Package ‘distr6’

February 20, 2020

Title The Complete R6 Probability Distributions Interface

Version 1.3.4


Imports checkmate, data.table, R6, R62S3 (>= 1.4.0), set6 (>= 0.1.2), stats

Suggests expint, GoFKernel, knitr, testthat, devtools, rmarkdown, magrittr, extraDistr, actuar, remotes, pracma

License MIT + file LICENSE

LazyData true


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

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.0.2

Collate 'helpers.R' 'distr6_globals.R' 'Distribution.R'
  'DistributionDecorator.R'
  'DistributionDecorator_CoreStatistics.R'
  'DistributionDecorator_ExoticStatistics.R'
  'DistributionDecorator_FunctionImputation.R'
  'Distribution_Kernel.R' 'Distribution_SDistribution.R'
  'Kernel_Cosine.R' 'Kernel_Epanechnikov.R' 'Kernel_Logistic.R'
  'Kernel_Normal.R' 'Kernel_Quartic.R' 'Kernel_Sigmoid.R'

1
'Kernel_Silverman.R' 'Kernel_Triangular.R' 'Kernel_Tricube.R'
'Kernel_Triweight.R' 'Kernel_Uniform.R' 'ParameterSet.R'
'SDistribution_Arcsine.R' 'SDistribution_Bernoulli.R'
'SDistribution_Beta.R' 'SDistribution_BetaNoncentral.R'
'SDistribution_Binomial.R' 'SDistribution_Categorical.R'
'SDistribution_Cauchy.R' 'SDistribution_ChiSquared.R'
'SDistribution_ChiSquaredNoncentral.R'
'SDistribution_Degenerate.R' 'SDistribution_Dirichlet.R'
'SDistribution_Dirichlet.R' 'SDistribution_Empirical.R'
'SDistribution_Exponential.R' 'SDistribution_FDistribution.R'
'SDistribution_FDistributionNoncentral.R'
'SDistribution_Frechet.R' 'SDistribution_Gamma.R'
'SDistribution_Geometric.R' 'SDistribution_Gompertz.R'
'SDistribution_Gumbel.R' 'SDistribution_Hypergeometric.R'
'SDistribution_InverseGamma.R' 'SDistribution_Laplace.R'
'SDistribution_Logarithmic.R' 'SDistribution_Logistic.R'
'SDistribution_Loglogistic.R' 'SDistribution_Lognormal.R'
'SDistribution_Multinomial.R'
'SDistribution_MultivariateNormal.R'
'SDistribution_NegBinomal.R' 'SDistribution_Normal.R'
'SDistribution_Pareto.R' 'SDistribution_Poisson.R'
'SDistribution_Rayleigh.R' 'SDistribution_StudentT.R'
'SDistribution_StudentTNoncentral.R'
'SDistribution_Triangular.R' 'SDistribution_Uniform.R'
'SDistribution_Wald.R' 'SDistribution_Weibull.R'
'SDistribution_WeightedDiscrete.R' 'Wrapper.R'
'Wrapper_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'
'c.Distribution.R' 'decomposeMixture.R' 'decorate.R'
'distr6-deprecated.R' 'distr6.R' 'distr6.news.R'
'distrSimulate.R' 'exkurtosisType.R' 'generalPNorm.R'
'getParameterSet.R' 'lines_continuous.R' 'lines_discrete.R'
'lines.R' 'listDecorators.R' 'listDistributions.R'
'listKernels.R' 'listSpecialSets.R' 'listWrappers.R'
'makeUniqueDistributions.R' 'measures.R' 'plot_continuous.R'
'plot_discrete.R' 'plot.R' 'qqplot.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],
R topics documented:

Chloe Zhaoyuan Gu [ctb],
Yunjie He [ctb],
Xiaowen Huang [ctb],
Shuhan Liu [ctb],
Runlong Yu [ctb],
Chijing Zeng [ctb],
Qian Zhou [ctb]

Maintainer Raphael Sonabend <raphael.sonabend.15@ucl.ac.uk>
Repository CRAN
Date/Publication 2020-02-19 23:30:02 UTC

R topics documented:

distr6-package .......................................................... 6
Arcsine ................................................................. 8
as.data.table.ParameterSet ........................................... 11
as.ParameterSet ....................................................... 11
Bernoulli ............................................................... 12
Beta ................................................................. 15
BetaNoncentral ......................................................... 18
Binomial ............................................................. 21
c.Distribution ......................................................... 24
Categorical ........................................................... 25
Cauchy ............................................................... 28
cdf ................................................................. 31
cdfAntiDeriv .......................................................... 32
cdfPNorm ........................................................... 33
cf ................................................................. 34
ChiSquared .......................................................... 35
ChiSquaredNoncentral ............................................... 38
Convolution .......................................................... 41
CoreStatistics ......................................................... 42
correlation ........................................................... 43
Cosine ............................................................... 44
cumHazard ............................................................ 46
decorate ............................................................... 47
decorators ............................................................. 48
Degenerate ........................................................... 48
Dirichlet .............................................................. 51
DiscreteUniform ....................................................... 55
distr6News ............................................................. 58
Distribution .......................................................... 58
DistributionDecorator .................................................. 61
DistributionWrapper ................................................... 62
distrSimulate .......................................................... 64
dmax ................................................................. 65
dmin ................................................................. 66
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empirical</td>
<td>66</td>
</tr>
<tr>
<td>entropy</td>
<td>69</td>
</tr>
<tr>
<td>Epanechnikov</td>
<td>70</td>
</tr>
<tr>
<td>exkurtosisType</td>
<td>72</td>
</tr>
<tr>
<td>ExoticStatistics</td>
<td>73</td>
</tr>
<tr>
<td>Exponential</td>
<td>74</td>
</tr>
<tr>
<td>Extract.VectorDistribution</td>
<td>78</td>
</tr>
<tr>
<td>FDistribution</td>
<td>78</td>
</tr>
<tr>
<td>FDistributionNoncentral</td>
<td>81</td>
</tr>
<tr>
<td>Frechet</td>
<td>84</td>
</tr>
<tr>
<td>FunctionImputation</td>
<td>87</td>
</tr>
<tr>
<td>Gamma</td>
<td>89</td>
</tr>
<tr>
<td>generalPNorm</td>
<td>92</td>
</tr>
<tr>
<td>genExp</td>
<td>93</td>
</tr>
<tr>
<td>Geometric</td>
<td>94</td>
</tr>
<tr>
<td>getParameterSupport</td>
<td>97</td>
</tr>
<tr>
<td>getParameterValue</td>
<td>98</td>
</tr>
<tr>
<td>Gompertz</td>
<td>99</td>
</tr>
<tr>
<td>Gumbel</td>
<td>102</td>
</tr>
<tr>
<td>hazard</td>
<td>105</td>
</tr>
<tr>
<td>huberize</td>
<td>106</td>
</tr>
<tr>
<td>HuberizedDistribution</td>
<td>106</td>
</tr>
<tr>
<td>Hypergeometric</td>
<td>109</td>
</tr>
<tr>
<td>inf</td>
<td>112</td>
</tr>
<tr>
<td>InverseGamma</td>
<td>113</td>
</tr>
<tr>
<td>iqr</td>
<td>116</td>
</tr>
<tr>
<td>isCdf</td>
<td>117</td>
</tr>
<tr>
<td>isPdf</td>
<td>117</td>
</tr>
<tr>
<td>isQuantile</td>
<td>118</td>
</tr>
<tr>
<td>isRand</td>
<td>118</td>
</tr>
<tr>
<td>Kernel</td>
<td>119</td>
</tr>
<tr>
<td>kthmoment</td>
<td>121</td>
</tr>
<tr>
<td>kurtosis</td>
<td>122</td>
</tr>
<tr>
<td>kurtosisType</td>
<td>123</td>
</tr>
<tr>
<td>Laplace</td>
<td>123</td>
</tr>
<tr>
<td>liesInSupport</td>
<td>126</td>
</tr>
<tr>
<td>liesInType</td>
<td>127</td>
</tr>
<tr>
<td>lines.Distribution</td>
<td>128</td>
</tr>
<tr>
<td>listDecorators</td>
<td>129</td>
</tr>
<tr>
<td>listDistributions</td>
<td>130</td>
</tr>
<tr>
<td>listKernels</td>
<td>131</td>
</tr>
<tr>
<td>listWrappers</td>
<td>131</td>
</tr>
<tr>
<td>Logarithmic</td>
<td>132</td>
</tr>
<tr>
<td>Logistic</td>
<td>135</td>
</tr>
<tr>
<td>LogisticKernel</td>
<td>138</td>
</tr>
<tr>
<td>Loglogistic</td>
<td>140</td>
</tr>
<tr>
<td>Lognormal</td>
<td>143</td>
</tr>
<tr>
<td>makeUniqueDistributions</td>
<td>147</td>
</tr>
</tbody>
</table>
R topics documented:

- mean.Distribution .................................................. 148
- median.Distribution ................................................. 149
- merge.ParameterSet .................................................. 149
- mgf .................................................................... 150
- MixtureDistribution .................................................. 151
- mode ................................................................... 153
- Multinomial .............................................................. 154
- MultivariateNormal ..................................................... 157
- NegativeBinomial ......................................................... 161
- Normal .................................................................. 165
- NormalKernel .............................................................. 168
- parameters ................................................................. 170
- ParameterSet ............................................................... 171
- Pareto .................................................................. 173
- pdf ..................................................................... 176
- pdfPNorm .................................................................. 177
- pgf ..................................................................... 178
- plot.Distribution .......................................................... 179
- Poisson .................................................................. 180
- prec .................................................................... 183
- print.ParameterSet ...................................................... 184
- ProductDistribution ..................................................... 185
- properties ................................................................. 188
- qqplot .................................................................. 188
- quantile.Distribution .................................................. 189
- Quartic .................................................................. 190
- rand .................................................................... 192
- Rayleigh ................................................................. 193
- SDistribution ............................................................... 196
- setParameterValue ....................................................... 198
- Sigmoid ................................................................. 199
- Silverman ............................................................... 201
- simulateEmpiricalDistribution ...................................... 203
- skewness ................................................................. 204
- skewnessType ........................................................... 205
- skewType ................................................................. 206
- squared2Norm ............................................................ 207
- stdev .................................................................. 208
- strprint ................................................................. 208
- StudentT ............................................................... 209
- StudentTNoncentral ..................................................... 212
- summary.Distribution .................................................. 215
- sup ..................................................................... 216
- support ................................................................. 217
- survival ............................................................... 217
- survivalAntiDeriv ......................................................... 218
- survivalPNorm ............................................................ 219
- symmetry ............................................................... 220
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


Author(s)

Maintainer: Raphael Sonabend <raphael.sonabend.15@ucl.ac.uk> (ORCID)
Authors:
- Franz Kiraly <f.kiraly@ucl.ac.uk>

Other contributors:
- Peter Ruckdeschel <peter.ruckdeschel@uni-oldenburg.de> (Author of distr) [contributor]
- Matthias Kohl <Matthias.Kohl@stamats.de> (Author of distr) [contributor]
- 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://github.com/alan-turing-institute/distr6/
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,

\[
f(x) = \frac{1}{\pi \sqrt{(x - a)(b - x)}}
\]
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 \( \text{rbeta} \) is used for sampling, otherwise via inverse transform

Value
Returns an R6 object inheriting from class SDistribution.

