a Distribution*

Description

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

Usage

```
genExp(object, trafo = NULL)
```

Arguments

object	Distribution.
trafo	transformation for expectation calculation, see details.

Details

The expectation of a probability distribution can be numerically calculated in a variety of different ways, some more efficient than others depending on what is available, this function first checks which analytic methods are present before selecting a numeric strategy.

If trafo = NULL, then the arithmetic mean is calculated, i.e. the approximation to $E[X]$. Any transformation must be given as a function, for example `trafo = function(x) x^2` (which is the second moment).

Can only be used after decorating with `CoreStatistics`.

Value

The given expectation as a numeric, otherwise NULL.

R6 Usage

$genExp(trafo = NULL)

See Also

`mean`, `CoreStatistics` and `decorate`.

Geometric *Geometric Distribution Class*

Description

Mathematical and statistical functions for the Geometric distribution, which is commonly used to model the number of trials (or number of failures) before the first success.

Details

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

$$f(x) = (1-p)^{k-1}p$$

for $p \epsilon [0, 1]$.

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.

Constructor

Geometric$new(prob = 0.5, qprob = NULL, trials = FALSE, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
prob	numeric	probability of success.
qprob	numeric	probability of failure.
trials	logical	number of trials or failures, see details.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Geometric distribution is parameterised with prob or qprob as a number between 0 and 1. These are related via,

$$qprob = 1 - prob$$

If qprob is given then prob is ignored.

The logical parameter trials determines which Geometric distribution is constructed and cannot be changed after construction. If trials is TRUE then the Geometric distribution that models the number of trials, x, before the first success is constructed. Otherwise the Geometric distribution calculates the probability of y failures before the first success. Mathematically these are related by $Y = X - 1$.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf

dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	**Link**
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

listDistributions for all available distributions.

Examples

```
# Different parameterisations
Geometric$new(prob = 0.2)
Geometric$new(qprob = 0.7)

# Different forms of the distribution
Geometric$new(trials = TRUE) # Number of trials before first success
Geometric$new(trials = FALSE) # Number of failures before first success

# Use description to see which form is used
Geometric$new(trials = TRUE)$description
Geometric$new(trials = FALSE)$description

# Default is prob = 0.5 and number of failures before first success
x <- Geometric$new()

# Update parameters
# When any parameter is updated, all others are too!
x$setParameterValue(qprob = 0.2)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

---

getParameterSupport *Parameter Support Accessor*

Description

Returns the support of the given parameter.

Usage

```
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.

Details

Returns NULL and warning if the given parameter is not in the Distribution, otherwise returns the support of the given parameter as a SetInterval object.

stopwarn either breaks the code with an error if "error" is given or returns NULL with warning otherwise.

Value

An R6 object of class SetInterval.

R6 Usage

$getParameterSupport(id, error = "warn")

See Also

parameters and SetInterval

---

getParameterValue *Parameter Value Accessor*

Description

Returns the value of the given parameter.

Usage

```
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.

Details

Returns NULL and warning if the given parameter is not in the Distribution, otherwise returns the value of the given parameter.

`stopwarn` either breaks the code with an error if "error" is given or returns `NULL` with warning otherwise.

Value

The current value of a given parameter as a numeric.

R6 Usage

$getParameterValue(id, error = "warn")

See Also

`parameters` and `setParameterValue`

`getSymbol.SetInterval` *SetInterval Symbol Accessor*

Description

Returns the SetInterval symbol.

Details

This is an R6 method only, no S3 dispatch is available.

Value

Returns string representation of a SetInterval.

R6 Usage

$getSymbol()

See Also

`SetInterval`

`Gompertz` *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, α, and scale, β, 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.

`mean`, `var`, `mgf`, `cf`, `entropy`, `skewness` and `kurtosis` are omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results.

Unfortunately the Gompertz distribution is quite complex to deal with and as such no closed form expressions exist for its mathematical and statistical properties.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Gompertz$new(shape = 1, scale = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`shape`	numeric	positive shape parameter.
`scale`	numeric	positive scale parameter.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Gompertz distribution is parameterised with `shape` and `scale` as positive numerics.

Public Variables

Variable	Return

name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	Link
`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	Link
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	Link
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `CoreStatistics` for numerical results.

Examples

```
x <- Gompertz$new(shape = 2, scale = 3)

# Update parameters
x$setParameterValue(scale = 1)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

summary(x)
```

`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, μ, and scale, β, is defined by the pdf,

$$f(x) = exp(-(z + exp(-z)))/\beta$$

for $z = (x - \mu)/\beta$, $\mu \epsilon R$ and $\beta > 0$.

The distribution is supported on the Reals.

Apery's Constant to 16 significant figures is used in the skewness calculation. The `gammaz` function from the `pracma` package is used in the `cf` to allow complex inputs.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Gumbel$new(location = 0, scale = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`location`	numeric	location parameter.
`scale`	numeric	scale parameter.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Gumbel distribution is parameterised with `location` as a numeric and `scale` as a positive numeric.

Public Variables

Variable	Return

`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	**Link**
`decorators()`	decorators
`traits()`	traits
`valueSupport()`	valueSupport
`variateForm()`	variateForm
`type()`	type
`properties()`	properties
`support()`	support
`symmetry()`	symmetry
`sup()`	sup
`inf()`	inf
`dmax()`	dmax
`dmin()`	dmin
`skewnessType()`	skewnessType
`kurtosisType()`	kurtosisType

Statistical Methods	**Link**
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	pdf
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	cdf
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	quantile.Distribution
`rand(n, simplify = TRUE)`	rand
`mean()`	mean.Distribution
`variance()`	variance
`stdev()`	stdev
`prec()`	prec
`cor()`	cor
`skewness()`	skewness
`kurtosis(excess = TRUE)`	kurtosis
`entropy(base = 2)`	entropy
`mgf(t)`	mgf
`cf(t)`	cf
`pgf(z)`	pgf
`median()`	median.Distribution
`iqr()`	iqr

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

listDistributions for all available distributions. Frechet and Weibull for other special cases of the generalized extreme value distribution. gammaz for the references for the gamma function with complex inputs.

Examples

```
x = Gumbel$new(location = 2, scale = 5)

# Update parameters
x$setParameterValue(scale = 3)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
```

```
x$variance()

summary(x)
```

`hazard` *Hazard Function*

Description

The hazard function of a probability distribution is the risk of instantaneous event at a point x.

Usage

```
hazard(object, x1, log = FALSE)
```

Arguments

`object`	Distribution.
`x1`	Point to evaluate the hazard function at.
`log`	logical, if TRUE then the (natural) logarithm of the hazard function is returned.

Details

The hazard function is defined analytically 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.

Can only be used after decorating with `ExoticStatistics`.

Value

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

R6 Usage

$hazard(x1, log = FALSE)

See Also

`ExoticStatistics` and `decorate`

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

Huberizes a distribution at lower and upper limits, using the formula

$f_H(x) = F(x), if x \leq lower$

$f_H(x) = f(x), if lower < x < upper$

$f_H(x) = F(x), if x \geq upper$

where f_H is the pdf of the truncated distribution H = Huberize(X, lower, upper) and f_X/F_X is the pdf/cdf of the original distribution.

If lower or upper are NULL they are taken to be `self$inf()` and `self$sup()` respectively.

The pdf and cdf of the distribution are required for this wrapper, if unavailable decorate with `FunctionImputation` first.

Value

Returns an R6 object of class HuberizedDistribution.

Constructor

HuberizedDistribution$new(distribution, lower = NULL, upper = NULL)

Constructor Arguments

Argument	Type	Details
`distribution`	distribution	Distribution to huberize.
`lower`	numeric	Lower limit for huberization.
`upper`	numeric	Upper limit for huberization.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`wrappedModels(model = NULL)`	`wrappedModels`
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`
`support()`	`support`
`symmetry()`	`symmetry`
`sup()`	`sup`
`inf()`	`inf`
`dmax()`	`dmax`
`dmin()`	`dmin`
`skewnessType()`	`skewnessType`
`kurtosisType()`	`kurtosisType`

d/p/q/r Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	`pdf`
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`cdf`

`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	quantile.Distribution
`rand(n, simplify = TRUE)`	rand

Statistical Methods	**Link**
`prec()`	prec
`stdev()`	stdev
`median()`	median.Distribution
`iqr()`	iqr
`cor()`	cor

Parameter Methods	**Link**
`parameters(id)`	parameters
`getParameterValue(id, error = "warn")`	getParameterValue
`setParameterValue(..., lst = NULL, error = "warn")`	setParameterValue

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	liesInSupport
`liesInType(x, all = TRUE, bound = FALSE)`	liesInType

Representation Methods	**Link**
`strprint(n = 2)`	strprint
`print(n = 2)`	print
`summary(full = T)`	summary.Distribution
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

See Also

listWrappers, FunctionImputation, huberize

Examples

```
hubBin <- HuberizedDistribution$new(
          Binomial$new(prob = 0.5, size = 10),
          lower = 2, upper = 4)
hubBin$getParameterValue("prob")
hubBin$pdf(2)
```

`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,

$$f(x) = C(K, x)C(N - K, n - x)/C(N, n)$$

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

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

`mgf` and `cf` are omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Hypergeometric$new(size = 10, successes = 5, failures = NULL, draws = 2, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`size`	numeric	population size.
`successes`	numeric	number of population successes.
`failures`	numeric	number of population failures.
`draws`	numeric	number of draws.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Hypergeometric distribution is parameterised with `size` and `draws` as positive whole numbers, and either `successes` or `failures` as positive whole numbers. These are related via,

$$failures = size - successes$$

If `failures` is given then `successes` is ignored.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`
`support()`	`support`
`symmetry()`	`symmetry`
`sup()`	`sup`
`inf()`	`inf`
`dmax()`	`dmax`
`dmin()`	`dmin`
`skewnessType()`	`skewnessType`
`kurtosisType()`	`kurtosisType`

Statistical Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	`pdf`
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`cdf`
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`quantile.Distribution`
`rand(n, simplify = TRUE)`	`rand`
`mean()`	`mean.Distribution`
`variance()`	`variance`
`stdev()`	`stdev`
`prec()`	`prec`
`cor()`	`cor`

`skewness()`	`skewness`
`kurtosis(excess = TRUE)`	`kurtosis`
`entropy(base = 2)`	`entropy`
`mgf(t)`	`mgf`
`cf(t)`	`cf`
`pgf(z)`	`pgf`
`median()`	`median.Distribution`
`iqr()`	`iqr`

Parameter Methods	**Link**
`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `CoreStatistics` for numerical results.

Examples

```
Hypergeometric$new(size = 10, successes = 7, draws = 5)
Hypergeometric$new(size = 10, failures = 3, draws = 5)

# Default is size = 50, successes = 5, draws = 10
```

```
x = Hypergeometric$new(verbose = TRUE)

# Update parameters
# When any parameter is updated, all others are too!
x$setParameterValue(failures = 10)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`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()

See Also

`support`, `dmax`, `dmin`, `sup`

inf.SetInterval *SetInterval Infimum Accessor*

Description

Returns the SetInterval infimum.

Details

This is an R6 method only, no S3 dispatch is available.

Value

Infimum as a numeric.

R6 Usage

$inf()

See Also

SetInterval

Integers *Set of Integers*

Description

The mathematical set of integers.

Details

The set of Integers is defined as the set of numbers that can be written without a fractional component, i.e.

$$Integers = \{..., -3, -2, -1, 0, 1, 2, 3, ...\}$$

Value

Returns R6 object of class Integers.

Constructor

Integers$new(dim = 1,...)

Constructor Arguments

Argument	Type	Details
dim	numeric	Dimension of the set.
...	ANY	Additional arguments.

Constructor Details

Generally the ... argument should be ignored, its primary use-case is for the child-classes.

See Also

listSpecialSets

Examples

```
Integers$new()
Integers$new(dim = 2)
```

Interval *R6 Generalised Class for Symbolic Intervals*

Description

A symbolic R6 Interval class.

Details

Intervals are distinguished from sets in R6 as they can take an infinite range and are defined over a continuous range (albeit integer or numeric).

Value

Returns an R6 object of class Interval.

Constructor

Interval$new(lower = -Inf, upper = Inf, type = "[]", class = "numeric", dim = 1)

Constructor Arguments

Argument	Type	Details
lower	numeric	Lower limit of interval.
upper	numeric	Upper limit of interval.
type	character	Interval type, one of (), (], [), [].
class	character	Atomic class, one of "numeric" or "integer".
dim	integer	Dimension of SetInterval.