Constructor
\[
\text{Arcsine}\$new(\text{lower} = 0, \text{upper} = 1, \text{decorators} = \text{NULL}, \text{verbose} = \text{FALSE})
\]

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>lower</td>
<td>integer</td>
<td>lower distribution limit.</td>
</tr>
<tr>
<td>upper</td>
<td>integer</td>
<td>upper distribution limit.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

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

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
</table>
Arcsine

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**

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

**Statistical Methods**

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

**Link**

- decorators
- traits
- valueSupport
- variateForm
- type
- properties
- support
- symmetry
- sup
- inf
- dmax
- dmin
- skewnessType
- kurtosisType
- pdf
- cdf
- quantile.Distribution
- rand
- mean.Distribution
- variance
- stdev
- prec
- cor
- skewness
- kurtosis
- entropy
- mgf
- cf
- pgf
- median.Distribution
- iqr
- mode
**Parameter Methods**
parameters(id)  
getParameterValue(id, error = "warn")  
setParameterValue(..., lst = NULL, error = "warn")

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

**Representation Methods**
strprint(n = 2)  
print(n = 2)  
summary(full = T)

**References**

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

**Examples**
```r
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()
```
**as.data.table.ParameterSet**

Coerce ParameterSet to data.table

**Description**

Coerces a ParameterSet to a data.table.

**Usage**

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

**Arguments**

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

**Value**

A data.table.

**See Also**

`ParameterSet`

---

**as.ParameterSet**

Coerce to a ParameterSet

**Description**

Coerces objects to ParameterSet.

**Usage**

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

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

```r
## 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,

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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>prob</td>
<td>numeric</td>
<td>probability of success.</td>
</tr>
<tr>
<td>qprob</td>
<td>numeric</td>
<td>probability of failure.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
</tr>
</tbody>
</table>
**Statistical Methods**

- `pdf(x1, ..., log = FALSE, simplify = TRUE)`
- `cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`
- `quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`
- `rand(n, simplify = TRUE)`
- `mean()`
- `variance()`
- `stdev()`
- `prec()`
- `cor()`
- `skewness()`
- `kurtosis(excess = TRUE)`
- `entropy(base = 2)`
- `mgf(t)`
- `cf(t)`
- `pgf(z)`
- `median()`
- `iqr()`
- `mode(which = "all")`

**Parameter Methods**

- `parameters(id)`
- `getParameterValue(id, error = "warn")`
- `setParameterValue(..., lst = NULL, error = "warn")`

**Validation Methods**

- `liesInSupport(x, all = TRUE, bound = FALSE)`
- `liesInType(x, all = TRUE, bound = FALSE)`

**Representation Methods**

- `strprint(n = 2)`
- `print(n = 2)`
- `summary(full = T)`

**References**

See Also

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

Examples

```r
# Can be parameterised with probability of success or failure
Bernoulli$new(prob = 0.2)
Bernoulli$new(qprob = 0.3)

x = Bernoulli$new(verbos = 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, \( \alpha, \beta \), is defined by the pdf,

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

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

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

`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**

\[
\text{Beta}\$\text{new}(\text{shape1} = 1, \text{shape2} = 1, \text{decorators} = \text{NULL}, \text{verbose} = \text{FALSE})
\]

**Constructor Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape1, shape2</td>
<td>numeric</td>
<td>positive shape parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

**Constructor Details**

The Beta distribution is parameterised with \text{shape1} and \text{shape2} as positive numerics.

**Public Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

**Public Methods**

**Accessor Methods**

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
</tr>
</tbody>
</table>
**Statistical Methods**

- `pdf(x1, ..., log = FALSE, simplify = TRUE)`
- `cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`
- `quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`
- `rand(n, simplify = TRUE)`
- `mean()`
- `variance()`
- `stdev()`
- `prec()`
- `cor()`
- `skewness()`
- `kurtosis(excess = TRUE)`
- `entropy(base = 2)`
- `mgf(t)`
- `cf(t)`
- `pgf(z)`
- `median()`
- `iqr()`
- `mode(which = "all")`

**Parameter Methods**

- `parameters(id)`
- `getParameterValue(id, error = "warn")`
- `setParameterValue(..., lst = NULL, error = "warn")`

**Validation Methods**

- `liesInSupport(x, all = TRUE, bound = FALSE)`
- `liesInType(x, all = TRUE, bound = FALSE)`

**Representation Methods**

- `strprint(n = 2)`
- `print(n = 2)`
- `summary(full = T)`
References

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)

BetaNoncentral Noncentral Beta Distribution Class

Description

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

Details

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

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

for \(\alpha, \beta > 0, \lambda \geq 0\), where \(B\) is the Beta function.
The distribution is supported on \([0, 1]\).
mean, variance, skewness, kurtosis, entropy, mode, mgf and cf are omitted as no closed form
analytic expression could be found, decorate with CoreStatistics for numerical results.
BetaNoncentral

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

BetaNoncentral$new(shape1 = 1, shape2 = 1, location = 0, decorators = NULL, verbose = FALSE)

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape1, shape2</td>
<td>numeric</td>
<td>positive shape parameter.</td>
</tr>
<tr>
<td>location</td>
<td>numeric</td>
<td>location (ncp in rstats).</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

The Noncentral Beta distribution is parameterised with shape1, shape2 as positive numerics, location as non-negative numeric.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
</tbody>
</table>
skewnessType()
skewnessType
kurtosisType()
kurtosisType

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

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

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

**Representation Methods**
strprint(n = 2)
print(n = 2)
summary(full = T)
**Binomial**

**Author(s)**

Jordan Deenichin

**References**


**See Also**

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

**Examples**

```r
x = BetaNoncentral$new(shape1 = 2, shape2 = 5, location = 3)
# Update parameters
x$setParameterValue(shape1 = 1)
x$parameters()
# d/p/q/r
x$pdf(5)
x$cdf(5)
x$quantile(0.42)
x$rand(4)
summary(x)
```

---

**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 \( p \in [0, 1] \), where \( C(a, b) \) is the combination (or binomial coefficient) function. The distribution is supported on \( 0, 1, \ldots, n \).

**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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>numeric</td>
<td>number of trials.</td>
</tr>
<tr>
<td>prob</td>
<td>numeric</td>
<td>probability of success.</td>
</tr>
<tr>
<td>qprob</td>
<td>numeric</td>
<td>probability of failure.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
</tbody>
</table>
Binomial

\texttt{dmin()}
\texttt{skewnessType()}
\texttt{kurtosisType()}

**Statistical Methods**
\texttt{pdf(x1, \ldots, \log = \text{FALSE}, \text{ simplify = TRUE})}
\texttt{cdf(x1, \ldots, \text{ lower.tail = TRUE, log.p = FALSE, simplify = TRUE})}
\texttt{quantile(p, \ldots, \text{ lower.tail = TRUE, log.p = FALSE, simplify = TRUE})}
\texttt{rand(n, \text{ simplify = TRUE})}
\texttt{mean()}
\texttt{variance()}
\texttt{stdev()}
\texttt{prec()}
\texttt{cor()}
\texttt{skewness()}
\texttt{kurtosis(excess = \text{TRUE})}
\texttt{entropy(base = 2)}
\texttt{mgf(t)}
\texttt{cf(t)}
\texttt{pgf(z)}
\texttt{median()} 
\texttt{iqr()}
\texttt{mode(\text{which = "all"})}

**Parameter Methods**
\texttt{parameters(id)}
\texttt{getParameterValue(id, \text{ error = "warn"})}
\texttt{setParameterValue(\ldots, \text{ lst = NULL, error = "warn"})}

**Validation Methods**
\texttt{liesInSupport(x, \text{ all = TRUE, bound = FALSE})}
\texttt{liesInType(x, \text{ all = TRUE, bound = FALSE})}

**Representation Methods**
\texttt{strprint(n = 2)}
\texttt{print(n = 2)}
\texttt{summary(full = T)}
References


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设定ParameterValue(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)

---

c.Distribution Combine Distributions into a VectorDistribution

Description

Helper function for quickly combining distributions into a VectorDistribution.

Usage

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

Arguments

... distributions to be concatenated.
name, short_name, description, decorators

See VectorDistribution
Categorical

Value
A VectorDistribution

See Also
VectorDistribution

Examples

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

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

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

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, $x_1,\ldots,x_k$, and respective probabilities, $p_1,\ldots,p_k$, 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 $x_1,\ldots,x_k$.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>ANY</td>
<td>elements in the support Set. See details.</td>
</tr>
<tr>
<td>probs</td>
<td>numeric</td>
<td>vector of probabilities. See details.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

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

\[
probs = \frac{probs}{\text{sum}(probs)}
\]

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

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
</tbody>
</table>
**Categorical**

dmin()
skewnessType()
kurtosisType()

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

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

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

**Representation Methods**
strprintf(n = 2)
print(n = 2)
summary(full = T)
Cauchy

References


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, $\alpha$, and scale, $\beta$, is defined by the pdf,

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

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

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.
Cauchy

Constructor

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

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>numeric</td>
<td>location parameter.</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>scale parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

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

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
</tr>
</tbody>
</table>
Statistical Methods

pdf(x1, ..., log = FALSE, simplify = TRUE)
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
rand(n, simplify = TRUE)
mean()
variance()
stddev()
prec()
cor()
skewness()
kurtosis(excess = TRUE)
entropy(base = 2)
mgf(t)
cf(t)
pgf(z)
median()
iqr()
mode(which = "all")

Parameter Methods

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

Validation Methods

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

Representation Methods

strprint(n = 2)
print(n = 2)
summary(full = T)

Author(s)

Chijing Zeng
References

See Also
listDistributions for all available distributions.

Examples

\begin{verbatim}
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)
\end{verbatim}

---

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

\begin{verbatim}
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.
\end{verbatim}

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

```r
$\text{cdf}(x_1, \ldots, \text{lower.tail} = \text{TRUE}, \text{log.p} = \text{FALSE}, \text{simplify} = \text{TRUE})$
```