Public Methods

Accessor Methods	Link
type()	type.SetInterval
dimension()	dimension.SetInterval
max()	max.SetInterval
min()	min.SetInterval
sup()	sup.SetInterval
inf()	inf.SetInterval
getSymbol()	getSymbol.SetInterval
class()	class.SetInterval

Interval Methods	Link
length()	length.Interval
as.numeric()	as.numeric.Interval

Validation Methods	Link
liesInSetInterval(x, all = FALSE, bound = FALSE)	liesInSetInterval

Representation Methods	Link
print()	print

See Also

Set

InverseGamma *Inverse Gamma Distribution Class*

Description

Mathematical and statistical functions for the Inverse Gamma distribution, which is commonly used in Bayesian statistics as the posterior distribution from the unknown variance in a Normal distribution.

Details

The Inverse Gamma distribution parameterised with shape, α, and scale, β, is defined by the pdf,

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

for $\alpha, \beta > 0$, where Γ is the gamma function.

The distribution is supported on the Positive Reals.

`cf` is omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results.

The distribution is implemented by interfacing the `extraDistr` package.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

InverseGamma$new(shape = 1, scale = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`shape`	numeric	shape parameter.
`scale`	numeric	scale parameter.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Inverse Gamma distribution is parameterised with `shape` and `scale` as positive numerics.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link

decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	**Link**
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `InvGamma` for the d/p/q/r implementation. `CoreStatistics` for numerical results.

Examples

```
x  = InverseGamma$new(shape = 1, scale = 4)

# Update parameters
# When any parameter is updated, all others are too!
x$setParameterValue(scale = 2)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

iqr *Distribution Interquartile Range*

Description

Interquartile range of a distribution

Usage

```
iqr(object)
```

Arguments

object	Distribution.

Details

The interquartile range of a distribution is defined by

$$iqr_X = q(0.75) - q(0.25)$$

where q is the quantile, or inverse distribution function.

Returns error if the quantile function is missing.

Value

Interquartile range of distribution as a numeric.

R6 Usage

$iqr()

Kernel *Abstract Kernel Class*

Description

Abstract class that cannot be constructed directly. See `listKernels` for a list of implemented kernels.

Value

Returns error. Abstract classes cannot be constructed directly.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	Link
prec()	prec
stdev()	stdev
mode()	mode
mean()	mean.Distribution
median()	median.Distribution
iqr()	iqr
correlation()	correlation

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

See Also

`listKernels`

`kthmoment` *Kth Moment*

Description

Kth standardised or central moment of a distribution

Usage

```
kthmoment(object, k, type = "central")
```

Arguments

`object`	Distribution.
`k`	the kth moment to calculate
`type`	one of 'central', 'standard' or 'raw', abbreviations allowed

Details

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, μ is the mean of the distribution and σ is the standard deviation of the distribution.

Abbreviations for the type are allowed but if an unfamiliar input is given then the central moment is computed.

Can only be used after decorating with `CoreStatistics`.

Value

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

R6 Usage

$kthmoment(k, type = "central")

See Also

`CoreStatistics` and `decorate`

`kurtosis` *Distribution Kurtosis*

Description

Kurtosis of a distribution

Usage

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

Arguments

`object`	Distribution.
`excess`	logical, if TRUE (default) excess Kurtosis returned

Details

The kurtosis of a distribution is defined by the fourth standardised moment of the distribution,

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

where E_X is the expectation of distribution X, μ is the mean of the distribution and σ is the standard deviation of the distribution. Excess Kurtosis is Kurtosis - 3.

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

Value

Kurtosis as a numeric.

R6 Usage

$kurtosis(excess = TRUE)

See Also

`CoreStatistics` and `decorate`

---

kurtosisType *Type of Kurtosis Accessor*

Description

Returns the type of kurtosis (in relation to Normal distribution)

Usage

```
kurtosisType(object)
```

Arguments

`object`	Distribution.

Value

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

R6 Usage

$kurtosisType()

See Also

`kurtosis`, `properties` and `skewnessType`

`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,

$$f(x) = exp(-|x - \mu|/\beta)/(2\beta)$$

for $\mu \epsilon R$ and $\beta > 0$.

The distribution is supported on the Reals.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Laplace$new(mean = 0, scale = 1, var = NULL, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`mean`	numeric	location parameter.
`scale`	numeric	scale parameter.
`var`	numeric	alternate scale parameter.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Laplace distribution is parameterised with `mean` as a numeric and either `scale` or `var` as positive numerics. These are related via,

$$var = 2 * scale^2$$

If `var` is given then scale is ignored.

Public Variables

Variable	Return

name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	Link
`parameters(id)`	parameters
`getParameterValue(id, error = "warn")`	getParameterValue
`setParameterValue(..., lst = NULL, error = "warn")`	setParameterValue

Validation Methods	Link
`liesInSupport(x, all = TRUE, bound = FALSE)`	liesInSupport
`liesInType(x, all = TRUE, bound = FALSE)`	liesInType

Representation Methods	Link
`strprint(n = 2)`	strprint
`print(n = 2)`	print
`summary(full = T)`	summary.Distribution
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

listDistributions for all available distributions.

Examples

```
Laplace$new(scale = 2)
Laplace$new(var = 4)

x = Laplace$new(verbose = TRUE) # Default is mean = 0, scale = 1

# Update parameters
# When any parameter is updated, all others are too!
x$setParameterValue(var = 2)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)
```

```
# Statistics
x$mean()
x$variance()

summary(x)
```

`length.Interval` *Length of Interval*

Description

Returns the length of the Interval after coercing to numeric.

Details

If the interval is of class 'numeric', returns Inf. Otherwise coerces to numeric and returns that length.

Value

Returns the length of interval as a numeric if class 'integer' otherwise Inf.

This is an R6 method only, no S3 dispatch is available.

R6 Usage

$length()

See Also

`Set`, `length.Interval`

`length.Set` *Length of Set*

Description

Returns the length of the Set as the number of elements.

Details

This is an R6 method only, no S3 dispatch is available.

Value

Number of elements in the set.

R6 Usage

$length()

See Also

Set

`liesInSetInterval` *Test if Data Lies in SetInterval.*

Description

Tests if the given data lies in the SetInterval, either tests if all data lies in the type or any of it, can choose if bounds should be included.

Arguments

x	vector of numerics to test.
all	logical, see details.
bound	logical, if FALSE (default) tests against dmin/dmax otherwise inf/sup.

Details

If `all` is TRUE (default) returns TRUE only if every element in x lies in the type. If `all` is FALSE then returns a vector of logicals for each corresponding element in the vector x.

If called on a set, then the bound argument is ignored and returns TRUE if x is an element in the set.

This is an R6 method only, no S3 dispatch is available.

Value

Either a vector of logicals if `all` is FALSE otherwise returns TRUE if every element lies in the SetInterval or FALSE otherwise.

R6 Usage

$liesInSetInterval(x, all = FALSE, bound = FALSE)

See Also

SetInterval

`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.

Details

If `all` is TRUE (default) returns TRUE only if every element in `x` lies in the support. If `all` is FALSE then returns a vector of logicals for each corresponding element in the vector `x`.

Value

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

R6 Usage

$liesInSupport(x, all = TRUE, bound = FALSE)

See Also

`liesInType`

`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.

Details

If `all` is `TRUE` (default) returns `TRUE` only if every element in `x` lies in the type. If `all` is `FALSE` then returns a vector of logicals for each corresponding element in the vector `x`.

Value

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

R6 Usage

$liesInType(x, all = TRUE, bound = FALSE)

See Also

`liesInSupport`

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 `Decorator` 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 and implemented package.

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 = "distr6"))
```

---

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()
```

listSpecialSets *Lists Implemented R6 Special Sets*

Description

Lists special sets that can be used in SetInterval.

Usage

```
listSpecialSets(simplify = FALSE)
```

Arguments

simplify	logical. If FALSE (default) returns data.table of set name and symbol, otherwise set names as characters.

Value

Either a list of characters (if simplify is TRUE) or a data.table of SpecialSets and their traits.

See Also

SpecialSet

Examples

```
listSpecialSets()
listSpecialSets(TRUE)
```

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.

See Also

DistributionWrapper

Examples

```
listWrappers()
listWrappers(TRUE)
```

Logarithmic *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, θ, is defined by the pmf,

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

for $0 < \theta < 1$.

The distribution is supported on $1, 2, 3, \ldots$.

entropy is omitted as no closed form analytic expression could be found, decorate with CoreStatistics for numerical results.

The distribution is implemented by interfacing the extraDistr package.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Logarithmic$new(theta = 0.5, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
theta	numeric	theta parameter.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Logarithmic distribution is parameterised with `theta` as a number between 0 and 1.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`
`support()`	`support`
`symmetry()`	`symmetry`
`sup()`	`sup`
`inf()`	`inf`
`dmax()`	`dmax`
`dmin()`	`dmin`
`skewnessType()`	`skewnessType`
`kurtosisType()`	`kurtosisType`

Statistical Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	`pdf`
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`cdf`
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`quantile.Distribution`
`rand(n, simplify = TRUE)`	`rand`
`mean()`	`mean.Distribution`
`variance()`	`variance`
`stdev()`	`stdev`
`prec()`	`prec`
`cor()`	`cor`
`skewness()`	`skewness`
`kurtosis(excess = TRUE)`	`kurtosis`
`entropy(base = 2)`	`entropy`
`mgf(t)`	`mgf`

`cf(t)`	`cf`
`pgf(z)`	`pgf`
`median()`	`median.Distribution`
`iqr()`	`iqr`

Parameter Methods	**Link**
`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `LogSeries` for the d/p/q/r implementation. `CoreStatistics` for numerical results.

Examples

```
x = Logarithmic$new(theta = 0.2)

# Update parameters
x$setParameterValue(theta = 0.3)
x$parameters()

# d/p/q/r
```

```
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`Logistic` *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, μ, and scale, s, is defined by the pdf,

$$f(x) = exp(-(x-\mu)/s)/(s(1+exp(-(x-\mu)/s))^2)$$

for $\mu \epsilon R$ and $s > 0$.

The distribution is supported on the Reals.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Logistic$new(mean = 0, scale = 1, sd = NULL, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`mean`	numeric	location parameter.
`scale`	numeric	scale parameter.
`sd`	numeric	standard deviation, alternate scale parameter.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Logistic distribution is parameterised with `mean` as a numeric and either `scale` or `sd` as positive numerics. These are related via,

$$sd = scale * \pi / \sqrt{(3)}$$

If `sd` is given then scale is ignored.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`
`support()`	`support`
`symmetry()`	`symmetry`
`sup()`	`sup`
`inf()`	`inf`
`dmax()`	`dmax`
`dmin()`	`dmin`
`skewnessType()`	`skewnessType`
`kurtosisType()`	`kurtosisType`

Statistical Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	`pdf`
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`cdf`
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`quantile.Distribution`
`rand(n, simplify = TRUE)`	`rand`
`mean()`	`mean.Distribution`
`variance()`	`variance`
`stdev()`	`stdev`
`prec()`	`prec`
`cor()`	`cor`
`skewness()`	`skewness`

`kurtosis(excess = TRUE)`	`kurtosis`
`entropy(base = 2)`	`entropy`
`mgf(t)`	`mgf`
`cf(t)`	`cf`
`pgf(z)`	`pgf`
`median()`	`median.Distribution`
`iqr()`	`iqr`

Parameter Methods	**Link**
`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions.

Examples

```
x <- Logistic$new(mean = 2, scale = 3)

# Update parameters
# When any parameter is updated, all others are too!
x$setParameterValue(sd = 2)
```

```
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`LogisticKernel` *Logistic Kernel*

Description

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

$$f(x) = (exp(x) + 2 + exp(-x))^{-1}$$

over the support $x \in R$.

Value

Returns an R6 object inheriting from class Kernel.

Constructor

LogisticKernel$new(decorators = NULL)

Constructor Arguments

Argument	Type	Details
`decorators`	Decorator	decorators to add functionality.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.

package The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	Link
prec()	prec
stdev()	stdev
mode()	mode
mean()	mean.Distribution
median()	median.Distribution
iqr()	iqr
correlation()	correlation

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	Link
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

`Loglogistic` *Log-Logistic Distribution Class*

Description

Mathematical and statistical functions for the Log-Logistic distribution, which is commonly used in survival analysis for its non-monotonic hazard as well as in economics.

Details

The Log-Logistic distribution parameterised with shape, β, scale, α, and location, γ, is defined by the pdf,

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

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

The distribution is supported on the non-negative Reals.

`entropy`, `mgf` and `cf` are omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results.

Also known as the Fisk distribution.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Loglogistic$new(scale = 1, shape = 1, location = 0, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
shape	numeric	shape parameter.
scale	numeric	scale parameter.
location	numeric	location parameter.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Log-Logistic distribution is parameterised with `shape` and `scale` as positive numerics and `location` as a numeric.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution

rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

listDistributions for all available distributions. Logistic for the Logistic distribution. CoreStatistics for numerical results.

Examples

```
x <- Loglogistic$new(shape = 2, scale = 3)

# Update parameters
x$setParameterValue(scale = 2)
x$parameters()

# d/p/q/r
x$pdf(5:6)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`Lognormal` *Log-Normal Distribution Class*

Description

Mathematical and statistical functions for the Log-Normal distribution, which is commonly used to model many natural phenomena as a result of growth driven by small percentage changes.

Details

The Log-Normal distribution parameterised with logmean, μ, and logvar, σ, is defined by the pdf,

$$exp(-(log(x) - \mu)^2/2\sigma^2)/(x\sigma\sqrt{(2\pi)})$$

for $\mu \epsilon R$ and $\sigma > 0$.

The distribution is supported on the Positive Reals.

`cf` is omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results.

Also known as the Log-Gaussian distribution.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Lognormal$new(meanlog = 0, varlog = 1, sdlog = NULL, preclog = NULL, mean = 1, var = NULL, sd = NULL, prec = NULL, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
meanlog	numeric	mean of the distribution on the log scale.
varlog	numeric	variance of the distribution on the log scale.
sdlog	numeric	standard deviation of the distribution on the log scale.
preclog	numeric	precision of the distribution on the log scale.
mean	numeric	mean of the distribution on the natural scale.
var	numeric	variance of the distribution on the natural scale.
sd	numeric	standard deviation of the distribution on the natural scale.
prec	numeric	precision of the distribution on the natural scale.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Log-Normal distribution is parameterised with either meanlog and varlog, sdlog or preclog, or mean and var, sd or prec. These are related via

$$var = (exp(var) - 1)) * exp(2 * meanlog + varlog)$$

$$sdlog = varlog^2$$

$$sd = var^2$$

Analogously for prec and preclog. If prec is given then all other parameters other than mean are ignored. If sd is given then all other parameters (except prec) are ignored. If var is given then all log parameters are ignored. If preclog is given then varlog and sdlog are ignored. Finally if sdlog is given then varlog is ignored.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm

type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	**Link**
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**

`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `Normal` for the Normal distribution. `CoreStatistics` for numerical results.

Examples

```
# Many parameterisations are possible
Lognormal$new(var = 2, mean = 1)
Lognormal$new(meanlog = 2, preclog = 5)
# Note parameters must be on same scale (log or natural)
Lognormal$new(meanlog = 4, sd = 2)

x <- Lognormal$new(verbose = TRUE) # meanlog = 0, sdlog = 1 default

# Update parameters
# When any parameter is updated, all others are too!
x$setParameterValue(meanlog = 3)
x$parameters()

# But you can only set parameters on the same scale, the below has no effect
x$setParameterValue(sd = 3)
# But this does
x$setParameterValue(sdlog = 3)

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

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()))
```

max.SetInterval *SetInterval Maximum Accessor*

Description

Returns the SetInterval maximum.

Details

This is an R6 method only, no S3 dispatch is available.

Value

Maximum as a numeric.

R6 Usage

$max()

See Also

SetInterval

`mean.Distribution` *Distribution Mean*

Description

Arithmetic mean for the probability distribution.

Usage

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

Arguments

x	Distribution.
...	Additional arguments.

Details

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.

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

Value

Mean as a numeric.

R6 Usage

$mean()

See Also

CoreStatistics, decorate and genExp.

median.Distribution *Median of a Distribution*

Description

Median of a distribution assuming quantile is provided.

Usage

```
## 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.

Details

The median is computed as the quantile function evaluated at 0.5. If the quantile is not found in the distribution (analytically or numerically), returns error.

Value

Quantile function evaluated at 0.5 as a numeric.

R6 Usage

$median()

See Also

quantile.Distribution

merge.ParameterSet *Combine ParameterSets*

Description

merge dispatch method to combine parameter sets by rows.

Usage

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

Arguments

x	ParameterSet
y	ParameterSet
...	ParameterSets

Value

An R6 object of class ParameterSet.

R6 Usage

$merge(y, ...)

See Also

ParameterSet

---

mgf *Moment Generating Function*

Description

Moment generating function of a distribution

Usage

```
mgf(object, t)
```

Arguments

object	Distribution.
t	integer to evaluate moment generating function at.

Details

The moment generating function is defined by

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

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

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

Value

Moment generating function evaluated at t as a numeric.

R6 Usage

$mgf(t)

See Also

CoreStatistics and decorate

---

min.SetInterval *SetInterval Minimum Accessor*

Description

Returns the SetInterval minimum.

Details

This is an R6 method only, no S3 dispatch is available.

Value

Minimum as a numeric.

R6 Usage

$min()

See Also

SetInterval

MixtureDistribution *Mixture Distribution Wrapper*

Description

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

Details

A Mixture Distribution is a weighted combination of two or more distributions such that for pdf/cdfs of n distribution $f_1, ..., f_n$/$F_1, ..., F_n$ and a given weight associated to each distribution, $w_1, ..., w_n$. The pdf of the mixture distribution $M(X1, ..., XN)$, f_M is given by

$$f_M = \sum_i (f_i)(w_i)$$

and the cdf, F_M is given by

$$F_M = \sum_i (F_i)(w_i)$$

If weights are given, they should be provided as a vector of numerics. If they don't sum to one then they are normalised automatically. If NULL, they are taken to be uniform, i.e. for n distributions, $w_i = 1/n, \forall\, i \in [1, n]$.

Can optionally be constructed using a `VectorDistribution`, in which case `distlist` is ignored and the mixture is constructed with the wrapped models in the vector.

Value

Returns an R6 object of class MixtureDistribution.

Constructor

MixtureDistribution$new(distlist, weights = NULL, vectordist = NULL)

Constructor Arguments

Argument	Type	Details
`distlist`	list	List of distributions.
`weights`	numeric	Vector of weights. See Details.
`vectordist`	numeric	Vector Distribution. See Details.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	**Link**
`wrappedModels(model = NULL)`	wrappedModels
`decorators()`	decorators
`traits()`	traits
`valueSupport()`	valueSupport
`variateForm()`	variateForm
`type()`	type
`properties()`	properties
`support()`	support
`symmetry()`	symmetry
`sup()`	sup
`inf()`	inf
`dmax()`	dmax
`dmin()`	dmin
`skewnessType()`	skewnessType
`kurtosisType()`	kurtosisType

d/p/q/r Methods	**Link**
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	pdf
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	cdf
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	quantile.Distribution
`rand(n, simplify = TRUE)`	rand

Statistical Methods	**Link**
`prec()`	prec
`stdev()`	stdev
`median()`	median.Distribution
`iqr()`	iqr
`cor()`	cor

Parameter Methods	**Link**
`parameters(id)`	parameters
`getParameterValue(id, error = "warn")`	getParameterValue
`setParameterValue(..., lst = NULL, error = "warn")`	setParameterValue

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	liesInSupport
`liesInType(x, all = TRUE, bound = FALSE)`	liesInType

Representation Methods	Link
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

See Also

`listWrappers`

Examples

```
mixture <- MixtureDistribution$new(list(Binomial$new(prob = 0.5, size = 10), Binomial$new()),
                                   weights = c(0.2,0.8))
mixture$pdf(1)
mixture$cdf(1)
```

`mode` *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`.

R6 Usage

$mode(which = "all")

See Also

`CoreStatistics` and `decorate`.

`Multinomial` *Multinomial Distribution Class*

Description

Mathematical and statistical functions for the Multinomial distribution, which is commonly used to extend the binomial distribution to multiple variables, for example to model the rolls of multiple dice multiple times.

Details

The Multinomial distribution parameterised with number of trials, n, and probabilities of success, $p_1, ..., p_k$, is defined by the pmf,

$$f(x_1, x_2, \ldots, x_k) = n!/(x_1! * x_2! * \ldots * x_k!) * p_1^{x_1} * p_2^{x_2} * \ldots * p_k^{x_k}$$

for $p_i, i = 1, \ldots, k; \sum p_i = 1$ and $n = 1, 2, \ldots$.

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.

Constructor

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

Constructor Arguments

Argument	Type	Details
`size`	numeric	number of trials. See details.
`probs`	numeric	vector of probabilities. See details.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Multinomial distribution is parameterised with `size` as a positive whole number and `probs` as a vector of numerics between 0 and 1. The length of the probability vector, K, tells the constructor how many arguments to expect to be passed to the maths/stats methods. The probability vector is automatically normalised with

$$probs = probs/sum(probs)$$

.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	decorators
`traits()`	traits
`valueSupport()`	valueSupport
`variateForm()`	variateForm
`type()`	type
`properties()`	properties
`support()`	support
`symmetry()`	symmetry
`sup()`	sup
`inf()`	inf
`dmax()`	dmax
`dmin()`	dmin
`skewnessType()`	skewnessType
`kurtosisType()`	kurtosisType

Statistical Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	pdf
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	cdf
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	quantile.Distribution
`rand(n, simplify = TRUE)`	rand

`mean()`	`mean.Distribution`
`variance()`	`variance`
`stdev()`	`stdev`
`prec()`	`prec`
`cor()`	`cor`
`skewness()`	`skewness`
`kurtosis(excess = TRUE)`	`kurtosis`
`entropy(base = 2)`	`entropy`
`mgf(t)`	`mgf`
`cf(t)`	`cf`
`pgf(z)`	`pgf`
`median()`	`median.Distribution`
`iqr()`	`iqr`

Parameter Methods	**Link**
`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `Binomial` for a special case of the Multinomial distribution. `FunctionImputation` to numerically impute d/p/q/r.

Examples

```
x <- Multinomial$new(size = 5, probs = c(0.1, 0.5, 0.9)) # Automatically normalised

# Update parameters
x$setParameterValue(size = 10)
# Number of categories cannot be changed after construction
x$setParameterValue(probs = c(1,2,3))
x$parameters()

# d/p/q/r
# Note the difference from R stats
x$pdf(4, 4, 2)
# This allows vectorisation:
x$pdf(c(1,4),c(2,4),c(7,2))

x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`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, μ, and covariance matrix, Σ, is defined by the pdf,

$$f(x_1, ..., x_k) = (2 * \pi)^{-k/2} det(\Sigma)^{-1/2} exp(-1/2(x - \mu)^T \Sigma^{-1}(x - \mu))$$

for $\mu \epsilon R^k$ and $\Sigma \epsilon R^{kxk}$.

The distribution is supported on the Reals and only when the covariance matrix is positive-definite.

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

The parameter `K` is automatically updated by counting the length of the mean vector and once constructed this cannot be changed. If a `mean` vector of length greater than K is given then this is

truncated to the correct length. If a mean vector of length less than K is given then this replicated and truncated to the correct length. Similarly cov and prec are internally coerced with matrix(cov,nrow = K,byrow = FALSE).

Sampling is performed via the Cholesky decomposition using chol.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

MultivariateNormal$new(mean = rep(0,2), cov = c(1,0,0,1), prec = NULL, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
mean	numeric	vector of means.
cov	numeric	vector or matrix. See details.
prec	numeric	vector or matrix. See details.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Multivariate Normal distribution is parameterised with mean as a vector of numerics and either cov or prec as positive semi-definite matrices. These are related via,

$$prec = cov^{-1}$$

If prec is given then cov is ignored.

The covariance matrix can either be supplied as a matrix or as a vector that can be coerced via matrix(cov,nrow = K,byrow = FALSE).

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link

`liesInSupport(x, all = TRUE, bound = FALSE)`	liesInSupport
`liesInType(x, all = TRUE, bound = FALSE)`	liesInType

Representation Methods	**Link**
`strprint(n = 2)`	strprint
`print(n = 2)`	print
`summary(full = T)`	summary.Distribution
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

Gentle, J.E. (2009). Computational Statistics. Statistics and Computing. New York: Springer. pp. 315–316. doi:10.1007/978-0-387-98144-4. ISBN 978-0-387-98143-7.

See Also

listDistributions for all available distributions. chol for the implementation of the Cholesky decomposition. Normal for a special case of the Multivariate Normal distribution. CoreStatistics for numerical results. FunctionImputation to numerically impute d/p/q/r.

Examples

```
# Different parameterisations
MultivariateNormal$new(mean = c(0,0,0), cov = matrix(c(3,-1,-1,-1,1,0,-1,0,1), byrow=TRUE,nrow=3))
MultivariateNormal$new(mean = c(0,0,0), cov = c(3,-1,-1,-1,1,0,-1,0,1)) # Equivalently
MultivariateNormal$new(mean = c(0,0,0), prec = c(3,-1,-1,-1,1,0,-1,0,1))

# Default is bivariate standard normal
x <- MultivariateNormal$new()

# Update parameters
x$setParameterValue(mean = c(1, 2))
# When any parameter is updated, all others are too!
x$setParameterValue(prec = c(1,0,0,1))
x$parameters()

# d/p/q/r
# Note the difference from R stats
x$pdf(1, 2)
# This allows vectorisation:
x$pdf(1:3, 2:4)
x$rand(4)
```

```
# Statistics
x$mean()
x$variance()

summary(x)
```

`Naturals` *Set of Natural Numbers*

Description

The mathematical set of natural numbers.