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

```r
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. |
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 \texttt{ExoticStatistics}.

Value
Antiderivative of the cdf evaluated between limits as a numeric.

R6 Usage
\[ \texttt{cdfAntiDeriv(lower = NULL, upper = NULL)} \]

See Also
\texttt{ExoticStatistics} and \texttt{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**
- \texttt{object} Distribution.
- \texttt{p} p-norm to calculate.
- \texttt{lower} lower limit for integration, default is infimum.
- \texttt{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 \texttt{ExoticStatistics}.
Value

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

R6 Usage

$\text{ScdfPNorm}(\text{object, p = 2, lower = NULL, upper = NULL})$

See Also

ExoticStatistics and decorate

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

$$c_{\text{f}_X}(t) = E_{\text{f}_X}[\exp(xt)]$$

where $X$ is the distribution and $E_{\text{f}_X}$ 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

$\text{Scf}(t)$

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, $\nu$, is defined by the pdf,

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

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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>df</td>
<td>numeric</td>
<td>degrees of freedom.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

**Constructor Details**

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

**Public Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>
Public Methods

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

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

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

Link
- decorators
- traits
- valueSupport
- variateForm
- type
- properties
- support
- symmetry
- sup
- inf
- dmax
- dmin
- skewnessType
- kurtosisType
- pdf
- cdf
- quantile.Distribution
- rand
- mean.Distribution
- variance
- stdev
- prec
- skewness
- kurtosis
- entropy
- mgf
- cf
- pgf
- median.Distribution
- iqr
- mode
- parameters
- getParameterValue
- setParameterValue
Validation Methods
liesInSupport(x, all = TRUE, bound = FALSE)
liesInType(x, all = TRUE, bound = FALSE)

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

References

See Also
listDistributions for all available distributions. Normal for the Normal distribution, ChiSquaredNoncentral for the noncentral Chi-Squared distribution.

Examples
x = ChiSquared$new(df = 2)

# Update parameters
x$setParameterValue(location = 3)
x$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)
ChiSquaredNoncentral  
*Noncentral Chi-Squared Distribution Class*

**Description**

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

**Details**

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

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

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

The distribution is supported on the Positive Reals.

entropy and mode 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**

ChiSquaredNoncentral$new(df = 1, location = 0, decorators = NULL, verbose = FALSE)

**Constructor Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>df</td>
<td>numeric</td>
<td>degrees of freedom.</td>
</tr>
<tr>
<td>location</td>
<td>numeric</td>
<td>location (ncp in rstats).</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

**Constructor Details**

The Noncentral Chi-Squared distribution is parameterised with \(df\) and \(location\) as non-negative numerics.

**Public Variables**
**ChiSquaredNoncentral**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

### Public Methods

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

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

**Link**
- decorators
- traits
- valueSupport
- variateForm
- type
- properties
- support
- symmetry
- sup
- inf
- dmax
- dmin
- skewnessType
- kurtosisType
- pdf
- cdf
- quantile.Distribution
- rand
- mean.Distribution
- variance
- stdev
- prec
- cor
- skewness
- kurtosis
- entropy
- mgf
- cf
- pgf
- median.Distribution
- iqr
- mode
ChiSquaredNoncentral

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

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

Author(s)
Jordan Deenichin

References

See Also
listDistributions for all available distributions. Normal for the Normal distribution, ChiSquared for the central Chi-Squared distribution. CoreStatistics for numerical results.

Examples
x = ChiSquaredNoncentral$new(df = 2, location = 2)

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

---

## 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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>dist1</td>
<td>distribution</td>
<td>First distribution in convolution.</td>
</tr>
<tr>
<td>dist2</td>
<td>distribution</td>
<td>Second distribution in convolution.</td>
</tr>
<tr>
<td>add</td>
<td>logical</td>
<td>Add or subtract distributions.</td>
</tr>
<tr>
<td>type</td>
<td>logical</td>
<td>Type of new distribution, automated if NULL.</td>
</tr>
</tbody>
</table>
CoreStatistics

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>distribution</td>
<td>distribution</td>
<td>Distribution to decorate.</td>
</tr>
</tbody>
</table>

Added Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Name</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>mgf(t)</td>
<td>Moment generating function</td>
<td>mgf</td>
</tr>
<tr>
<td>pgf(t)</td>
<td>Probability generating function</td>
<td>pgf</td>
</tr>
<tr>
<td>cf(t)</td>
<td>Characteristic function</td>
<td>cf</td>
</tr>
<tr>
<td>entropy(base = 2)</td>
<td>(Shannon) Entropy</td>
<td>entropy</td>
</tr>
</tbody>
</table>
Correlation of a distribution.

**Usage**

```r
correlation(object)
```

**Arguments**

- `object` : Distribution.

**Details**

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

\[ \rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} \]

where \( \sigma_{XY} \) is the covariance of X and Y and \( \sigma_X, \sigma_Y \) 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.
Cosine

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,

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

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

Value

Returns an R6 object inheriting from class Kernel.

Constructor

Cosine$new(decorators = NULL)

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality.</td>
</tr>
</tbody>
</table>

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods
Cosine

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

**d/p/q/r Methods**

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

**Statistical Methods**
squared2Norm()
prec()
stdev()
mode(which = "all")
mean()
median()
iqr()
correlation()

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

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

$\text{ScumHazard}(x1, \text{log} = \text{FALSE})$

See Also

ExoticStatistics and decorate
**Description**

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

**Usage**

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

**Arguments**

- `distribution` distribution to decorate
- `decorators` list or vector of decorators. See Details.

**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 or their names 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**

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

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

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

```r
E <- Exponential$new()
decorate(E, c("CoreStatistics", "ExoticStatistics"))
```

---

### Decorators

<table>
<thead>
<tr>
<th>decorators</th>
<th>Decorators Accessor</th>
</tr>
</thead>
</table>

#### 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.
The Degenerate distribution parameterised with mean, $\mu$ is defined by the pmf,

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

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

The distribution is supported on $\mu$.

Also known as the Dirac distribution.

Value

Returns a R6 object inheriting from class SDistribution.

Constructor

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

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>numeric</td>
<td>location parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

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

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

`decorators()`

`traits()`
valueSupport.
variateForm.
type.
properties.
support.
symmetry.
sup.
in.
dmax.
dmin.
skewnessType.
kurtosisType.

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

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

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

Representation Methods

<table>
<thead>
<tr>
<th>strprint(n = 2)</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>print(n = 2)</td>
<td>strprint</td>
</tr>
<tr>
<td>summary(full = T)</td>
<td>print</td>
</tr>
<tr>
<td>summary.Distribution</td>
<td></td>
</tr>
</tbody>
</table>

References


See Also

listDistributions for all available distributions.

Examples

```r
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)
```
Details

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

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

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

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

$mgf$ 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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>params</td>
<td>numeric</td>
<td>vector of concentration parameters.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>
**Public Methods**

**Link**
- decorators
- traits
- valueSupport
- variateForm
- type
- properties
- support
- symmetry
- sup
- inf
- dmax
- dmin
- skewnessType
- kurtosisType

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

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

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

**Link**
- pdf
- cdf
- quantile.Distribution
- rand
- mean.Distribution
- variance
- stdev
- prec
- cor
- skewness
- kurtosis
- entropy
- mgf
- cf
- pgf
- median.Distribution
- iqr
- mode
- parameters
- getParameterValue
- setParameterValue
**Validation Methods**

- `liesInSupport(x, all = TRUE, bound = FALSE)`  
- `liesInType(x, all = TRUE, bound = FALSE)`

**Representation Methods**

- `strprint(n = 2)`  
- `print(n = 2)`  
- `summary(full = T)`

**References**


**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()
```
DiscreteUniform

\[ x \text{\$variance()} \]

\[ \text{summary}(x) \]

---

**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,

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

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

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

**Value**

Returns an R6 object inheriting from class SDistribution.

**Constructor**

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

**Constructor Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>lower</td>
<td>integer</td>
<td>lower distribution limit.</td>
</tr>
<tr>
<td>upper</td>
<td>integer</td>
<td>upper distribution limit.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

**Constructor Details**

The Discrete Uniform distribution is parameterised with \( \text{lower} \) and \( \text{upper} \) as whole numbers.

**Public Variables**
### Variable | Return
---|---
nname | 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

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

### Statistical Methods

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

- `parameters(id)`
- `getParameterValue(id, error = "warn")`
- `setParameterValue(..., lst = NULL, error = "warn")`

**Validation Methods**

- `liesInSupport(x, all = TRUE, bound = FALSE)`
- `liesInType(x, all = TRUE, bound = FALSE)`

**Representation Methods**

- `strprint(n = 2)`
- `print(n = 2)`
- `summary(full = T)`

**References**


**See Also**

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

**Examples**

```r
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()
```
**distr6News**

*Show distr6 NEWS.md File*

**Description**

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

**Usage**

```r
distr6News()
```

**Value**

NEWS.md in viewer.

**Examples**

```r
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**

```r
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, suppressMoments = TRUE)
```
Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>character</td>
<td>Full name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>character</td>
<td>Short name to identify distribution.</td>
</tr>
<tr>
<td>type</td>
<td>set6::Set</td>
<td>Scientific type.</td>
</tr>
<tr>
<td>support</td>
<td>set6::Set</td>
<td>Distribution support. See Details.</td>
</tr>
<tr>
<td>symmetric</td>
<td>logical</td>
<td>Is distribution symmetric?</td>
</tr>
<tr>
<td>pdf</td>
<td>function</td>
<td>See Details.</td>
</tr>
<tr>
<td>cdf</td>
<td>function</td>
<td>See Details.</td>
</tr>
<tr>
<td>quantile</td>
<td>function</td>
<td>See Details.</td>
</tr>
<tr>
<td>rand</td>
<td>function</td>
<td>See Details.</td>
</tr>
<tr>
<td>parameters</td>
<td>ParameterSet</td>
<td>See Details.</td>
</tr>
<tr>
<td>decorators</td>
<td>list</td>
<td>R6 decorators to add in construction.</td>
</tr>
<tr>
<td>valueSupport</td>
<td>character</td>
<td>continuous, discrete, mixture. See Details.</td>
</tr>
<tr>
<td>variateForm</td>
<td>character</td>
<td>univariate, multivariate, matrixvariate. See Details.</td>
</tr>
<tr>
<td>description</td>
<td>character</td>
<td>Short description of distribution.</td>
</tr>
<tr>
<td>suppressMoments</td>
<td>character</td>
<td>See Details.</td>
</tr>
</tbody>
</table>

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 a set6::Set 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.

suppressMoments can be used to prevent the skewness and kurtosis type being automatically calculated in construction. This has the benefit of drastically decreasing computational time but at the cost of losing these in the distribution properties.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
</tbody>
</table>
Distribution

Brief description of distribution.

Public Methods

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

Link
- decorators
- traits
- valueSupport
- variateForm
- type
- properties
- support
- symmetry
- sup
- inf
- dmax
- dmin
- skewnessType
- kurtosisType

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

Link
- pdf
- cdf
- quantile.Distribution
- rand

Statistical Methods
- prec()
- stdev()
- median()
- iqr()
- correlation()

Link
- prec
- stdev
- median.Distribution
- iqr
- correlation

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

Link
- parameters
- getParameterValue
- setParameterValue
DistributionDecorator

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

Representation Methods
strprint(n = 2) strprint
print(n = 2) print
summary(full = T) summary.Distribution

Active Bindings

<table>
<thead>
<tr>
<th>Active Binding</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>isPdf</td>
<td>isPdf</td>
</tr>
<tr>
<td>isCdf</td>
<td>isCdf</td>
</tr>
<tr>
<td>isQuantile</td>
<td>isQuantile</td>
</tr>
<tr>
<td>isRand</td>
<td>isRand</td>
</tr>
</tbody>
</table>

See Also
See set6 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.
DistributionWrapper

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

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

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>wrappedModels(model = NULL)</td>
<td>wrappedModels</td>
</tr>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
</tbody>
</table>
skewnessType()  
skewnessType

kurtosisType()  
kurtosisType

d/p/q/r Methods
pdf(x1, ..., log = FALSE, simplify = TRUE)  
Link
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
prec()  
prec
stdev()  
stdev
median()  
median.Distribution
iqr()  
iqr
cor()  
cor

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

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

Representation Methods
strprint(n = 2)  
strprint
print(n = 2)  
print
summary(full = T)  
summary.Distribution

Active Bindings

<table>
<thead>
<tr>
<th>Active Binding</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>isPdf</td>
<td>isPdf</td>
</tr>
<tr>
<td>isCdf</td>
<td>isCdf</td>
</tr>
</tbody>
</table>
**Public Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

**See Also**

listWrappers

---

**distrSimulate**

*Simulate from a Distribution*

**Description**

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

**Usage**

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

**Arguments**

- **n**: number of points to simulate.
- **distribution**: distribution to simulate from, corresponds to ClassName of distr6 distribution, abbreviations allowed.
- **pars**: parameters to pass to distribution. If omitted then distribution defaults used.
Simplify

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

See Also

rand

---

### dmax

**Distribution Maximum Accessor**

**Description**

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

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

**Usage**

dmax(object)

**Arguments**

- object: Distribution.

**Value**

Maximum as a numeric.

**R6 Usage**

$\text{dmax}()$

**See Also**

support, dmin, sup, inf
Empirical Distribution Minimum Accessor

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

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

Usage
dmin(object)

Arguments
object Distribution.

Value
Minimum as a numeric.

R6 Usage
$\text{dmin}()$

See Also
support, dmax, sup, inf

Empirical Distribution Class

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

Details
The Empirical distribution is defined by the pmf,

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

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

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

Sampling from this distribution is performed with the 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).
Empirical

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

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

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>samples</td>
<td>numeric</td>
<td>vector of observed samples.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

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

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
</tr>
</tbody>
</table>
Statistical Methods
pdf(x1, ..., log = FALSE, simplify = TRUE)
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
rand(n, simplify = TRUE)
mean()
variance()
stdev()
prec()
cor()
skewness()
kurtosis(excess = TRUE)
entropy(base = 2)
mgf(t)
cf(t)
pgf(z)
median()
iqr()
mode(which = "all")

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

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)
entropy

References


See Also

listDistributions for all available distributions. sample for the sampling function and WeightedDiscrete for the closely related WeightedDiscrete distribution.

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)

---

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)
Argument Type Details

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality.</td>
</tr>
</tbody>
</table>

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

**Accessor Methods**

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

**d/p/q/r Methods**

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

**Statistical Methods**

- squared2Norm()
- prec()
exkurtosisType

Parameter Methods
parameters(id)
get_Parameter_Value(id, error = "warn")
set_Parameter_Value(..., lst = NULL, error = "warn")

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

Representation Methods
strprint(n = 2)
print(n = 2)
supplyy(full = T)

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>distribution</td>
<td>distribution</td>
<td>Distribution to decorate.</td>
</tr>
</tbody>
</table>

Added Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Name</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>survival(x1, log = FALSE)</td>
<td>Survival function</td>
<td>survival</td>
</tr>
<tr>
<td>hazard(x1, log = FALSE)</td>
<td>Hazard function</td>
<td>hazard</td>
</tr>
<tr>
<td>cumHazard(x1, log = FALSE)</td>
<td>Cumulative hazard function</td>
<td>cumHazard</td>
</tr>
<tr>
<td>cdfAntiDeriv(lower = NULL, upper = NULL))</td>
<td>Anti-derivative of cdf</td>
<td>cdfAntiDeriv</td>
</tr>
<tr>
<td>survivalAntiDeriv(lower = NULL, upper = NULL)</td>
<td>Anti-derivative of survival function</td>
<td>survivalAntiDeriv</td>
</tr>
<tr>
<td>cdfPNorm(p = 2, lower = NULL, upper = NULL)</td>
<td>P-norm of cdf</td>
<td>cdfPNorm</td>
</tr>
<tr>
<td>pdfPNorm(p = 2, lower = NULL, upper = NULL)</td>
<td>P-norm of pdf</td>
<td>pdfPNorm</td>
</tr>
<tr>
<td>survivalPNorm(p = 2, lower = NULL, upper = NULL)</td>
<td>P-norm of survival function</td>
<td>survivalPNorm</td>
</tr>
</tbody>
</table>

See Also

decorate, listDecorators

Examples

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

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

---

**Exponential Class Description**

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

The Exponential distribution parameterised with rate, \( \lambda \), is defined by the pdf,

\[
f(x) = \lambda \exp(-x\lambda)
\]

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate</td>
<td>numeric</td>
<td>arrival rate.</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>scale parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

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

\[\text{scale} = 1/\text{rate}\]

If scale is given then rate is ignored.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
</tbody>
</table>
traits()
valueSupport()
variateForm()
type()
properties()
support()
symmetry()
sup()
inf()
dmax()
dmin()
skewnessType()
kurtosisType()

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

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

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

<table>
<thead>
<tr>
<th>Representation Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>strprint(n = 2)</td>
<td>strprint</td>
</tr>
<tr>
<td>print(n = 2)</td>
<td>print</td>
</tr>
<tr>
<td>summary(full = T)</td>
<td>summary.Distribution</td>
</tr>
</tbody>
</table>

**References**


**See Also**

`listDistributions` for all available distributions.

**Examples**

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

x = Exponential$new(verbos = 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)
```
Extract.VectorDistribution

Extract one or more Distributions from a VectorDistribution

Description

Once a VectorDistribution has been constructed, use [ to extract one or more Distributions from inside it.

Usage

Extract.VectorDistribution(vecdist, i)

## S3 method for class 'VectorDistribution'
vecdist[i]

Arguments

vecdist VectorDistribution from which to extract Distributions.
i indices specifying distributions to extract.

FDistribution 'F' Distribution Class

Description

Mathematical and statistical functions for the 'F' distribution, which is commonly used in ANOVA testing and is the ratio of scaled Chi-Squared distributions.

Details

The 'F' distribution parameterised with two degrees of freedom parameters, $\mu, \nu$, is defined by the pdf,

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

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.
**FDistribution**

**Constructor**

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

**Constructor Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>df1, df2</td>
<td>numeric</td>
<td>degrees of freedom.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

**Constructor Details**

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

**Public Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

**Public Methods**

**Accessor Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
</tr>
</tbody>
</table>

**Statistical Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Link</th>
</tr>
</thead>
</table>

pdf(x1, ..., log = FALSE, simplify = TRUE)
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
rand(n, simplify = TRUE)
mean()
variance()
stddev()
prec()
cor()
skewness()
kurtosis(excess = TRUE)
entropy(base = 2)
mgf(t)
cf(t)
pgf(z)
median()
iqr()
mode(which = "all")

Parameter Methods

- parameters(id)
- getParameterValue(id, error = "warn")
- setParameterValue(..., lst = NULL, error = "warn")

Validation Methods

- liesInSupport(x, all = TRUE, bound = FALSE)
- liesInType(x, all = TRUE, bound = FALSE)

Representation Methods

- strprint(n = 2)
- print(n = 2)
- summary(full = T)

References

FDistributionNoncentral

See Also

listDistributions for all available distributions. Normal, ChiSquared and FDistributionNoncentral for the Normal, Chi-Squared and noncentral F 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)

FDistributionNoncentral

Noncentral F Distribution Class

Description

Mathematical and statistical functions for the Noncentral F distribution, which is commonly used in ANOVA testing and is the ratio of scaled Chi-Squared distributions.

Details

The Noncentral F distribution parameterised with two degrees of freedom parameters, \( \mu, \nu \), and location, \( \lambda \), is defined by the pdf,

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

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

The distribution is supported on the Positive Reals.

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

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

FDistributionNoncentral$new(df1 = 1, df2 = 1, location = 0, decorators = NULL, verbose = FALSE)

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>df1, df2</td>
<td>numeric</td>
<td>degrees of freedom.</td>
</tr>
<tr>
<td>location</td>
<td>numeric</td>
<td>location (ncp in rstats).</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

The Noncentral F distribution is parameterised with df1 and df2 as positive numerics, location as non-negative numeric.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

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

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

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

**Validation Methods**
iliesInSupport(x, all = TRUE, bound = FALSE)
iliesInType(x, all = TRUE, bound = FALSE)

**Representation Methods**
strprint(n = 2)
print(n = 2)
summary(full = T)
Author(s)
Jordan Deenichin

References

See Also
listDistributions for all available distributions. Normal, ChiSquared and FDistribution for the Normal, Chi-Squared and central F distributions. CoreStatistics for numerical results.

Examples
x <- FDistributionNoncentral$new(df1 = 1, df2 = 3, location = 2)

# 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 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,

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

for \( \alpha, \beta \in R^+ \) and \( \gamma \in R \).
The distribution is supported on $x > \gamma$.

`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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape</td>
<td>numeric</td>
<td>shape parameter.</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>scale parameter.</td>
</tr>
<tr>
<td>minimum</td>
<td>numeric</td>
<td>location parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

**Constructor Details**

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

**Public Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

**Public Methods**

**Accessor Methods**

<table>
<thead>
<tr>
<th>Link</th>
<th>Accessor Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>decorators()</td>
</tr>
<tr>
<td></td>
<td>traits()</td>
</tr>
<tr>
<td></td>
<td>valueSupport()</td>
</tr>
<tr>
<td></td>
<td>variateForm()</td>
</tr>
<tr>
<td></td>
<td>type()</td>
</tr>
</tbody>
</table>
Frechet properties() properties
support() support
symmetry() symmetry
sup() sup
inf() inf
dmax() dmax
dmin() dmin
skewnessType() skewnessType
kurtosisType() kurtosisType

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

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

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

Representation Methods
FunctionImputation

strprint(n = 2) strprint
print(n = 2) print
summary(full = T) summary.Distribution

References


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)

---

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>distribution</td>
<td>distribution</td>
<td>Distribution to decorate.</td>
</tr>
</tbody>
</table>

Added Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Name</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdf(x1, ..., log = FALSE, simplify = TRUE)</td>
<td>Density/mass function</td>
<td>pdf</td>
</tr>
<tr>
<td>cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)</td>
<td>Distribution function</td>
<td>cdf</td>
</tr>
<tr>
<td>quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)</td>
<td>Quantile function</td>
<td>quantile.Distribution</td>
</tr>
<tr>
<td>rand(n, simplify = TRUE)</td>
<td>Simulation function</td>
<td>rand</td>
</tr>
</tbody>
</table>

See Also
`decorate, listDecorators`

Examples
```r
x = Distribution$new("Test", pdf = function(x) 1/(4-1),
support = set6::Interval$new(1,4),
type = set6::Reals$new())
decorate(x, FunctionImputation)
```
**Description**

Mathematical and statistical functions for the Gamma distribution, which is commonly used as the prior in Bayesian modelling, the convolution of exponential distributions, and to model waiting times.

**Details**

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

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

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

The distribution is supported on the Positive Reals.

**Value**

Returns an R6 object inheriting from class SDistribution.