Details

The set of Naturals is defined as the counting numbers, i.e.

$$Naturals = \{0, 1, 2, ...\}$$

Value

Returns R6 object of class Naturals.

Constructor

Naturals$new(dim = 1, lower = 0)

Constructor Arguments

Argument	Type	Details
`dim`	numeric	Dimension of the set.
`lower`	integer	Where to start the set.

Constructor Details

Generally the `lower` argument should be ignored, its primary use-case is for the `PosNaturals` child-class.

See Also

`listSpecialSets`

Examples

```
Naturals$new()
Naturals$new(dim = 2)
```

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 successes.

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 $p\epsilon[0, 1]$, 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, `prob` is always the probability of success and `qprob` is the probability of failure. Use `$description` to see the Negative Binomial form.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

NegativeBinomial$new(size = 10, prob = 0.5, qprob = NULL, mean = NULL, form = "fbs", decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
size	numeric	number of failures/successes.
prob	numeric	probability of success.
qprob	numeric	probability of failure.
mean	numeric	location parameter.
form	character	form of negative binomial, see details.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Negative Binomial distribution is parameterised with size as a positive whole number, and either prob or qprob as a number between 0 and 1, or mean as a numeric greater than the number of failures/successes (if form is 'tbf' or 'tbs'). These are related via,

$$qprob = 1 - prob$$

and the mean formula is dependent on the form. If mean is given then qprob and prob are ignored. If qprob is given then prob is ignored.

The additional form argument determines which of the four Negative Binomial distributions should be constructed, this cannot be updated after construction. form should be one of "sbf" (successes before failures), "tbf" (trials before failures), "fbs" (failures before successes) or "tbs" (trials before successes). "fbs" is taken as default if none are supplied or an unrecognised form is given.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry

sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.

`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `Binomial` for the Binomial distribution and `Geometric` for the Geometric distribution.

Examples

```
# Different parameterisations
NegativeBinomial$new(size = 5, prob = 0.2)
NegativeBinomial$new(size = 5, qprob = 0.2)
NegativeBinomial$new(size = 5, mean = 4)

# Different forms of the distribution
NegativeBinomial$new(form = "fbs")
NegativeBinomial$new(form = "sbf")

# Use description to see which form is used
NegativeBinomial$new(form = "tbf")
NegativeBinomial$new(form = "tbs")

x <- NegativeBinomial$new() # Default is size = 10, prob = 0.5 and failures before successes

# Update parameters (form cannot be updated)
x$setParameterValue(qprob = 0.2)  # When any parameter is updated, all others are too!
x$parameters()


# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

NegIntegers *Set of Negative Integers*

Description

The mathematical set of negative integers.

Details

The set of NegIntegers is defined as the set of negative or non-positive numbers that can be written without a fractional component, i.e.

$$NegIntegers = \{..., -3, -2, -1, 0\}$$

0 may or may not be included (depending on the `zero` argument).

Value

Returns R6 object of class NegIntegers.

Constructor

NegIntegers$new(dim = 1, zero = FALSE)

Constructor Arguments

Argument	Type	Details
`dim`	numeric	Dimension of the set.
`zero = FALSE`	logical	If TRUE, zero is included in the set.

See Also

`listSpecialSets`

Examples

```
NegIntegers$new()
NegIntegers$new(zero = TRUE)
NegIntegers$new(dim = 2)
```

`NegRationals` *Set of Negative Rationals*

Description

The mathematical set of negative rational numbers.

Details

The set of Positive Rationals is defined as the set of numbers that can be written as a fraction of two integers and are non-negative, i.e.

$$NegRationals = \{\frac{p}{q} | p, q \in Z, \frac{p}{q} \leq 0\}$$

where Z is the set of integers.

0 may or may not be included (depending on the `zero` argument).

Value

Returns R6 object of class NegRationals.

Constructor

NegRationals$new(dim = 1, zero = FALSE)

Constructor Arguments

Argument	Type	Details
`dim`	numeric	Dimension of the set.
`zero = FALSE`	logical	If TRUE, zero is included in the set.

See Also

`listSpecialSets`

Examples

```
NegRationals$new()
NegRationals$new(zero = TRUE)
NegRationals$new(dim = 2)
```

NegReals *Set of Negative Reals*

Description

The mathematical set of negative real numbers.

Details

The set of Negative Reals is defined as the union of the set of negative rationals and negative irrationals, i.e.

$$NegReals = I^- \cup Q^-$$

where I^- is the set of negative irrationals and Q^- is the set of negative rationals.

0 may or may not be included (depending on the `zero` argument).

Value

Returns R6 object of class NegReals.

Constructor

NegReals$new(dim = 1, zero = FALSE)

Constructor Arguments

Argument	Type	Details
`dim`	numeric	Dimension of the set.
`zero = FALSE`	logical	If TRUE, zero is included in the set.

See Also

`listSpecialSets`

Examples

```
NegReals$new()
NegReals$new(zero = TRUE)
NegReals$new(dim = 2)
```

`Normal` *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, σ^2, and mean, μ, is defined by the pdf,

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

for $\mu \epsilon R$ 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.

Constructor

Normal$new(mean = 0, var = 1, sd = NULL, prec = NULL, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`mean`	numeric	mean, location parameter.
`var`	numeric	variance, squared scale parameter.
`sd`	numeric	standard deviation, scale parameter.
`prec`	numeric	precision, inverse squared scale parameter.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Normal distribution is parameterised with `mean` as a numeric, and either `var`, `sd` or `prec` as numerics. These are related via,

$$sd = \sqrt{(var)}$$

$$prec = 1/var$$

If `prec` is given then `sd` and `var` are ignored. If `sd` is given then `var` is ignored.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	Link
`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	Link
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	Link
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions.

Examples

```
# Different parameterisations
Normal$new(var = 1, mean = 1)
Normal$new(prec = 2, mean = 1)
Normal$new(mean = 1, sd = 2)
x <- Normal$new(verbose = TRUE) # Standard normal default

# Update parameters
x$setParameterValue(var = 2)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
```

```
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`NormalKernel` *Normal Kernel*

Description

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

$$f(x) = exp(-x^2/2)/\sqrt{2\pi}$$

over the support $x \in \mathsf{R}$.

Details

We use the `erf` and `erfinv` error and inverse error functions from the Pracma package.

Value

Returns an R6 object inheriting from class Kernel.

Constructor

NormalKernel$new(decorators = NULL)

Constructor Arguments

Argument	Type	Details
`decorators`	Decorator	decorators to add functionality.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	Link
prec()	prec
stdev()	stdev
mode()	mode
mean()	mean.Distribution
median()	median.Distribution
iqr()	iqr
correlation()	correlation

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	Link
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

`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.

Details

If `id` is given and matches a parameter in the distribution, the parameter is returned with all details. If `id` is given but doesn't match a parameter, an empty data.table is returned. Finally if `id` is not given, returns self.

Value

An R6 object of class ParameterSet or a data.table.

R6 Usage

$parameters(id = NULL)

See Also

`getParameterValue` and `setParameterValue`

`ParameterSet` *Make an R6 Parameter Set for Distributions*

Description

ParameterSets are passed to a `Distribution$new` constructor when creating a custom probability distribution that takes parameters.

Value

An R6 object of class ParameterSet.

Constructor

ParameterSet$new(id, value, support, settable, updateFunc = NULL, description = NULL)

Constructor Arguments

Argument	Type	Details
`id`	character	unique one-word identifier.
`value`	numeric	initial parameter value.
`support`	numeric	range of values parameter can take.
`settable`	logical	if TRUE the parameter is printed. See Details.
`updateFunc = NULL`	function	evaluated to update parameter. See Details.
`description = NULL`	character	optional description of parameter.

Constructor Details

An R6 ParameterSet is required to construct a custom Probability Distribution that takes parameters. This constructor ensures that the correct format of parameters is supplied to the distribution.

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

Each parameter requires a unique one-word `id` that is used to get and set parameters after construction. The parameterisation of the distribution is determined by the parameters that have `settable = TRUE`, this is a slightly confusing term as it actually refers to a parameter being 'machine-settable'. Here it just means that the given parameter is used in construction and therefore will be included in a call to `$print`. `updateFunc` is used to update the parameters not used in the parameterisation. These should be given as a function that could be understood in the body of a Distribution and should start with `function(self)`, see examples.

Internally after calling `$setParameterValue`, `$update` is called to update all parameters with a non-NA `updateFunc`.

Public Methods

Method	Link
print(hide_cols = c("updateFunc","settable"))	print.ParameterSet
update()	update.ParameterSet
parameters(id = NULL)	parameters
getParameterSupport(id, error = "warn")	getParameterSupport
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue
merge(y, ...)	merge.ParameterSet
as.data.table()	as.data.table

See Also

Distribution

Examples

```
id = list("prob", "size")
value = list(0.2, 5)
support = list(Interval$new(0,1), PosNaturals$new())
settable = list(TRUE, TRUE)
description = list("Probability of success",NULL)
ps = ParameterSet$new(id, value, support, settable,
                      description = description)
ps$parameters()
ps$getParameterValue("prob")
ps$getParameterSupport("prob")


id = list("rate", "scale")
value = list(1, 1)
support = list(PosReals$new(), PosReals$new())
settable = list(TRUE, FALSE)
updateFunc = list(NULL, function(self) 1/self$getParameterValue('rate'))
description = list("Arrival rate","Scale parameter")
ps = ParameterSet$new(id, value, support, settable,
                      updateFunc, description)
ps$parameters(id = "rate")
ps$setParameterValue(rate = 2) # Automatically calls $update
ps$getParameterValue("scale") # Auto-updated to 1/2
```

`Pareto` *Pareto Distribution Class*

Description

Mathematical and statistical functions for the Pareto distribution, which is commonly used in Economics to model the distribution of wealth and the 80-20 rule.

Details

The Pareto distribution parameterised with shape, α, and scale, β, is defined by the pdf,

$$f(x) = (\alpha\beta^{\alpha})/(x^{\alpha+1})$$

for $\alpha, \beta > 0$.

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

`cf` is omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Pareto$new(shape = 1, scale = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`shape`	numeric	shape parameter.
`scale`	numeric	scale parameter.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Pareto distribution is parameterised with `shape` and `scale` as positive numerics.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.

description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	Link
parameters(id)	parameters

`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `CoreStatistics` for numerical results.

Examples

```
x = Pareto$new(shape = 2, scale = 1)

# Update parameters
x$setParameterValue(scale = 5.1)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

pdf *Probability Density/Mass Function*

Description

Returns the probability density/mass function for continuous/discrete (or mixture) distributions evaluated at a given point.

Usage

```
pdf(object, x1, ..., log = FALSE, simplify = TRUE)
```

Arguments

object	Distribution.
x1	vector of numerics to evaluate function at.
...	additional arguments.
log	logical; if TRUE, probabilities p are given as log(p).
simplify	if TRUE (default) returns results in simplest form (vector or data.table) otherwise as data.table.

Details

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 without warning using an analytic expression. Otherwise, if the distribution has not been decorated with `FunctionImputation`, `NULL` is returned. To impute the pdf, use `decorate(distribution,FunctionImputation)`, this will provide a numeric calculation for the pdf with warning.

Additional named arguments can be passed, which are required for composite distributions such as `ProductDistribution` and `ArrayDistribution`.

Value

Probability density function evaluated at given points as either a numeric if `simplify` is TRUE or as a data.table.

R6 Usage

$pdf(x1, ..., log = FALSE, simplify = TRUE)

See Also

cdf, quantile, rand for other statistical functions. FunctionImputation, decorate for imputing missing functions.

---

pdfPNorm *Probability Density Function P-Norm*

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.

Details

The p-norm of the pdf is defined by

$$(\int_a^b |f_X|^p d\mu)^{1/p}$$

where X is the distribution, f_X is the pdf and a,b are the limits of integration.

Returns NULL if distribution is not continuous.

Can only be used after decorating with ExoticStatistics.

Value

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

R6 Usage

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

See Also

ExoticStatistics and decorate

pgf *Probability Generating Function*

Description

Probability generating function of a distribution

Usage

```
pgf(object, z)
```

Arguments

object	Distribution.
z	integer to evaluate characteristic function at.

Details

The probability generating function is defined by

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

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

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

Value

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

R6 Usage

$pgf(z)

See Also

CoreStatistics and decorate

`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, λ, is defined by the pmf,

$$f(x) = (\lambda^x * exp(-\lambda))/x!$$

for $\lambda > 0$.

The distribution is supported on the Naturals.

`entropy` is omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Poisson$new(rate = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`rate`	numeric	arrival rate.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Poisson distribution is parameterised with `rate` as a positive numeric.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.

`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	**Link**
`decorators()`	decorators
`traits()`	traits
`valueSupport()`	valueSupport
`variateForm()`	variateForm
`type()`	type
`properties()`	properties
`support()`	support
`symmetry()`	symmetry
`sup()`	sup
`inf()`	inf
`dmax()`	dmax
`dmin()`	dmin
`skewnessType()`	skewnessType
`kurtosisType()`	kurtosisType

Statistical Methods	**Link**
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	pdf
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	cdf
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	quantile.Distribution
`rand(n, simplify = TRUE)`	rand
`mean()`	mean.Distribution
`variance()`	variance
`stdev()`	stdev
`prec()`	prec
`cor()`	cor
`skewness()`	skewness
`kurtosis(excess = TRUE)`	kurtosis
`entropy(base = 2)`	entropy
`mgf(t)`	mgf
`cf(t)`	cf
`pgf(z)`	pgf
`median()`	median.Distribution
`iqr()`	iqr

Parameter Methods	**Link**
`parameters(id)`	parameters

`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `CoreStatistics` for numerical results.

Examples

```
x = Poisson$new(rate = 2)

# Update parameters
x$setParameterValue(rate = 3)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

PosIntegers *Set of Positive Integers*

Description

The mathematical set of positive integers.

Details

The set of PosIntegers is defined as the set of positive or non-negative numbers that can be written without a fractional component, i.e.

$$PosIntegers = \{0, 1, 2, 3, ...\}$$

0 may or may not be included (depending on the `zero` argument).

Value

Returns R6 object of class PosIntegers.

Constructor

PosIntegers$new(dim = 1, zero = FALSE)

Constructor Arguments

Argument	Type	Details
`dim`	numeric	Dimension of the set.
`zero = FALSE`	logical	If TRUE, zero is included in the set.

See Also

`listSpecialSets`

Examples

```
PosIntegers$new()
PosIntegers$new(zero = TRUE)
PosIntegers$new(dim = 2)
```

PosNaturals *Set of Positive Natural Numbers*

Description

The mathematical set of positive natural numbers.

Details

The set of Positive Naturals is defined as the positive counting numbers, i.e.

$$PosNaturals = \{1, 2, 3, ...\}$$

Value

Returns R6 object of class PosNaturals.

Constructor

PosNaturals$new(dim = 1)

Constructor Arguments

Argument	Type	Details
dim	numeric	Dimension of the set.

See Also

listSpecialSets

Examples

```
PosNaturals$new()
PosNaturals$new(dim = 2)
```

PosRationals *Set of Positive Rationals*

Description

The mathematical set of positive rational numbers.

Details

The set of Positive Rationals is defined as the set of numbers that can be written as a fraction of two integers and are non-negative, i.e.

$$PosRationals = \{\frac{p}{q} | p, q \in Z, \frac{p}{q} \geq 0\}$$

where Z is the set of integers.

0 may or may not be included (depending on the `zero` argument).

Value

Returns R6 object of class PosRationals.

Constructor

PosRationals$new(dim = 1, zero = FALSE)

Constructor Arguments

Argument	Type	Details
`dim`	numeric	Dimension of the set.
`zero = FALSE`	logical	If TRUE, zero is included in the set.

See Also

`listSpecialSets`

Examples

```
PosRationals$new()
PosRationals$new(zero = TRUE)
PosRationals$new(dim = 2)
```

PosReals *Set of Positive Reals*

Description

The mathematical set of positive real numbers.

Details

The set of Positive Reals is defined as the union of the set of positive rationals and positive irrationals, i.e.

$$PosReals = I^+ \cup Q^+$$

where I^+ is the set of positive irrationals and Q^+ is the set of positive rationals.

0 may or may not be included (depending on the `zero` argument).

Value

Returns R6 object of class PosReals.

Constructor

PosReals$new(dim = 1, zero = FALSE)

Constructor Arguments

Argument	Type	Details
`dim`	numeric	Dimension of the set.
`zero = FALSE`	logical	If TRUE, zero is included in the set.

See Also

`listSpecialSets`

Examples

```
PosReals$new()
PosReals$new(zero = TRUE)
PosReals$new(dim = 2)
```

`power.SetInterval` *Symbolic Exponentiation for SetInterval*

Description

Makes a symbolic representation for the exponentiation of a given set/interval.

Usage

```
power.SetInterval(x, power)

## S3 method for class 'SetInterval'
x ^ power
```

Arguments

x	SetInterval
power	power to raise SetInterval to

Details

This does not calculate the exponentiation but is just a symbolic representation using unicode.

Value

Returns an R6 object inheriting from SetInterval.

See Also

product.SetInterval, union.SetInterval, complement.SetInterval

Examples

```
PosNaturals$new() ^ 2
power.SetInterval(Reals$new(), 3)
```

---

prec *Precision of a Distribution*

Description

Precision of a distribution assuming variance is provided.

Usage

```
prec(object)
```

Arguments

object	Distribution.

Details

The precision is analytically computed as the reciprocal of the variance. If the variance is not found in the distribution (analytically or numerically), returns error.

Value

Reciprocal of variance as a numeric.

R6 Usage

$prec()

See Also

variance

`print.ParameterSet` *Print a ParameterSet*

Description

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

Usage

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

Arguments

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

Details

If given the `hide_cols` argument can be used to hide specific columns from the data.table.

R6 Usage

$print(hide_cols = c("updateFunc","settable"))

See Also

ParameterSet

`product.SetInterval` *Symbolic Cartesian Product for SetInterval*

Description

Makes a symbolic representation for the cartesian product of sets/intervals.

Usage

```
product.SetInterval(...)

## S3 method for class 'SetInterval'
x * y
```

Arguments

`x`	SetInterval
`y`	SetInterval
`...`	SetIntervals to take the cartesian product of.

Details

This does not calculate the cartesian product of the arguments but is just a symbolic representation using unicode.

Value

Returns an R6 object inheriting from SetInterval.

See Also

`union.SetInterval`, `complement.SetInterval`, `power.SetInterval`

Examples

```
PosNaturals$new() * Reals$new()
product.SetInterval(PosNaturals$new(), Reals$new())
```

`ProductDistribution` *Product Distribution*

Description

A wrapper for creating the joint distribution of multiple independent probability distributions.

Details

Exploits the following relationships of independent distributions

$$f_P(X1 = x1, ..., XN = xN) = f_{X1}(x1) * ... * f_{XN}(xn)$$

$$F_P(X1 = x1, ..., XN = xN) = F_{X1}(x1) * ... * F_{XN}(xn)$$

where f_P/F_P is the pdf/cdf of the joint (product) distribution P and $X1, ..., XN$ are independent distributions.

`ProductDistribution` inherits all methods from `Distribution` and `DistributionWrapper`.

Value

Returns an R6 object of class ProductDistribution.

Constructor

ProductDistribution$new(distlist = NULL, distribution = NULL, params = NULL, name = NULL, short_name = NULL, description = NULL)

Constructor Arguments

Argument	Type	Details
`distlist`	list	List of distributions.
`distribution`	distribution	Distribution to wrap.
`params`	a R object	Either list of parameters or matrix-type frame, see examples.
`name`	list	Optional new name for distribution.
`short_name`	list	Optional new short_name for distribution.
`description`	list	Optional new description for distribution.

Constructor Details

A product distribution can either be constructed by a list of distributions passed to `distlist` or by passing the name of a distribution implemented in distr6 to `distribution`, as well as a list or table of parameters to `params`. The former case provides more flexibility in the ability to use multiple distributions but the latter is useful for quickly combining many distributions of the same type. See

examples.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`wrappedModels(model = NULL)`	`wrappedModels`
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`
`support()`	`support`
`symmetry()`	`symmetry`
`sup()`	`sup`
`inf()`	`inf`
`dmax()`	`dmax`
`dmin()`	`dmin`
`skewnessType()`	`skewnessType`
`kurtosisType()`	`kurtosisType`

d/p/q/r Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	`pdf`
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`cdf`
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	`quantile.Distribution`
`rand(n, simplify = TRUE)`	`rand`

Statistical Methods	Link
`prec()`	`prec`
`stdev()`	`stdev`
`median()`	`median.Distribution`
`iqr()`	`iqr`
`cor()`	`cor`

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

See Also

listWrappers and VectorDistribution

Examples

```
prodBin <- ProductDistribution$new(list(Binomial$new(prob = 0.5,
                           size = 10), Normal$new(mean = 15)))
prodBin$pdf(x1 = 2, x2 =3)
prodBin$cdf(1:5, 12:16)
prodBin$quantile(c(0.1,0.2),c(0.3,0.4))
prodBin$rand(10)

prodBin = ProductDistribution$new(distribution = Binomial,
       params = list(list(prob = 0.1, size = 2),
                     list(prob = 0.6, size = 4),
                     list(prob = 0.2, size = 6)))
prodBin$pdf(x1=1,x2=2,x3=3)
prodBin$cdf(x1=1,x2=2,x3=3)
prodBin$rand(10)

#Equivalently
prodBin = ProductDistribution$new(distribution = Binomial,
       params = data.table::data.table(prob = c(0.1,0.6,0.2), size = c(2,4,6)))
prodBin$pdf(x1=1,x2=2,x3=3)
```

```
prodBin$cdf(x1=1,x2=2,x3=3)
prodBin$rand(10)
```

`properties` *Properties Accessor*

Description

Returns the properties of the distribution.

Usage

```
properties(object)
```

Arguments

`object`	Distribution.

Value

List of distribution properties.

R6 Usage

$properties()

`quantile.Distribution` *Inverse Cumulative Distribution Function*

Description

Returns the inverse cumulative distribution, aka quantile, function for a distribution evaluated at a given point between 0 and 1.

Usage

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

Arguments

x	Distribution.
p	vector of probabilities to evaluate function at.
...	additional arguments.
lower.tail	logical; if TRUE, probabilities p are given as log(p).
log.p	logical; if TRUE then $q_X(exp(p))$ is returned.
simplify	if TRUE (default) returns results in simplest form (vector or data.table) otherwise as data.table.

Details

The quantile function, q_X, is the inverse cdf, i.e.

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

If lower.tail is FALSE then $q_X(1-p)$ is returned.

If available a quantile will be returned without warning using an analytic expression. Otherwise, if the distribution has not been decorated with FunctionImputation, NULL is returned. To impute the quantile, use decorate(distribution,FunctionImputation), this will provide a numeric calculation for the quantile with warning.

Additional named arguments can be passed, which are required for composite distributions such as ProductDistribution and VectorDistribution.

Value

Inverse cumulative distribution function evaluated at given points as either a numeric if simplify is TRUE or as a data.table.

R6 Usage

$quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)

See Also

pdf, cdf, rand for other statistical functions. FunctionImputation, decorate for imputing missing functions.

`Quartic` *Quartic Kernel*

Description

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

$$f(x) = 15/16(1 - x^2)^2$$

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

Details

Quantile is omitted as no closed form analytic expression could be found, decorate with FunctionImputation for numeric results.

Value

Returns an R6 object inheriting from class Kernel.

Constructor

Quartic$new(decorators = NULL)

Constructor Arguments

Argument	Type	Details
`decorators`	Decorator	decorators to add functionality.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`

traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	**Link**
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	**Link**
prec()	prec
stdev()	stdev
mode()	mode
mean()	mean.Distribution
median()	median.Distribution
iqr()	iqr
correlation()	correlation

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

---

rand *Random Simulation Function*

Description

Returns a given number of points sampled from the distribution.

Usage

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

Arguments

object	Distribution.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
simplify	if TRUE (default) returns results in simplest form (vector or data.table) otherwise as data.table.

Details

If available a rand will be returned without warning using an analytic expression. Otherwise, if the distribution has not been decorated with FunctionImputation, NULL is returned. To impute the rand, use decorate(distribution,FunctionImputation), this will provide a numeric calculation for the rand with warning.

Additional named arguments can be passed, which are required for composite distributions such as ProductDistribution and ArrayDistribution.

Value

Simulated draws from the distribution as either a numeric if simplify is TRUE or as a data.table.

R6 Usage

$rand(n, simplify = TRUE)

See Also

pdf, cdf, quantile for other statistical functions. FunctionImputation, decorate for imputing missing functions.

`Rationals` *Set of Rationals*

Description

The mathematical set of rational numbers.

Details

The set of Rationals is defined as the set of numbers that can be written as a fraction of two integers, i.e.

$$Rationals = \{\frac{p}{q} | p, q \in Z\}$$

where Z is the set of integers.

Value

Returns R6 object of class Rationals.