**Constructor**

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

**Constructor Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape</td>
<td>numeric</td>
<td>shape parameter.</td>
</tr>
<tr>
<td>rate</td>
<td>numeric</td>
<td>inverse scale parameter.</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>scale parameter.</td>
</tr>
<tr>
<td>mean</td>
<td>numeric</td>
<td>alternate scale parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
<td>valueSupport</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
<td>variateForm</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>support</td>
<td>type</td>
</tr>
<tr>
<td>variateForm()</td>
<td>properties</td>
<td>type</td>
</tr>
<tr>
<td>type()</td>
<td>properties</td>
<td>support</td>
</tr>
<tr>
<td>properties()</td>
<td>support</td>
<td>symmetry</td>
</tr>
<tr>
<td>support()</td>
<td>symmetry</td>
<td>sup</td>
</tr>
<tr>
<td>symmetry()</td>
<td>sup</td>
<td>inf</td>
</tr>
<tr>
<td>sup()</td>
<td>inf</td>
<td>dmax</td>
</tr>
<tr>
<td>inf()</td>
<td>dmax</td>
<td>dmin</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmin</td>
<td>skewnessType</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
<td>kurtosisType</td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
<td></td>
</tr>
</tbody>
</table>

Statistical Methods

<table>
<thead>
<tr>
<th>Statistical Methods</th>
<th>Link</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdf(x1,..., log = FALSE, simplify = TRUE)</td>
<td>pdf</td>
<td>mean.Distribution</td>
</tr>
<tr>
<td>cdf(x1,..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)</td>
<td>cdf</td>
<td>mean.Distribution</td>
</tr>
<tr>
<td>quantile(p,..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)</td>
<td>quantile.Distribution</td>
<td></td>
</tr>
<tr>
<td>rand(n, simplify = TRUE)</td>
<td>rand</td>
<td>mean.Distribution</td>
</tr>
<tr>
<td>mean()</td>
<td>mean</td>
<td>variance</td>
</tr>
<tr>
<td>variance()</td>
<td>variance</td>
<td>stdev</td>
</tr>
<tr>
<td>stdev()</td>
<td>stdev</td>
<td>prec</td>
</tr>
<tr>
<td>prec()</td>
<td>prec</td>
<td></td>
</tr>
</tbody>
</table>
### Gamma

```r
cor()
skewness()
kurtosis(excess = TRUE)
entropy(base = 2)
mgf(t)
cf(t)
pgf(z)
median()
iqr()
mode(which = "all")
```

### Parameter Methods

- `parameters(id)`
- `getParameterValue(id, error = "warn")`
- `setParameterValue(..., lst = NULL, error = "warn")`

### Validation Methods

- `liesInSupport(x, all = TRUE, bound = FALSE)`
- `liesInType(x, all = TRUE, bound = FALSE)`

### Representation Methods

- `strprint(n = 2)`
- `print(n = 2)`
- `summary(full = T)`

### References


### See Also

- `listDistributions` for all available distributions.

### Examples

```r
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,

\[
\left( \int_S |f|^{p} d\mu \right)^{1/p}
\]

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)

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

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**

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

#### Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>prob</td>
<td>numeric</td>
<td>probability of success.</td>
</tr>
<tr>
<td>qprob</td>
<td>numeric</td>
<td>probability of failure.</td>
</tr>
<tr>
<td>trials</td>
<td>logical</td>
<td>number of trials or failures, see details.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

**Public Methods**

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

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

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

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

Representation Methods
strprint(n = 2)              strprint
print(n = 2)                 print
summary(full = T)            summary.Distribution

References

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**

```r
generateParameterSupport(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 `set6::Set` object.

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

**Value**

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

**R6 Usage**

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

**See Also**

`parameters`

---

**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`
Gompertz Distribution Class

Description
Mathematical and statistical functions for the Gompertz distribution, which is commonly used in survival analysis particularly to model adult mortality rates.

Details
The Gompertz distribution parameterised with shape, $\alpha$, and scale, $\beta$, is defined by the pdf,

$$f(x) = \alpha \beta \exp(x\beta) \exp(\alpha) \exp(-\exp(x\beta)\alpha)$$

for $\alpha, \beta > 0$.

The distribution is supported on the Non-Negative Reals.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape</td>
<td>numeric</td>
<td>positive shape parameter.</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>positive scale parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

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

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
</table>
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

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

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

Link
- decorators
decorators
- traits
- valueSupport
- variateForm
- type
- properties
- support
- symmetry
- sup
- inf
dmax
dmin
- skewnessType
- kurtosisType
- pdf
cdf
- quantile.Distribution
- rand
- mean.Distribution
- variance
- stdev
- prec
cor
- skewness
- kurtosis
- entropy
- mgf
cf
- pgf
- median.Distribution
- iqr
- mode
**Parameter Methods**
- `parameters(id)`
- `getParameterValue(id, error = "warn")`
- `setParameterValue(..., lst = NULL, error = "warn")`

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

**Representation Methods**
- `strprint(n = 2)`
- `print(n = 2)`
- `summary(full = T)`

**References**

**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 Distribution Class

Description

Mathematical and statistical functions for the Gumbel distribution, which is commonly used to model the maximum (or minimum) of a number of samples of different distributions, and is a special case of the Generalised Extreme Value distribution.

Details

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

\[
f(x) = \frac{\exp(-z + \exp(-z))}{\beta}
\]

for \( z = \frac{x - \mu}{\beta}, \mu \in \mathbb{R} \) and \( \beta > 0 \).

The distribution is supported on the Reals.

Apery’s Constant to 16 significant figures is used in the skewness calculation. The \texttt{gammaz} function from the \texttt{pracma} package is used in the \texttt{cf} to allow complex inputs.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

\texttt{Gumbel$new(location = 0, scale = 1, decorators = NULL, verbose = FALSE) }

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>numeric</td>
<td>location parameter.</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>scale parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

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

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
</table>
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

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

Statistical Methods

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

\begin{Verbatim}
parameters(id)
getParameterValue(id, error = "warn")
setParameterValue(..., lst = NULL, error = "warn")
\end{Verbatim}

Link
\begin{Verbatim}
parameters
getParameterValue
setParameterValue
\end{Verbatim}

Validation Methods

\begin{Verbatim}
liesInSupport(x, all = TRUE, bound = FALSE)
liesInType(x, all = TRUE, bound = FALSE)
\end{Verbatim}

Link
\begin{Verbatim}
liesInSupport
liesInType
\end{Verbatim}

Representation Methods

\begin{Verbatim}
strprint(n = 2)
print(n = 2)
summary(full = T)
\end{Verbatim}

Link
\begin{Verbatim}
strprint
print
summary.Distribution
\end{Verbatim}

References


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

\begin{Verbatim}
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()
\end{Verbatim}
### hazard