Constructor

Rationals$new(dim = 1,...)

Constructor Arguments

Argument	Type	Details
`dim`	numeric	Dimension of the set.
`...`	ANY	Additional arguments.

Constructor Details

Generally the `...` argument should be ignored, its primary use-case is for the child-classes.

See Also

listSpecialSets

Examples

```
Rationals$new()
Rationals$new(dim = 2)
```

`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), α, is defined by the pdf,

$$f(x) = x/\alpha^2 exp(-x^2/(2\alpha^2))$$

for $\alpha > 0$.

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

`cf` and `mgf` are omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Rayleigh$new(mode = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`mode`	numeric	mode, scale parameter.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Rayleigh distribution is parameterised with `mode` as a non-negative numeric.

Public Variables

Variable	Return
`name`	Name of distribution.

short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	Link

`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `CoreStatistics` for numerical results.

Examples

```
x <- Rayleigh$new(mode = 2)

# Update parameters
x$setParameterValue(mode = 4)
x$parameters()

# d/p/q/r
x$pdf(1:4)
x$cdf(2)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`Reals` *Set of Reals*

Description

The mathematical set of real numbers.

Details

The set of Reals is defined as the union of the set of rationals and irrationals, i.e.

$$Reals = I \cup Q$$

where I is the set of irrationals and Q is the set of rationals.

Value

Returns R6 object of class Reals.

Constructor

Reals$new(dim = 1,...)

Constructor Arguments

Argument	Type	Details
`dim`	numeric	Dimension of the set.
`...`	ANY	Additional arguments.

Constructor Details

Generally the `...` argument should be ignored, its primary use-case is for the child-classes.

See Also

`listSpecialSets`

Examples

```
Reals$new()
Reals$new(dim = 2)
```

SDistribution *Abstract Special Distribution Class*

Description

Abstract class that cannot be constructed directly.

Value

Returns error. Abstract classes cannot be constructed directly.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

Set *R6 Generalised Class for Symbolic Sets*

Description

A symbolic R6 Set class.

Details

Sets are distinguished from intervals in R6 as they are finite mathematical sets with elements that can be printed. The elements can be of any class.

Value

Returns an R6 object of class Set.

Constructor

Set$new(..., dim = 1)

Constructor Arguments

Argument	Type	Details
`...`	ANY	Elements in the set.
`dim`	integer	Dimension of the set.

Public Methods

Accessor Methods	Link
`type()`	`type.SetInterval`
`dimension()`	`dimension.SetInterval`
`max()`	`max.SetInterval`
`min()`	`min.SetInterval`
`sup()`	`sup.SetInterval`
`inf()`	`inf.SetInterval`
`getSymbol()`	`getSymbol.SetInterval`
`class()`	`class.SetInterval`
`elements()`	`elements`

Set Methods	Link
`length()`	`length.Set`

Validation Methods	Link
`liesInSetInterval(x, all = FALSE, bound = FALSE)`	`liesInSetInterval`

Representation Methods	Link
`print()`	`print`

See Also

Interval

SetInterval *R6 Generalised Class for Symbolic Sets and Intervals*

Description

A generic SetInterval class primarily used as the parent class to Set and Interval.

Details

Whilst this is not an abstract class, direct construction is generally not advised. Construction should instead be called on 'Set' or 'Interval'.

Value

Returns an R6 object of class SetInterval.

Constructor

SetInterval$new(symbol, lower, upper, type, class = "numeric", dimension)

Constructor Arguments

Argument	Type	Details
symbol	character	String representation of SetInterval.
lower	numeric	Lower limit of SetInterval.
upper	numeric	Upper limit of SetInterval.
type	character	Interval type, one of (), (], [), [].
class	character	Atomic class, one of "numeric" or "integer".
dimension	integer	Dimension of SetInterval.

Public Methods

Accessor Methods	Link
type()	type.SetInterval
dimension()	dimension.SetInterval
max()	max.SetInterval
min()	min.SetInterval
sup()	sup.SetInterval
inf()	inf.SetInterval

`getSymbol()`	`getSymbol.SetInterval`
`class()`	`class.SetInterval`

Validation Methods	**Link**
`liesInSetInterval(x, all = FALSE, bound = FALSE)`	`liesInSetInterval`

Representation Methods	**Link**
`print()`	`print`

See Also

`Set` for R6 Set objects and `Interval` for R6 Interval objects.

`setOperation` *Symbolic Operations for SetInterval*

Description

Operations for SetInterval objects and subclasses, symbolic only.

Usage

```
setOperation(unicode, sets, lower = NULL, upper = NULL, type = NULL,
  dim = NULL)
```

Arguments

`unicode`	unicode symbol for the setOperation.
`sets`	list of sets and/or intervals to combine via the setOperation.
`lower`	lower bound of new SetInterval
`upper`	upper bound of new SetInterval
`type`	type of new SetInterval
`dim`	dimension of new SetInterval

Details

Generally not recommended to use this function directly but instead via one of the implemented operations.

Value

An R6 object of class SetInterval.

Returns an R6 object inheriting from SetInterval.

See Also

`product.SetInterval`, `union.SetInterval`, `complement.SetInterval`, `power.SetInterval`

`setParameterValue` *Parameter Value Setter*

Description

Sets the value of the given parameter.

Usage

```
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`.

Details

Parameters can be updated in one of two ways, either by passing the parameters to update as named arguments or as a list with the the list names are parameter IDs and the list values are the respective values to set the parameters. Using a list may be preferred for parameters that take multiple values. See examples. If `lst` is given then any additional arguments are ignored.

`stopwarn` either breaks the code with an error if "error" is given or returns `NULL` with warning otherwise.

Value

An R6 object of class ParameterSet.

R6 Usage

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

See Also

`parameters` and `setParameterValue`

Examples

```
ps <- Normal$new()$parameters()
ps$setParameterValue(mean  = 2, var = 5)$print()

ps <- MultivariateNormal$new()$parameters()
ps$setParameterValue(lst = list(mean = c(1,1)))$print()
```

`setSymbol` *Unicode Symbol of Special Sets*

Description

Gets the unicode symbol for standard mathematical special sets.

Usage

```
setSymbol(set)
```

Arguments

`set`	special set

Details

Special set can be supplied as a character string or class, case-insensitive. See `listSpecialSets` for full list of currently supported sets.

Value

Returns the unicode representation of mathematical special sets.

See Also

`SpecialSet`, `listSpecialSets`

Examples

```
# Supplied as class
setSymbol(empty)

# Supplied as string
setSymbol("empty")

# Case-insensitive
setSymbol(EmPtY)
```

`Sigmoid` *Sigmoid Kernel*

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.

Value

Returns an R6 object inheriting from class Kernel.

Constructor

Sigmoid$new(decorators = NULL)

Constructor Arguments

Argument	Type	Details
`decorators`	Decorator	decorators to add functionality.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`

traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	**Link**
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	**Link**
prec()	prec
stdev()	stdev
mode()	mode
mean()	mean.Distribution
median()	median.Distribution
iqr()	iqr
correlation()	correlation

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

`Silverman` *Silverman Kernel*

Description

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

$$f(x) = exp(-|x|/\sqrt{2})/2 * sin(|x|/\sqrt{2} + \pi/4)$$

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.

Value

Returns an R6 object inheriting from class Kernel.

Constructor

Silverman$new(decorators = NULL)

Constructor Arguments

Argument	Type	Details
`decorators`	Decorator	decorators to add functionality.

Public Variables

Variable	Return
`name`	Name of distribution.

`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	**Link**
`decorators()`	decorators
`traits()`	traits
`valueSupport()`	valueSupport
`variateForm()`	variateForm
`type()`	type
`properties()`	properties
`support()`	support
`symmetry()`	symmetry
`sup()`	sup
`inf()`	inf
`dmax()`	dmax
`dmin()`	dmin
`skewnessType()`	skewnessType
`kurtosisType()`	kurtosisType

d/p/q/r Methods	**Link**
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	pdf
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	cdf
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	quantile.Distribution
`rand(n, simplify = TRUE)`	rand

Statistical Methods	**Link**
`prec()`	prec
`stdev()`	stdev
`mode()`	mode
`mean()`	mean.Distribution
`median()`	median.Distribution
`iqr()`	iqr
`correlation()`	correlation

Parameter Methods	**Link**
`parameters(id)`	parameters
`getParameterValue(id, error = "warn")`	getParameterValue

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

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

`simulateEmpiricalDistribution`

Sample Empirical Distribution Without Replacement

Description

Function to sample Distributions of class `Empirical` 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

set.seed, rand, and Empirical

---

skewness *Distribution Skewness*

Description

Skewness of a distribution

Usage

```
skewness(object)
```

Arguments

object	Distribution.

Details

The skewness of a distribution is defined by the third standardised moment of the distribution,

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

where E_X is the expectation of distribution X, μ is the mean of the distribution and σ is the standard deviation of the distribution.

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

Value

Skewness as a numeric.

R6 Usage

$skewness()

See Also

CoreStatistics and decorate

skewnessType *Type of Skewness Accessor*

Description

Returns the type of 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.

R6 Usage

$skewnessType()

See Also

skewness, properties and kurtosisType

skewType *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)
```

SpecialSet *Special Mathematical Sets*

Description

Abstract class for the representation of the 'special' mathematical sets.

Details

Special sets refer to the most commonly used (and important) sets in mathematics. Including the sets of natural numbers, integers and reals.

This is an abstract class that cannot be constructed, instead construct one of the implemented SpecialSet child-classes. For a full list of these see listSpecialSets.

Value

Returns error. Abstract classes cannot be constructed directly.

See Also

listSpecialSets

`squared2Norm` *Squared Probability Density Function 2-Norm*

Description

The squared 2-norm of the pdf evaluated over the whole support by default or given limits.

Usage

```
squared2Norm(object, lower = NULL, upper = NULL)
```

Arguments

`object`	Distribution.
`lower`	lower limit for integration, default is infimum.
`upper`	upper limit for integration, default is supremum.

Details

The squared 2-norm of the pdf is defined by

$$\int (f_X(u))^2 du$$

where X is the Distribution and f_X is its pdf.

If an analytic expression isn't available, returns error. To impute a numerical expression, use the `ExoticStatistics` decorator.

Value

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

R6 Usage

$squared2Norm(lower = NULL, upper = NULL)

See Also

`ExoticStatistics` and `decorate`

stdev *Standard Deviation of a Distribution*

Description

Standard deviation of a distribution assuming variance is provided.

Usage

```
stdev(object)
```

Arguments

object	Distribution.

Details

The standard deviation is analytically computed as the square root of the variance. If the variance is not found in the distribution (analytically or numerically), returns error.

Value

Square-root of variance as a numeric.

R6 Usage

$stdev()

See Also

variance

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, ν, is defined by the pdf,

$$f(x) = \Gamma((\nu+1)/2)/(\sqrt(\nu\pi)\Gamma(\nu/2)) * (1+(x^2)/\nu)^(- (\nu+1)/2)$$

for $\nu > 0$.

The distribution is supported on the Reals.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

StudentT$new(df = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`df`	numeric	degrees of freedom.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Student's T distribution is parameterised with `df` as a positive numeric.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	decorators
`traits()`	traits
`valueSupport()`	valueSupport
`variateForm()`	variateForm
`type()`	type
`properties()`	properties
`support()`	support
`symmetry()`	symmetry
`sup()`	sup
`inf()`	inf
`dmax()`	dmax
`dmin()`	dmin
`skewnessType()`	skewnessType
`kurtosisType()`	kurtosisType

Statistical Methods	Link
`pdf(x1, ..., log = FALSE, simplify = TRUE)`	pdf
`cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	cdf
`quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`	quantile.Distribution
`rand(n, simplify = TRUE)`	rand
`mean()`	mean.Distribution
`variance()`	variance
`stdev()`	stdev
`prec()`	prec
`cor()`	cor
`skewness()`	skewness
`kurtosis(excess = TRUE)`	kurtosis

`entropy(base = 2)`	`entropy`
`mgf(t)`	`mgf`
`cf(t)`	`cf`
`pgf(z)`	`pgf`
`median()`	`median.Distribution`
`iqr()`	`iqr`

Parameter Methods	**Link**
`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `Normal` for the Normal distribution.

Examples

```
x = StudentT$new(df = 2)

# Update parameters
x$setParameterValue(df = 3)
x$parameters()
```

```
# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`summary.Distribution` *Distribution Summary*

Description

Summary method for distribution objects (and all child classes).

Usage

```
## S3 method for class 'Distribution'
summary(object, full = TRUE, ...)
```

Arguments

`object`	Distribution.
`full`	logical; if TRUE (default), gives an extended summary, otherwise brief.
`...`	additional arguments.

Value

Printed summary of the distribution.

R6 Usage

$summary(full = TRUE)

See Also

`Distribution`

sup *Supremum Accessor*

Description

Returns the distribution supremum as the supremum of the support.

Usage

```
sup(object)
```

Arguments

object	Distribution.

Value

Supremum as a numeric.

R6 Usage

$sup()

See Also

support, dmax, dmin, inf

sup.SetInterval *SetInterval Supremum Accessor*

Description

Returns the SetInterval supremum.

Details

This is an R6 method only, no S3 dispatch is available.

Value

Supremum as a numeric.

R6 Usage

$sup()

See Also

SetInterval

support *Support Accessor*

Description

Returns the support of the distribution.

Usage

```
support(object)
```

Arguments

object	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.

Value

An R6 object of class SetInterval.

R6 Usage

$support()

See Also

SetInterval and properties

survival *Survival Function*

Description

The survival function of a probability distribution is the probability of surviving after a point x.

Usage

```
survival(object, x1, log = FALSE)
```

Arguments

object	Distribution.
x1	Point to evaluate the survival function at.
log	logical, if TRUE then the (natural) logarithm of the survival function is returned.

Details

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.

Can only be used after decorating with ExoticStatistics.

Value

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

R6 Usage

$survival(x1, log = FALSE)

See Also

ExoticStatistics and decorate

---

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.

Details

The survival anti-derivative is defined by

$$as(a,b) = \int_a^b S_X(x)dx$$

where X is the distribution, S_X is the survival function of the distribution X and a, b are the limits of integration.

Can only be used after decorating with `ExoticStatistics`.

Value

Antiderivative of the survival function evaluated between limits as a numeric.

R6 Usage

$survivalAntiDeriv(lower = NULL, upper = NULL)

See Also

`ExoticStatistics` and `decorate`

---

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.

Details

The p-norm of the survival function is defined by

$$(\int_a^b |S_X|^p d\mu)^{1/p}$$

where X is the distribution, S_X is the survival function and a,b are the limits of integration.

Returns NULL if distribution is not continuous.

Can only be used after decorating with `ExoticStatistics`.

Value

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

R6 Usage

$survivalPNorm(object, p = 2, lower = NULL, upper = NULL)

See Also

ExoticStatistics and decorate

---

symmetry *Symmetry Accessor*

Description

Returns the distribution symmetry.

Usage

```
symmetry(object)
```

Arguments

object	Distribution.

Value

One of "symmetric" or "asymmetric".

R6 Usage

$symmetry()

See Also

properties

testContinuous *assert/check/test/Continuous*

Description

Validation checks to test if Distribution is continuous.

Usage

```
testContinuous(object)

checkContinuous(object)

assertContinuous(object)
```

Arguments

object	Distribution

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

```
testContinuous(Binomial$new()) # FALSE
```

testDiscrete *assert/check/test/Discrete*

Description

Validation checks to test if Distribution is discrete.

Usage

```
testDiscrete(object)

checkDiscrete(object)

assertDiscrete(object)
```

Arguments

object	Distribution

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` *assert/check/test/Distribution*

Description

Validation checks to test if a given object is an R6 Distribution.

Usage

```
testDistribution(object)

checkDistribution(object)

assertDistribution(object)
```

Arguments

`object`	object to test

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 R6 Distributions.

Usage

```
testDistributionList(object)

checkDistributionList(object)

assertDistributionList(object)
```

Arguments

object	object to test

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
```

testLeptokurtic *assert/check/test/Leptokurtic*

Description

Validation checks to test if Distribution is leptokurtic.

Usage

```
testLeptokurtic(object)

checkLeptokurtic(object)

assertLeptokurtic(object)
```

Arguments

object	Distribution

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

```
testLeptokurtic(Binomial$new())
```

---

testMatrixvariate *assert/check/test/Matrixvariate*

Description

Validation checks to test if Distribution is matrixvariate.

Usage

```
testMatrixvariate(object)

checkMatrixvariate(object)

assertMatrixvariate(object)
```

Arguments

object	Distribution

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
```

testMesokurtic *assert/check/test/Mesokurtic*

Description

Validation checks to test if Distribution is mesokurtic.

Usage

```
testMesokurtic(object)

checkMesokurtic(object)

assertMesokurtic(object)
```

Arguments

object	Distribution

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 *assert/check/test/Mixture*

Description

Validation checks to test if Distribution is mixture.

Usage

```
testMixture(object)

checkMixture(object)

assertMixture(object)
```

Arguments

object	Distribution

Value

If check passes then `assert` returns invisibly and `test`/`check` return `TRUE`. If check fails, `assert` stops code with error, `check` returns an error message as string, `test` returns `FALSE`.

Examples

```
testMixture(Binomial$new()) # FALSE
```

`testMultivariate` *assert/check/test/Multivariate*

Description

Validation checks to test if Distribution is multivariate.

Usage

```
testMultivariate(object)

checkMultivariate(object)

assertMultivariate(object)
```

Arguments

`object`	Distribution

Value

If check passes then `assert` returns invisibly and `test`/`check` return `TRUE`. If check fails, `assert` stops code with error, `check` returns an error message as string, `test` returns `FALSE`.

Examples

```
testMultivariate(Binomial$new()) # FALSE
```

testNegativeSkew *assert/check/test/NegativeSkew*

Description

Validation checks to test if Distribution is negative skew.

Usage

```
testNegativeSkew(object)

checkNegativeSkew(object)

assertNegativeSkew(object)
```

Arguments

object	Distribution

Value

If check passes then `assert` returns invisibly and `test`/`check` return `TRUE`. If check fails, `assert` stops code with error, `check` returns an error message as string, `test` returns `FALSE`.

Examples

```
testNegativeSkew(Binomial$new())
```

testNoSkew *assert/check/test/NoSkew*

Description

Validation checks to test if Distribution is no skew.

Usage

```
testNoSkew(object)

checkNoSkew(object)

assertNoSkew(object)
```

Arguments

object	Distribution

Value

If check passes then `assert` returns invisibly and `test`/`check` return `TRUE`. If check fails, `assert` stops code with error, `check` returns an error message as string, `test` returns `FALSE`.

Examples

```
testNoSkew(Binomial$new())
```

`testPlatykurtic` *assert/check/test/Platykurtic*

Description

Validation checks to test if Distribution is platykurtic.

Usage

```
testPlatykurtic(object)

checkPlatykurtic(object)

assertPlatykurtic(object)
```

Arguments

`object`	Distribution

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())
```

testPositiveSkew *assert/check/test/PositiveSkew*

Description

Validation checks to test if Distribution is positive skew.

Usage

```
testPositiveSkew(object)

checkPositiveSkew(object)

assertPositiveSkew(object)
```

Arguments

object	Distribution

Value

If check passes then `assert` returns invisibly and `test`/`check` return `TRUE`. If check fails, `assert` stops code with error, `check` returns an error message as string, `test` returns `FALSE`.

Examples

```
testPositiveSkew(Binomial$new())
```

testSymmetric *assert/check/test/Symmetric*

Description

Validation checks to test if Distribution is symmetric.

Usage

```
testSymmetric(object)

checkSymmetric(object)

assertSymmetric(object)
```

Arguments

object	Distribution

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

```
testSymmetric(Binomial$new()) # FALSE
```

`testUnivariate` *assert/check/test/Univariate*

Description

Validation checks to test if Distribution is univariate.

Usage

```
testUnivariate(object)

checkUnivariate(object)

assertUnivariate(object)
```

Arguments

`object`	Distribution

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.

R6 Usage

$traits()

Triangular *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,

$f(x) = 0, x < a$
$f(x) = 2(x-a)/((b-a)(c-a)), a \leq x < c$
$f(x) = 2/(b-a), x = c$
$f(x) = 2(b-x)/((b-a)(b-c)), c < x \leq b$
$f(x) = 0, x > b$ for $a, b, c \in R$, $a \leq c \leq b$.

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

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Triangular$new(lower = 0, upper = 1, mode = 0.5, symmetric = FALSE, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
lower	numeric	lower limit.
upper	numeric	upper limit.
mode	numeric	mode.
symmetric	logical	see details.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Triangular distribution is parameterised with lower, upper and mode as numerics. If symmetric = TRUE then the mode parameter is determined automatically and is defined by

$$mode = (lower + upper)/2$$

this cannot be changed after construction. If symmetric = FALSE (default) then mode can be updated after construction.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm

type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

Statistical Methods	**Link**
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**

`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `Uniform` for the Uniform distribution.

Examples

```
Triangular$new(lower = 2, upper = 5, symmetric = TRUE)
Triangular$new(lower = 2, upper = 5, symmetric = FALSE) # Note mode defaults to a symmetric shape
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

x = Triangular$new(lower = -1, upper = 1)

# Update parameters
x$setParameterValue(lower = 2, upper = 7)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`TriangularKernel` *Triangular Kernel*

Description

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

$$f(x) = 1 - |x|$$

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

Value

Returns an R6 object inheriting from class Kernel.

Constructor

TriangularKernel$new(decorators = NULL)

Constructor Arguments

Argument	Type	Details
`decorators`	Decorator	decorators to add functionality.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`

support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	Link
prec()	prec
stdev()	stdev
mode()	mode
mean()	mean.Distribution
median()	median.Distribution
iqr()	iqr
correlation()	correlation

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.

`qqplot()`	Coming Soon.

`Tricube` *Tricube Kernel*

Description

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

$$f(x) = 70/81(1 - |x|^3)^3$$

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

Details

The cdf and quantile functions are omitted as no closed form analytic expressions could be found, decorate with FunctionImputation for numeric results.

Value

Returns an R6 object inheriting from class Kernel.

Constructor

Tricube$new(decorators = NULL)

Constructor Arguments

Argument	Type	Details
`decorators`	Decorator	decorators to add functionality.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	Link
prec()	prec
stdev()	stdev
mode()	mode
mean()	mean.Distribution
median()	median.Distribution
iqr()	iqr
correlation()	correlation

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

`Triweight` *Triweight Kernel*

Description

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

$$f(x) = 35/32(1 - x^2)^3$$

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.

Value

Returns an R6 object inheriting from class Kernel.

Constructor

Triweight$new(decorators = NULL)

Constructor Arguments

Argument	Type	Details
`decorators`	Decorator	decorators to add functionality.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	Link
prec()	prec
stdev()	stdev
mode()	mode
mean()	mean.Distribution
median()	median.Distribution
iqr()	iqr
correlation()	correlation

Parameter Methods	Link

`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

`truncate` *Truncate a Distribution*

Description

S3 functionality to truncate an R6 distribution.

Usage

```
truncate(x, lower = NULL, upper = NULL)
```

Arguments

`x`	Distribution.
`lower`	lower limit for truncation.
`upper`	upper limit for truncation.

See Also

`TruncatedDistribution`

`TruncatedDistribution` *Distribution Truncation Wrapper*

Description

A wrapper for truncating any probability distribution at given limits.

Details

Truncates a distribution at lower and upper limits, using the formulae

$$f_T(x) = f_X(x)/(F_X(upper) - F_X(lower))$$

$$F_T(x) = (F_X(x) - F_X(lower))/(F_X(upper) - F_X(lower))$$

where f_T/F_T is the pdf/cdf of the truncated distribution T = Truncate(X, lower, upper) and f_X, F_X is the pdf/cdf of the original distribution.

If lower or upper are NULL they are taken to be `self$inf()` and `self$sup()` respectively. The support of the new distribution is the interval of points between lower and upper.

The pdf and cdf of the distribution are required for this wrapper, if unavailable decorate with `FunctionImputation` first.

Value

Returns an R6 object of class TruncatedDistribution.

Constructor

TruncatedDistribution$new(distribution, lower = NULL, upper = NULL)

Constructor Arguments

Argument	Type	Details
`distribution`	distribution	Distribution to truncate.
`lower`	numeric	Lower limit for truncation.
`upper`	numeric	Upper limit for truncation.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
wrappedModels(model = NULL)	wrappedModels
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	Link
prec()	prec
stdev()	stdev
median()	median.Distribution
iqr()	iqr
cor()	cor

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link

`liesInSupport(x, all = TRUE, bound = FALSE)`	liesInSupport
`liesInType(x, all = TRUE, bound = FALSE)`	liesInType

Representation Methods	**Link**
`strprint(n = 2)`	strprint
`print(n = 2)`	print
`summary(full = T)`	summary.Distribution
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

See Also

listWrappers, FunctionImputation, truncate

Examples

```
truncBin <- TruncatedDistribution$new(
            Binomial$new(prob = 0.5, size = 10),
            lower = 2, upper = 4)
truncBin$getParameterValue("prob")
```

---

type *Type Accessor*

Description

Returns the scientific type of the distribution.

Usage

```
type(object)
```

Arguments

object	Distribution.

Value

An R6 object of class SetInterval.