**Hazard Function**

#### Description

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

#### Usage

```r
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

```r
$\text{hazard}(x1, \log = \text{FALSE})
```

#### See Also

`ExoticStatistics` and `decorate`
huberize  

**Huberization**

**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

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

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

If lower or upper are NULL they are taken to be \( \text{self}$\text{inf()} \) and \( \text{self}$\text{sup()} \) 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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>distribution</td>
<td>distribution</td>
<td>Distribution to huberize.</td>
</tr>
<tr>
<td>lower</td>
<td>numeric</td>
<td>Lower limit for huberization.</td>
</tr>
<tr>
<td>upper</td>
<td>numeric</td>
<td>Upper limit for huberization.</td>
</tr>
</tbody>
</table>

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

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

Link
wrappedModels
decorators
traits
valueSupport
variateForm
type
properties
support
symmetry
sup
inf
dmax
dmin
skewnessType
kurtosisType

d/p/q/r Methods

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

Link
dpdf
cdf
HuberizedDistribution

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

Statistical Methods
prec()
stdev()
median()
iqr()
cor()

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

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

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 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 \( \{\max(0, n + K - N), \ldots, \min(n, K)\} \).

\( \text{mgf} \) and \( \text{cf} \) are omitted as no closed form analytic expression could be found, decorate with \texttt{CoreStatistics} for numerical results.

Value
Returns an R6 object inheriting from class SDistribution.

Constructor

\[
\text{Hypergeometric}\$\text{new}(\text{size} = 10, \text{successes} = 5, \text{failures} = \text{NULL}, \text{draws} = 2, \text{decorators} = \text{NULL}, \text{verbose} = \text{FALSE})
\]

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>numeric</td>
<td>population size.</td>
</tr>
<tr>
<td>successes</td>
<td>numeric</td>
<td>number of population successes.</td>
</tr>
<tr>
<td>failures</td>
<td>numeric</td>
<td>number of population failures.</td>
</tr>
<tr>
<td>draws</td>
<td>numeric</td>
<td>number of draws.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>
Hypergeometric 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,

\[ \text{failures} = \text{size} - \text{successes} \]

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

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

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

Statistical Methods

- pdf(x1, ..., log = FALSE, simplify = TRUE)
- cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- rand(n, simplify = TRUE)
- mean()
- variance()
- stdev()
- prec()
- cor()
Hypergeometric

skewness()
kurtosis(excess = TRUE)
entropy(base = 2)
mgf(t)
cf(t)
pgf(z)
median()
iqr()
mode(which = "all")

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

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

References
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$new(verbos = TRUE)
# Update parameters
# When any parameter is updated, all others are too!
x$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
InverseGamma

Inverse Gamma Distribution Class

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

Details
The Inverse Gamma distribution parameterised with shape, $\alpha$, and scale, $\beta$, is defined by the pdf,

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

for $\alpha, \beta > 0$, where $\Gamma$ 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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape</td>
<td>numeric</td>
<td>shape parameter.</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>scale parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

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

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
</table>


In the document, there is a table and a list of public methods for the InverseGamma distribution. The table includes:

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

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

### Link
- `decorators`
- `traits`
- `valueSupport`
- `variateForm`
- `type`
- `properties`
- `support`
- `symmetry`
- `sup`
- `inf`
- `dmax`
- `dmin`
- `skewnessType`
- `kurtosisType`
- `pdf`
- `cdf`
- `quantile.Distribution`
- `rand`
- `mean.Distribution`
- `variance`
- `stdev`
- `prec`
- `cor`
- `skewness`
- `kurtosis`
- `entropy`
- `mgf`
- `cf`
- `pgf`
- `median.Distribution`
- `iqr`
- `mode`
Parameter Methods
parameters(id)
getParameterValue(id, error = "warn")
setParameterValue(..., lst = NULL, error = "warn")

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

References

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()
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
$\text{iqr}()$
isCdf

Test the Distribution Cdf Exist?

Description
Returns whether or not the distribution has a defined expression for the Cdf.

Value
Returns TRUE if an expression for the Cdf is defined for the distribution, FALSE otherwise.

R6 Usage
$\text{isCdf}$

See Also
$\text{isPdf}, \text{isQuantile}, \text{isRand}$

isPdf

Test the Distribution Pdf Exist?

Description
Returns whether or not the distribution has a defined expression for the pdf.

Value
Returns TRUE if an expression for the pdf is defined for the distribution, FALSE otherwise.

R6 Usage
$\text{isPdf}$

See Also
$\text{isCdf}, \text{isQuantile}, \text{isRand}$
isQuantile  
Test the Distribution Quantile Exist?

Description

Returns whether or not the distribution has a defined expression for the Quantile.

Value

Returns TRUE if an expression for the Quantile is defined for the distribution, FALSE otherwise.

R6 Usage

$\text{isQuantile}

See Also

\text{isPdf, isCdf, isRand}

isRand  
Test the Distribution Rand Exist?

Description

Returns whether or not the distribution has a defined expression for the Rand.

Value

Returns TRUE if an expression for the Rand is defined for the distribution, FALSE otherwise.

R6 Usage

$\text{isRand}

See Also

\text{isPdf, isCdf, isQuantile}
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

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>d/p/q/r Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdf(x1, ..., log = FALSE, simplify = TRUE)</td>
<td>pdf</td>
</tr>
<tr>
<td>cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)</td>
<td>cdf</td>
</tr>
<tr>
<td>quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)</td>
<td>quantile.Distribution</td>
</tr>
<tr>
<td>rand(n, simplify = TRUE)</td>
<td>rand</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statistical Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>squared2Norm()</td>
<td>squared2Norm</td>
</tr>
<tr>
<td>prec()</td>
<td>prec</td>
</tr>
<tr>
<td>stdev()</td>
<td>stdev</td>
</tr>
<tr>
<td>mode(which = &quot;all&quot;)</td>
<td>mode</td>
</tr>
<tr>
<td>mean()</td>
<td>mean.Distribution</td>
</tr>
<tr>
<td>median()</td>
<td>median.Distribution</td>
</tr>
</tbody>
</table>
iqr()
correlation()

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

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

Active Bindings

<table>
<thead>
<tr>
<th>Active Binding</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>isPdf</td>
<td>isPdf</td>
</tr>
<tr>
<td>isCdf</td>
<td>isCdf</td>
</tr>
<tr>
<td>isQuantile</td>
<td>isQuantile</td>
</tr>
<tr>
<td>isRand</td>
<td>isRand</td>
</tr>
</tbody>
</table>

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>
kthmoment

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
\[ CM(k) = E_X[(x - \mu)^k] \]
the kth standardised moment of a distribution is defined by
\[ SM(k) = \frac{CM(k)}{\sigma^k} \]
the kth raw moment of a distribution is defined by
\[ RM(k) = E_X[x^k] \]
where \( E_X \) is the expectation of distribution \( X \), \( \mu \) is the mean of the distribution and \( \sigma \) is the standard deviation of the distribution.
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,

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

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

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

Value

Kurtosis as a numeric.

R6 Usage

$\text{kurtosis}(\text{excess} = \text{TRUE})$

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**
$\text{kurtosisType}()$

**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, $\mu$, and scale, $\beta$, is defined by the pdf,

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

for $\mu \epsilon \mathbb{R}$ 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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>numeric</td>
<td>location parameter.</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>scale parameter.</td>
</tr>
<tr>
<td>var</td>
<td>numeric</td>
<td>alternate scale parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

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

\[ \text{var} = 2 \ast \text{scale}^2 \]

If var is given then scale is ignored.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package (d/p/q/r) are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
</tbody>
</table>
Laplace

\[ \text{sup()} \quad \text{sup} \]
\[ \text{inf()} \quad \text{inf} \]
\[ \text{dmax()} \quad \text{dmax} \]
\[ \text{dmin()} \quad \text{dmin} \]
\[ \text{skewnessType()} \quad \text{skewnessType} \]
\[ \text{kurtosisType()} \quad \text{kurtosisType} \]

**Statistical Methods**
- pdf\((x_1, \ldots, \log = \text{FALSE}, \text{simplify = TRUE})\)
- cdf\((x_1, \ldots, \text{lower.tail = TRUE, log.p = FALSE, simplify = TRUE})\)
- quantile\((p, \ldots, \text{lower.tail = TRUE, log.p = FALSE, simplify = TRUE})\)
- rand\((n, \text{simplify = TRUE})\)
- mean()
- variance()
- stdev()
- prec()
- cor()
- skewness()
- kurtosis\((\text{excess = TRUE})\)
- entropy\((\text{base = 2})\)
- mgf\((t)\)
- cf\((t)\)
- pgf\((z)\)
- median()
- iqr()
- mode\((\text{which = "all"})\)

**Parameter Methods**
- parameters\((id)\)
- getParameterValue\((id, \text{error = "warn"})\)
- setParameterValue\((\ldots, \text{lst = NULL, error = "warn"})\)

**Validation Methods**
- liesInSupport\((x, \text{all = TRUE, bound = FALSE})\)
- liesInType\((x, \text{all = TRUE, bound = FALSE})\)

**Representation Methods**
- strprint\((n = 2)\)
- print\((n = 2)\)
- summary\((\text{full = T})\)
**References**


**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)

---

`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)`
liesInType

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

lines.Distribution

Superimpose Distribution Functions Plots for a distr6 Object

Description

One of six plots can be selected to be superimposed in the plotting window, including: pdf, cdf, quantile, survival, hazard and cumulative hazard.

Usage

## S3 method for class 'Distribution'
lines(x, fun, npoints = 3000, ...)

Arguments

x distr6 object.

fun vector of functions to plot, one or more of: "pdf", "cdf", "quantile", "survival", "hazard", and "cumhazard"; partial matching available.

npoints number of evaluation points.

... graphical parameters.

Details

Unlike the plot.Distribution function, no internal checks are performed to ensure that the added plot makes sense in the context of the current plotting window. Therefore this function assumes that the current plot is of the same value support, see examples.

Author(s)

Chengyang Gao, Runlong Yu and Shuhan Liu

See Also

plot.Distribution for plotting a distr6 object.
Examples

plot(Normal$new(mean = 2), "pdf")
lines(Normal$new(mean = 3), "pdf", col = "red", lwd = 2)

## Not run:
# The code below gives examples of how not to use this function.
# Different value supports
plot(Binomial$new(), "cdf")
lines(Normal$new(), "cdf")

# Different functions
plot(Binomial$new(), "pdf")
lines(Binomial$new(), "cdf")

# Too many functions
plot(Binomial$new(), c("pdf","cdf"))
lines(Binomial$new(), "cdf")

## End(Not run)

listDecorators

Lists Implemented Distribution Decorators

Description

Lists decorators that can decorate an R6 Distribution.

Usage

listDecorators(simplify = TRUE)

Arguments

simplify logical. If TRUE (default) returns results as characters, otherwise as R6 classes.

Value

Either a list of characters (if simplify is TRUE) or a list of 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 = "extraDistr"))
listKernels

Lists Implemented Kernels

Description
Lists all implemented kernels in distr6.

Usage
listKernels(simplify = FALSE)

Arguments
simplify logical. If FALSE (default) returns kernels with support as a data.table, otherwise returns kernel names as characters.

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

See Also
Kernel

Examples
listKernels()

listWrappers

Lists Implemented Distribution Wrappers

Description
Lists wrappers that can wrap an R6 Distribution.

Usage
listWrappers(simplify = TRUE)

Arguments
simplify logical. If TRUE (default) returns results as characters, otherwise as R6 classes.

Value
Either a list of characters (if simplify is TRUE) or a list of Wrapper classes.
Logarithmic

Description

Mathematical and statistical functions for the Logarithmic distribution, which is commonly used to
to model consumer purchase habits in economics and is derived from the Maclaurin series expansion
of $-\ln(1 - p)$.

Details

The Logarithmic distribution parameterised with a parameter, $\theta$, is defined by the pmf,

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

for $0 < \theta < 1$.

The distribution is supported on 1, 2, 3, ....

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>theta</td>
<td>numeric</td>
<td>theta parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>
Logarithmic

Constructor Details

The Logarithmic distribution is parameterised with \( \theta \) as a number between 0 and 1.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

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

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

cf(t)
pgf(z)
median()
ilr()
mode(which = "all")

cf
pgf
median.Distribution
iqr
mode

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

parameters
getParameterValue
setParameterValue

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

liesInSupport
liesInType

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

strprint
print
summary.Distribution

References

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)
Logistic

$\text{cdf}(5)$
$\text{quantile}(0.42)$
$\text{rand}(4)$

# Statistics
$\text{mean()}$
$\text{variance()}$

summary(x)

---

Logistic Distribution Class

Description

Mathematical and statistical functions for the Logistic distribution, which is commonly used in logistic regression and feedforward neural networks.

Details

The Logistic distribution parameterised with mean, $\mu$, and scale, $s$, is defined by the pdf,

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

for $\mu \in \mathbb{R}$ and $s > 0$.

The distribution is supported on the Reals.

Value

Returns an R6 object inheriting from class SDistribution.

Constructor

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

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>numeric</td>
<td>location parameter.</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>scale parameter.</td>
</tr>
<tr>
<td>sd</td>
<td>numeric</td>
<td>standard deviation, alternate scale parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>
Constructor Details

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

\[ sd = scale \cdot \frac{\pi}{\sqrt{3}} \]

If sd is given then scale is ignored.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

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

Link

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

Statistical Methods

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

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

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

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

**References**

**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()
LogisticKernel

### Description

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

\[ f(x) = \frac{\exp(x) + 2 + \exp(-x)}{1} \]

over the support \( x \in \mathbb{R} \).

### Value

Returns an R6 object inheriting from class Kernel.

### Constructor

LogisticKernel$new(decorators = NULL)

#### Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality.</td>
</tr>
</tbody>
</table>

### Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>
Public Methods

Accessor Methods

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

d/p/q/r Methods

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

Statistical Methods

- squared2Norm()
- prec()
- stdev()
- mode(which = "all")
- mean()
- median()
- iqr()
- correlation()

Parameter Methods

- parameters(id)
- getParameterValue(id, error = "warn")
- setParameterValue(..., lst = NULL, error = "warn")

Link

- decorators
- traits
- valueSupport
- variateForm
- type
- properties
- support
- symmetry
- sup
- inf
- dmax
- dmin
- skewnessType
- kurtosisType
- pdf
- cdf
- quantile.Distribution
- rand
- squared2Norm
- prec
- stdev
- mode
- mean.Distribution
- median.Distribution
- iqr
- correlation
- parameters
- getParameterValue
- setParameterValue
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, $\beta$, scale, $\alpha$, and location, $\gamma$, is defined by the pdf,

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

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

The distribution is supported on the non-negative Reals.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape</td>
<td>numeric</td>
<td>shape parameter.</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>scale parameter.</td>
</tr>
<tr>
<td>location</td>
<td>numeric</td>
<td>location parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

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

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

<table>
<thead>
<tr>
<th>method</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
</tr>
</tbody>
</table>

Statistical Methods

<table>
<thead>
<tr>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
Loglogistic

pdf(x1, ..., log = FALSE, simplify = TRUE)
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
rand(n, simplify = TRUE)
mean()
variance()
stddev()
prec()
cor()
skewness()
kurtosis(excess = TRUE)
entropy(base = 2)
mgf(t)
cf(t)
pgf(z)
median()
iqr()
mode(which = "all")

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

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

References
**Lognormal**

**Log-Normal Distribution Class**

**Description**

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

**Details**

The Log-Normal distribution parameterised with logmean, \( \mu \), and logvar, \( \sigma \), is defined by the pdf,

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

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

The distribution is supported on the Positive Reals.

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

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

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

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

\[
sdlog = varlog^2
\]

\[
sd = var^2
\]

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods
Accessor Methods

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

Statistical Methods

pdf(x1, ..., log = FALSE, simplify = TRUE)
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
quartile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
rand(n, simplify = TRUE)
mean()
variance()
stddev()
prec()
cor()
skewness()
kurtosis(excess = TRUE)
entropy(base = 2)
mgf(t)
cf(t)
pgf(z)
median()
iqr()
mode(which = "all")

Parameter Methods

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

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

<table>
<thead>
<tr>
<th>Representation Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>strprint(n = 2)</td>
<td>strprint</td>
</tr>
<tr>
<td>print(n = 2)</td>
<td>print</td>
</tr>
<tr>
<td>summary(full = T)</td>
<td>summary.Distribution</td>
</tr>
</tbody>
</table>

References

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()
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()))

x$variance()

summary(x)
mean.Distribution  

Distribution Mean

Description

Arithmetic mean for the probability distribution.

Usage

```r
## 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

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

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

```r
$\text{mean}()$
```

See Also

`CoreStatistics`, `decorate` and `genExp`. 
**median.Distribution**  

**Median of a Distribution**

**Description**

Median of a distribution assuming quantile is provided.

**Usage**

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

**Arguments**

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

**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**

```r
$median()
```

**See Also**

- `quantile.Distribution`

---

**merge.ParameterSet**  

**Combine ParameterSets**

**Description**

merge dispatch method to combine parameter sets by rows.

**Usage**

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

- \( x \) ParameterSet
- \( y \) ParameterSet
- \( \ldots \) ParameterSets

Value

An R6 object of class ParameterSet.

R6 Usage

$merge(y, \ldots)$

See Also

ParameterSet

---

\textbf{mgf} \hspace{3cm} \textit{Moment Generating Function}

\begin{align*}
\text{Description} \\
\text{Moment generating function of a distribution}
\end{align*}

\begin{align*}
\text{Usage} \\
mgf(object, t)
\end{align*}

\textbf{Arguments}

- \( \text{object} \) Distribution.
- \( t \) integer to evaluate moment generating function at.

\textbf{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 \texttt{CoreStatistics} decorator.

\textbf{Value}

Moment generating function evaluated at \( t \) as a numeric.
MixtureDistribution

R6 Usage

$mgf(t)

See Also

CoreStatistics and decorate

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 $M(X_1, ..., X_N)$, $f_M$ 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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>distlist</td>
<td>list</td>
<td>List of distributions.</td>
</tr>
<tr>
<td>weights</td>
<td>numeric</td>
<td>Vector of weights. See Details.</td>
</tr>
<tr>
<td>vectordist</td>
<td>numeric</td>
<td>Vector Distribution. See Details.</td>
</tr>
</tbody>
</table>
MixtureDistribution

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

<table>
<thead>
<tr>
<th></th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>wrappedModels</td>
<td>wrappedModels</td>
</tr>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
</tr>
</tbody>
</table>

d/p/q/r Methods

<table>
<thead>
<tr>
<th></th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdf(x1, ..., log = FALSE, simplify = TRUE)</td>
<td>pdf</td>
</tr>
<tr>
<td>cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)</td>
<td>cdf</td>
</tr>
<tr>
<td>quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)</td>
<td>quantile.Distribution</td>
</tr>
<tr>
<td>rand(n, simplify = TRUE)</td>
<td>rand</td>
</tr>
</tbody>
</table>

Statistical Methods

<table>
<thead>
<tr>
<th></th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>prec()</td>
<td>prec</td>
</tr>
<tr>
<td>stdev()</td>
<td>stdev</td>
</tr>
<tr>
<td>median()</td>
<td>median.Distribution</td>
</tr>
<tr>
<td>iqr()</td>
<td>iqr</td>
</tr>
<tr>
<td>cor()</td>
<td>cor</td>
</tr>
</tbody>
</table>
Parameter Methods
parameters(id)
getParameterValue(id, error = "warn")
setParameterValue(..., lst = NULL, error = "warn")

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

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")
Multinomial 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,

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

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

The distribution is supported on $\sum x_i = N$.

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

Value

Returns an R6 object inheriting from class SDistribution.
Multinomial

Constructor

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

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>numeric</td>
<td>number of trials. See details.</td>
</tr>
<tr>
<td>probs</td>
<td>numeric</td>
<td>vector of probabilities. See details.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

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

$$probs = \frac{probs}{\sum probs}$$

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
</tbody>
</table>
Multinomial

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

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

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

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)
MultivariateNormal

References


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, \( \mu \), and covariance matrix, \( \Sigma \), is defined by the pdf,

\[
f(x_1, ..., x_k) = (2 \pi)^{-k/2} \frac{1}{\det(\Sigma)^{-1/2}} \exp(-1/2(x - \mu)^T \Sigma^{-1} (x - \mu))
\]

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

The distribution is supported on the Reals and only when the covariance matrix is positive-definite. skewness and kurtosis are omitted as no closed form analytic expression could be found, decorate with \texttt{CoreStatistics} for numerical results. cdf and quantile are omitted as no closed form analytic expression could be found, decorate with \texttt{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 \( \text{cov} \) and \( \text{prec} \) are internally coerced with \texttt{matrix(cov,nrow = K,byrow = FALSE)}.

Sampling is performed via the Cholesky decomposition using \texttt{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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>numeric</td>
<td>vector of means.</td>
</tr>
<tr>
<td>cov</td>
<td>numeric</td>
<td>vector or matrix. See details.</td>
</tr>
<tr>
<td>prec</td>
<td>numeric</td>
<td>vector or matrix. See details.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

### Public Methods

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

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

- \( pgf(z) \)
- \( \text{median()} \)
- \( \text{iqr()} \)
- \( \text{mode(} \text{which} = \text{"all"}) \)

### Parameter Methods

- \( \text{parameters(id)} \)
- \( \text{getParameterValue(id, error = "warn")} \)
- \( \text{setParameterValue(..., lst = NULL, error = "warn")} \)

### Validation Methods

- \( \text{liesInSupport(x, all = TRUE, bound = FALSE)} \)
- \( \text{liesInType(x, all = TRUE, bound = FALSE)} \)

### Representation Methods

- \( \text{strprint(n = 2)} \)
- \( \text{print(n = 2)} \)
- \( \text{summary(full = T)} \)

### References


### See Also

- \( \text{listDistributions} \) for all available distributions. \( \text{chol} \) for the implementation of the Cholesky decomposition. \( \text{Normal} \) for a special case of the Multivariate Normal distribution. \( \text{CoreStatistics} \) for numerical results. \( \text{FunctionImputation} \) to numerically impute \( d/p/q/r. \)

### Examples

```r
# 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,-1,1,0,-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)

---

**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 \( p \in [0, 1] \), 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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>numeric</td>
<td>number of failures/successes.</td>
</tr>
<tr>
<td>prob</td>
<td>numeric</td>
<td>probability of success.</td>
</tr>
<tr>
<td>qprob</td>
<td>numeric</td>
<td>probability of failure.</td>
</tr>
<tr>
<td>mean</td>
<td>numeric</td>
<td>location parameter.</td>
</tr>
<tr>
<td>form</td>
<td>character</td>
<td>form of negative binomial, see details.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
</tbody>
</table>
NegativeBinomial

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

Public Methods

Accessor Methods

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

Statistical Methods

pdf(x1, ..., log = FALSE, simplify = TRUE)
cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
rand(n, simplify = TRUE)
mean()
variance()
stdev()
prec()
cor()
skewness()
kurtosis(excess = TRUE)
entropy(base = 2)
mgf(t)
cf(t)
pgf(z)
mixedDist()
mode(which = "all")

Link

decorators
traits
valueSupport
variateForm
type
properties
support
symmetry
sup
inf
dmax
dmin
skewnessType
kurtosisType
pdf
cdf
quantile.Distribution
rand
mean.Distribution
variance
stdev
prec
cor
skewness
kurtosis
entropy
mgf
cf
pgf
median.Distribution
iqr
mode
### Parameter Methods

- `parameters(id)`
- `getParameterValue(id, error = "warn")`
- `setParameterValue(..., lst = NULL, error = "warn")`

### Validation Methods

- `liesInSupport(x, all = TRUE, bound = FALSE)`
- `liesInType(x, all = TRUE, bound = FALSE)`

### Representation Methods

- `strprint(n = 2)`
- `print(n = 2)`
- `summary(full = T)`

### References


### See Also

- `listDistributions` for all available distributions. **Binomial** for the Binomial distribution and **Geometric** for the Geometric distribution.

### Examples

```r
# 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!
```
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, $\sigma^2$, and mean, $\mu$, is defined by the pdf,

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

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

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>numeric</td>
<td>mean, location parameter.</td>
</tr>
<tr>
<td>var</td>
<td>numeric</td>
<td>variance, squared scale parameter.</td>
</tr>
<tr>
<td>sd</td>
<td>numeric</td>
<td>standard deviation, scale parameter.</td>
</tr>
<tr>
<td>prec</td>
<td>numeric</td>
<td>precision, inverse squared scale parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>
Constructor Details

The Normal distribution is parameterised with mean as a numeric, and either var, sd or prec as numerics. These are related via,

\[
sd = \sqrt{\text{var}}
\]

\[
prec = 1/\text{var}
\]

If prec is given then sd and var are ignored. If sd is given then var is ignored.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

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

Statistical Methods

- pdf(x1, ..., log = FALSE, simplify = TRUE)
- cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- rand(n, simplify = TRUE)
- mean()
- variance()
Normal

\begin{itemize}
\item stdev()
\item prec()
\item cor()
\item skewness()
\item kurtosis(excess = TRUE)
\item entropy(base = 2)
\item mgf(t)
\item cf(t)
\item pgf(z)
\item median()
\item iqr()
\item mode(which = "all")
\end{itemize}

**Parameter Methods**
- parameters(id)
- getParameterValue(id, error = "warn")
- setParameterValue(\ldots, lst = NULL, error = "warn")

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

**Representation Methods**
- strprint(n = 2)
- print(n = 2)
- summary(full = T)

**References**

**See Also**
- listDistributions for all available distributions.

**Examples**
\begin{verbatim}
# Different parameterisations
Normal$new(var = 1, mean = 1)
\end{verbatim}
Normal$new(prec = 2, mean = 1)
Normal$new(mean = 1, sd = 2)

x <- Normal$new(verbode = 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,

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

over the support \( x \in \mathbb{R} \).

**Details**

We use the `erf` and `erfinv` error and inverse error functions from the Pracma package.

**Value**

Returns an R6 object inheriting from class Kernel.

**Constructor**

NormalKernel$new(decorators = NULL)

**Constructor Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality.</td>
</tr>
</tbody>
</table>
Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

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

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

**Statistical Methods**
- squared2Norm()
- prec()
- stdev()
- mode(which = "all")
- mean()
- median()
- iqr()
parameters

```r
parameters(id)
getParameterValue(id, error = "warn")
setParameterValue(..., lst = NULL, error = "warn")
```

### Validation Methods

```r
liesInSupport(x, all = TRUE, bound = FALSE)
liesInType(x, all = TRUE, bound = FALSE)
```

### Representation Methods

```r
strprint(n = 2)
print(n = 2)
summary(full = T)
```

---

**parameters**  
*Parameters Accessor*

---

**Description**

Returns some or all the parameters in a distribution.

**Usage**

```r
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.
ParameterSet

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>character</td>
<td>unique one-word identifier.</td>
</tr>
<tr>
<td>value</td>
<td>numeric</td>
<td>initial parameter value.</td>
</tr>
<tr>
<td>support</td>
<td>numeric</td>
<td>range of values parameter can take.</td>
</tr>
<tr>
<td>settable</td>
<td>logical</td>
<td>if TRUE the parameter is printed. See Details.</td>
</tr>
<tr>
<td>updateFunc = NULL</td>
<td>function</td>
<td>evaluated to update parameter. See Details.</td>
</tr>
<tr>
<td>description = NULL</td>
<td>character</td>
<td>optional description of parameter.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>print(hide_cols = c(&quot;updateFunc&quot;,&quot;settable&quot;))</td>
<td>print.ParameterSet</td>
</tr>
<tr>
<td>update()</td>
<td>update.ParameterSet</td>
</tr>
<tr>
<td>parameters(id = NULL)</td>
<td>parameters</td>
</tr>
<tr>
<td>getParameterSupport(id, error = &quot;warn&quot;)</td>
<td>getParameterSupport</td>
</tr>
<tr>
<td>getParameterValue(id, error = &quot;warn&quot;)</td>
<td>getParameterValue</td>
</tr>
<tr>
<td>setParameterValue(..., lst = NULL, error = &quot;warn&quot;)</td>
<td>setParameterValue</td>
</tr>
<tr>
<td>merge(y, ...)</td>
<td>merge.ParameterSet</td>
</tr>
<tr>
<td>as.data.table()</td>
<td>as.data.table</td>
</tr>
</tbody>
</table>