R6 Usage

$type()

See Also

SetInterval

`type.SetInterval` *SetInterval Type Accessor*

Description

Returns the SetInterval 'type', one of "()","(]","[]","[)".

Details

This is an R6 method only, no S3 dispatch is available.

Value

Returns the type of SetInterval, one of "()","(]","[]","[)".

R6 Usage

$type()

See Also

SetInterval

`Uniform` *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,

$$f(x) = 1/(b-a)$$

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

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

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Uniform$new(lower = 0, upper = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
lower	integer	lower distribution limit.
upper	integer	upper distribution limit.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Uniform distribution is parameterised with lower and upper as numerics.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType

kurtosisType()	kurtosisType

Statistical Methods	**Link**
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions.

Examples

```
x <- Uniform$new(lower = -10, upper = 5)

# Update parameters
x$setParameterValue(lower = 2, upper = 7)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`UniformKernel` *Uniform Kernel*

Description

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

$$f(x) = 1/2$$

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

Value

Returns an R6 object inheriting from class Kernel.

Constructor

UniformKernel$new(decorators = NULL)

Constructor Arguments

Argument	Type	Details
decorators	Decorator	decorators to add functionality.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	Link
prec()	prec
stdev()	stdev

`mode()`	`mode`
`mean()`	`mean.Distribution`
`median()`	`median.Distribution`
`iqr()`	`iqr`
`correlation()`	`correlation`

Parameter Methods	**Link**
`parameters(id)`	`parameters`
`getParameterValue(id, error = "warn")`	`getParameterValue`
`setParameterValue(..., lst = NULL, error = "warn")`	`setParameterValue`

Validation Methods	**Link**
`liesInSupport(x, all = TRUE, bound = FALSE)`	`liesInSupport`
`liesInType(x, all = TRUE, bound = FALSE)`	`liesInType`

Representation Methods	**Link**
`strprint(n = 2)`	`strprint`
`print(n = 2)`	`print`
`summary(full = T)`	`summary.Distribution`
`plot()`	Coming Soon.
`qqplot()`	Coming Soon.

`union.SetInterval` *Symbolic Unions for SetInterval*

Description

Makes a symbolic representation for the union of sets/intervals.

Usage

```
union.SetInterval(..., dim = 1)

## S3 method for class 'SetInterval'
x + y
```

Arguments

x	SetInterval
y	SetInterval
...	SetIntervals to take the union of.
dim	dimension of new SetInterval.

Details

This does not calculate the union of the arguments but is just a symbolic representation using unicode.

Value

Returns an R6 object inheriting from SetInterval.

See Also

product.SetInterval, complement.SetInterval, power.SetInterval

Examples

```
PosNaturals$new() * Reals$new()
product.SetInterval(PosNaturals$new(), Reals$new())
```

---

update.ParameterSet *Updates a ParameterSet*

Description

Updates parameter in a ParameterSet using updateFuncs.

Usage

```
## S3 method for class 'ParameterSet'
update(object, ...)
```

Arguments

object	ParameterSet
...	ignored, added for S3 consistency

Details

In general this method should never need to be called manually by the user as it is internally called in setParameterValue.

The method works by cycling through parameters in a ParameterSet that have non-NA updateFuncs and parses these as expressions, thereby updating their values.

Value

An R6 object of class ParameterSet.

R6 Usage

$update()

See Also

ParameterSet

`valueSupport` *Value Support Accessor*

Description

Returns the valueSupport of the distribution.

Usage

```
valueSupport(object)
```

Arguments

`object`	Distribution.

Value

One of "discrete"/"continuous"/"mixture".

R6 Usage

$valueSupport()

`variance` *Distribution Variance*

Description

The variance or covariance of a distribution, either calculated analytically if or estimated numerically.

Usage

```
variance(object)
```

Arguments

`object`	Distribution.

Details

The variance of a distribution is defined by the formula

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

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

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

Value

Variance as a numeric.

R6 Usage

$variance()

See Also

`CoreStatistics`, `decorate` and `genExp`.

variateForm *Variate Form Accessor*

Description

Returns the variateForm of the distribution.

Usage

```
variateForm(object)
```

Arguments

object	Distribution.

Value

One of "univariate"/"multivariate"/"matrixvariate".

R6 Usage

$variateForm()

VectorDistribution *Vectorise Distributions*

Description

A wrapper for creating a vector of distributions.

Details

A vector of distributions has the following relationship

$$f_V(X1 = x1, ..., XN = xN) = f_{X1}(x1), ..., f_{XN}(xn)$$

$$F_V(X1 = x1, ..., XN = xN) = F_{X1}(x1), ..., F_{XN}(xn)$$

where f_V/F_V is the pdf/cdf of the vector of distributions V and $X1, ..., XN$ are distributions.

Value

Returns an R6 object of class VectorDistribution.

Constructor

VectorDistribution$new(distlist = NULL, distribution = NULL, params = NULL, name = NULL, short_name = NULL, description = NULL)

Constructor Arguments

Argument	Type	Details
distlist	list	List of distributions.
distribution	distribution	Distribution to wrap.
params	a R object	Either list of parameters or matrix-type frame, see examples.
name	list	Optional new name for distribution.
short_name	list	Optional new short_name for distribution.
description	list	Optional new description for distribution.

Constructor Details

A vector distribution can either be constructed by a list of distributions passed to `distlist` or by passing the name of a distribution implemented in distr6 to `distribution`, as well as a list or table of parameters to `params`. The former case provides more flexibility in the ability to use multiple distributions but the latter is useful for quickly combining many distributions of the same type. See examples.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
wrappedModels(model = NULL)	wrappedModels
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType
kurtosisType()	kurtosisType

d/p/q/r Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand

Statistical Methods	Link
prec()	prec
stdev()	stdev
median()	median.Distribution
iqr()	iqr
cor()	cor

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

See Also

listWrappers and ProductDistribution

Examples

```
vecBin <- VectorDistribution$new(list(Binomial$new(prob = 0.5,
```

```
                            size = 10), Normal$new(mean = 15)))
vecBin$pdf(x1 = 2, x2 =3)
vecBin$cdf(1:5, 12:16)
vecBin$quantile(c(0.1,0.2),c(0.3,0.4))
vecBin$rand(10)

vecBin = VectorDistribution$new(distribution = Binomial,
       params = list(list(prob = 0.1, size = 2),
                   list(prob = 0.6, size = 4),
                   list(prob = 0.2, size = 6)))
vecBin$pdf(x1=1,x2=2,x3=3)
vecBin$cdf(x1=1,x2=2,x3=3)
vecBin$rand(10)

#Equivalently
vecBin = VectorDistribution$new(distribution = Binomial,
       params = data.table::data.table(prob = c(0.1,0.6,0.2), size = c(2,4,6)))
vecBin$pdf(x1=1,x2=2,x3=3)
vecBin$cdf(x1=1,x2=2,x3=3)
vecBin$rand(10)
```

`Wald` *Wald Distribution Class*

Description

Mathematical and statistical functions for the Wald distribution, which is commonly used for modelling the first passage time for Brownian motion.

Details

The Wald distribution parameterised with mean, μ, and shape, λ, is defined by the pdf,

$$f(x) = (\lambda/(2x^3\pi))^{1/2} exp((-\lambda(x-\mu)^2)/(2\mu^2 x))$$

for $\lambda > 0$ and $\mu > 0$.

The distribution is supported on the Positive Reals.

`entropy` is omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results. `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.

Constructor

Wald$new(mean = 1, shape = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
`mean`	numeric	location parameter.
`shape`	numeric	shape parameter.
`decorators`	Decorator	decorators to add functionality. See details.
`verbose`	logical	if TRUE parameterisation messages produced.

Constructor Details

The Wald distribution is parameterised with `mean` and `shape` as positive numerics.

Public Variables

Variable	Return
`name`	Name of distribution.
`short_name`	Id of distribution.
`description`	Brief description of distribution.
`package`	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
`decorators()`	`decorators`
`traits()`	`traits`
`valueSupport()`	`valueSupport`
`variateForm()`	`variateForm`
`type()`	`type`
`properties()`	`properties`
`support()`	`support`
`symmetry()`	`symmetry`
`sup()`	`sup`
`inf()`	`inf`
`dmax()`	`dmax`
`dmin()`	`dmin`
`skewnessType()`	`skewnessType`
`kurtosisType()`	`kurtosisType`

Statistical Methods	Link
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	Link
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	Link
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	Link
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

Michael, John R.; Schucany, William R.; Haas, Roy W. (May 1976). "Generating Random Variates Using Transformations with Multiple Roots". The American Statistician. 30 (2): 88–90. doi:10.2307/2683801. JSTOR 2683801.

See Also

`listDistributions` for all available distributions. `Normal` for the Normal distribution. `CoreStatistics` for numerical results. `FunctionImputation` to numerically impute d/p/q/r.

Examples

```
x = Wald$new(mean = 2, shape = 5)

# Update parameters
x$setParameterValue(shape = 3)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`Weibull` *Weibull Distribution Class*

Description

Mathematical and statistical functions for the Weibull distribution, which is commonly used in survival analysis and is a special case of the Generalized Extreme Value distribution.

Details

The Weibull distribution parameterised with shape, α, and scale, β, is defined by the pdf,

$$f(x) = (\alpha/\beta)(x/\beta)^{\alpha-1} exp(-x/\beta)^{\alpha}$$

for $\alpha, \beta > 0$.

The distribution is supported on the Positive Reals.

`mgf` and `cf` are omitted as no closed form analytic expression could be found, decorate with `CoreStatistics` for numerical results.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

Weibull$new(shape = 1, scale = 1, decorators = NULL, verbose = FALSE)

Constructor Arguments

Argument	Type	Details
shape	numeric	shape parameter.
scale	numeric	scale parameter.
decorators	Decorator	decorators to add functionality. See details.
verbose	logical	if TRUE parameterisation messages produced.

Constructor Details

The Weibull distribution is parameterised with shape and scale as positive numerics.

Public Variables

Variable	Return
name	Name of distribution.
short_name	Id of distribution.
description	Brief description of distribution.
package	The package d/p/q/r are implemented in.

Public Methods

Accessor Methods	Link
decorators()	decorators
traits()	traits
valueSupport()	valueSupport
variateForm()	variateForm
type()	type
properties()	properties
support()	support
symmetry()	symmetry
sup()	sup
inf()	inf
dmax()	dmax
dmin()	dmin
skewnessType()	skewnessType

kurtosisType()	kurtosisType

Statistical Methods	**Link**
pdf(x1, ..., log = FALSE, simplify = TRUE)	pdf
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	cdf
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)	quantile.Distribution
rand(n, simplify = TRUE)	rand
mean()	mean.Distribution
variance()	variance
stdev()	stdev
prec()	prec
cor()	cor
skewness()	skewness
kurtosis(excess = TRUE)	kurtosis
entropy(base = 2)	entropy
mgf(t)	mgf
cf(t)	cf
pgf(z)	pgf
median()	median.Distribution
iqr()	iqr

Parameter Methods	**Link**
parameters(id)	parameters
getParameterValue(id, error = "warn")	getParameterValue
setParameterValue(..., lst = NULL, error = "warn")	setParameterValue

Validation Methods	**Link**
liesInSupport(x, all = TRUE, bound = FALSE)	liesInSupport
liesInType(x, all = TRUE, bound = FALSE)	liesInType

Representation Methods	**Link**
strprint(n = 2)	strprint
print(n = 2)	print
summary(full = T)	summary.Distribution
plot()	Coming Soon.
qqplot()	Coming Soon.

References

McLaughlin, M. P. (2001). A compendium of common probability distributions (pp. 2014-01). Michael P. McLaughlin.

See Also

`listDistributions` for all available distributions. `Frechet` and `Gumbel` for other special cases of the generalized extreme value distribution. `CoreStatistics` for numerical results.

Examples

```
x <- Weibull$new(shape = 2, scale = 3)

# Update parameters
x$setParameterValue(scale = 1)
x$parameters()

# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)

# Statistics
x$mean()
x$variance()

summary(x)
```

`wrappedModels` *Gets Internally Wrapped Models*

Description

Returns either a list of all the wrapped models or the models named by parameters.

Usage

```
wrappedModels(object, model = NULL)
```

Arguments

`object`	Distribution.
`model`	character, see details.

Details

Accessor for internally wrapped models. If the `model` parameter is matched by a single named wrapped model, this model is returned. If a vector is supplied to `model` parameter then a list of internal models is returned if matched, otherwise a list of all internal models is returned. If `model` is NULL (default) then a list of all internal models are returned.

Value

If `model` is NULL then returns list of models that are wrapped by the wrapper. Otherwise returns model given in `model`.

R6 Usage

$wrappedModels(model = NULL)

See Also

`DistributionWrapper`

Index