### See Also

**Distribution**

### Examples

```r
id = list("prob", "size")
value = list(0.2, 5)
support = list(set6::Interval$new(0,1), set6::PosNaturals$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")
```

```r
id = list("rate", "scale")
value = list(1, 1)
support = list(set6::PosReals$new(), set6::PosReals$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 Distribution Class**

**Description**

Mathematical and statistical functions for the Pareto distribution, which is commonly used in Economics to model the distribution of wealth and the 80-20 rule.

**Details**

The Pareto distribution parameterised with shape, $\alpha$, and scale, $\beta$, is defined by the pdf,

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

for $\alpha, \beta > 0$.

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

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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape</td>
<td>numeric</td>
<td>shape parameter.</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>scale parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

**Constructor Details**

The Pareto distribution is parameterised with shape and scale as positive numerics.
Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

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

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

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

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

**Representation Methods**
- strprint(n = 2)
- print(n = 2)
- summary(full = T)

**References**

**See Also**
- `listDistributions` for all available distributions. `CoreStatistics` for numerical results.

**Examples**
```r
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()
```
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.
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

\[
\left( \int_a^b |f_X|^p \, d\mu \right)^{1/p}
\]

where X is the distribution, \( f_X \) is the pdf and \( a, b \) are the 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**
**Description**

Probability generating function of a distribution

**Usage**

\[
\text{pgf}(\text{object}, z)
\]

**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**

\$\text{spgf}(z)\$

**See Also**

`CoreStatistics` and `decorate`
Description

Six plots, which can be selected with `fun` are available for discrete and continuous univariate distributions: pdf, cdf, quantile, survival, hazard and cumulative hazard. By default, the first two are plotted side by side.

Usage

```r
## S3 method for class 'Distribution'
plot(
  x,
  fun = c("pdf", "cdf"),
  npoints = 3000,
  plot = TRUE,
  ask = FALSE,
  arrange = TRUE,
  ...
)
```

Arguments

- `x`: distr6 object.
- `fun`: vector of functions to plot, one or more of: "pdf", "cdf", "quantile", "survival", "hazard", "cumhazard", and "all"; partial matching available.
- `npoints`: number of evaluation points.
- `plot`: logical; if TRUE (default), figures are displayed in the plot window; otherwise a data.table of points and calculated values is returned.
- `ask`: logical; if TRUE, the user is asked before each plot, see `par`.
- `arrange`: logical; if TRUE (default), margins are automatically adjusted with `layout` to accommodate all plotted functions.
- `...`: graphical parameters, see details.

Details

The evaluation points are calculated using inverse transform on a uniform grid between 0 and 1 with length given by `npoints`. Therefore any distribution without an analytical quantile method will first need to be imputed with the `FunctionImputation` decorator.

The order that the functions are supplied to `fun` determines the order in which they are plotted, however this is ignored if `ask` is TRUE. If `ask` is TRUE then `arrange` is ignored. For maximum flexibility in plotting layouts, set `arrange` and `ask` to FALSE.

The graphical parameters passed to `...` can either apply to all plots or selected plots. If parameters in `par` are prefixed with the plotted function name, then the parameter only applies to that function, otherwise it applies to them all. See examples for a clearer description.
Author(s)
Chengyang Gao, Runlong Yu and Shuhan Liu

See Also
lines.Distribution for superimposing a distr6 object and listDistributions for plottable distributions.

Examples

# Plot pdf and cdf of Normal
plot(Normal$new())

# Colour both plots red
plot(Normal$new(), col = "red")

# Change the colours of individual plotted functions
plot(Normal$new(), pdf_col = "red", cdf_col = "green")

# Interactive plotting in order - par still works here
plot(Geometric$new(), fun = "all", ask = TRUE, pdf_col = "black",
cdf_col = "red", quantile_col = "blue", survival_col = "purple",
hazard_col = "brown", cumhazard_col = "yellow")

# Return plotting structure
x = plot(Gamma$new(), plot = FALSE)

Poisson Distribution Class

Description
Mathematical and statistical functions for the Poisson distribution, which is commonly used to model the number of events occurring in at a constant, independent rate over an interval of time or space.

Details
The Poisson distribution parameterised with arrival rate, \( \lambda \), is defined by the pmf,

\[
f(x) = \frac{(\lambda^x \cdot e^{-\lambda})}{x!}
\]

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate</td>
<td>numeric</td>
<td>arrival rate.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

The Poisson distribution is parameterised with rate as a positive numeric.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

<table>
<thead>
<tr>
<th>Link</th>
<th>Accessor Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>decorators()</td>
</tr>
<tr>
<td>traits</td>
<td>traits()</td>
</tr>
<tr>
<td>valueSupport</td>
<td>valueSupport()</td>
</tr>
<tr>
<td>variateForm</td>
<td>variateForm()</td>
</tr>
<tr>
<td>type</td>
<td>type()</td>
</tr>
<tr>
<td>properties</td>
<td>properties()</td>
</tr>
<tr>
<td>support</td>
<td>support()</td>
</tr>
<tr>
<td>symmetry</td>
<td>symmetry()</td>
</tr>
<tr>
<td>sup</td>
<td>sup()</td>
</tr>
<tr>
<td>inf</td>
<td>inf()</td>
</tr>
<tr>
<td>dmax</td>
<td>dmax()</td>
</tr>
<tr>
<td>dmin</td>
<td>dmin()</td>
</tr>
<tr>
<td>skewnessType</td>
<td>skewnessType()</td>
</tr>
<tr>
<td>kurtosisType</td>
<td>kurtosisType()</td>
</tr>
</tbody>
</table>
Statistical Methods

- PDF: `pdf(x1, ..., log = FALSE, simplify = TRUE)`
- CDF: `cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`
- Quantile: `quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`
- Random: `rand(n, simplify = TRUE)`
- Mean: `mean()`
- Variance: `variance()`
- Standard Deviation: `stdev()`
- Precision: `prec()`
- Correlation: `cor()`
- Skewness: `skewness()`
- Kurtosis: `kurtosis(excess = TRUE)`
- Entropy: `entropy(base = 2)`
- Moment Generating Function: `mgf(t)`
- Characteristic Function: `cf(t)`
- Probability Generating Function: `pgf(z)`
- Median: `median()`
- Interquartile Range: `iqr()`
- Mode: `mode(which = "all")`

Parameter Methods

- Parameters: `parameters(id)`
- Get Parameter Value: `getParameterValue(id, error = "warn")`
- Set Parameter Value: `setParameterValue(..., lst = NULL, error = "warn")`

Validation Methods

- Lies in Support: `liesInSupport(x, all = TRUE, bound = FALSE)`
- Lies in Type: `liesInType(x, all = TRUE, bound = FALSE)`

Representation Methods

- Str Print: `strprint(n = 2)`
- Print: `print(n = 2)`
- Summary: `summary(full = T)`
References


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)

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

preci()

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

preci(hide_cols = c("updateFunc","settable"))

See Also

ParameterSet
ProductDistribution

**Description**

A wrapper for creating the joint distribution of multiple independent probability distributions.

**Details**

Exploits the following relationships of independent distributions

\[ f_P(X_1 = x_1, \ldots, X_N = x_N) = f_{X_1}(x_1) \ast \ldots \ast f_{X_N}(x_N) \]

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

where \( f_P/F_P \) is the pdf/cdf of the joint (product) distribution \( P \) and \( X_1, \ldots, X_N \) 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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>distlist</td>
<td>list</td>
<td>List of distributions.</td>
</tr>
<tr>
<td>distribution</td>
<td>distribution</td>
<td>Distribution to wrap.</td>
</tr>
<tr>
<td>params</td>
<td>a R object</td>
<td>Either list of parameters or matrix-type frame, see examples.</td>
</tr>
<tr>
<td>name</td>
<td>list</td>
<td>Optional new name for distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>list</td>
<td>Optional new short_name for distribution.</td>
</tr>
<tr>
<td>description</td>
<td>list</td>
<td>Optional new description for distribution.</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

### Accessor Methods

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

### d/p/q/r Methods

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

### Statistical Methods

- prec()
- stdev()
- median()
- iqr()
- cor()
Parameter Methods
parameters(id)
getParameterValue(id, error = "warn")
setParameterValue(..., lst = NULL, error = "warn")

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

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()

---

qqplot  

**Quantile-Quantile Plots for distr6 Objects**

**Description**
Quantile-quantile plots are used to compare a "theoretical" or empirical distribution to a reference distribution. They can also compare the quantiles of two reference distributions.

**Usage**
qqplot(x, y, npoints = 3000, idline = TRUE, plot = TRUE, ...)

**Arguments**
- x: distr6 object or numeric vector.
- y: distr6 object or numeric vector.
- npoints: number of evaluation points.
- idline: logical; if TRUE (default), the line $y = x$ is plotted
- plot: logical; if TRUE (default), figures are displayed in the plot window; otherwise a data.table of points and calculated values is returned.
- ...: graphical parameters.
Details

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

Author(s)

Chijing Zeng

See Also

plot.Distribution for plotting a distr6 object.

Examples

```r
qqplot(Normal$new(mean = 15, sd = sqrt(30)), ChiSquared$new(df = 15))
qqplot(rt(200, df = 5), rt(300, df = 5), main = QQ-Plot, xlab = t-200,
    ylab = t-300)
qqplot(Normal$new(mean = 2), rnorm(100, mean = 3))
```

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

```r
## 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 \( q_X(e^{xp}) \) 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.

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

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

```r
$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 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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality.</td>
</tr>
</tbody>
</table>

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
<td>pdf</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
<td>cdf</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
<td>quantile.Distribution</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
<td></td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
<td></td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
<td></td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
<td></td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
<td></td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
<td></td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
<td></td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
<td></td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
<td></td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
<td></td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
<td></td>
</tr>
</tbody>
</table>

d/p/q/r Methods

<table>
<thead>
<tr>
<th>Link</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdf(x1, ..., log = FALSE, simplify = TRUE)</td>
<td>pdf</td>
</tr>
<tr>
<td>cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)</td>
<td>cdf</td>
</tr>
<tr>
<td>quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)</td>
<td>quantile.Distribution</td>
</tr>
<tr>
<td>rand(n, simplify = TRUE)</td>
<td>rand</td>
</tr>
</tbody>
</table>
**Statistical Methods**
- `squared2Norm()`
- `prec()`
- `stdev()`
- `mode(which = "all")`
- `mean()`
- `median()`
- `iqr()`
- `correlation()`

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

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

**Representation Methods**
- `strprint(n = 2)`
- `print(n = 2)`
- `summary(full = T)`

---

**rand**

*Random Simulation Function*

**Description**
Returns a given number of points sampled from the distribution.

**Usage**

```r
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.

Rayleigh Distribution Class

Description

Mathematical and statistical functions for the Rayleigh distribution, which is commonly used to model random complex numbers.

Details

The Rayleigh distribution parameterised with mode (or scale), $\alpha$, is defined by the pdf,

$$f(x) = \frac{x}{\alpha^2} e^{-x^2/(2\alpha^2)}$$

for $\alpha > 0$.

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

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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>numeric</td>
<td>mode, scale parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

**Constructor Details**

The Rayleigh distribution is parameterised with mode as a non-negative numeric.

**Public Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

**Public Methods**

**Accessor Methods**

<table>
<thead>
<tr>
<th>Accessor Method</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
</tr>
</tbody>
</table>
## Statistical Methods

- pdf($x_1, \ldots$, log = FALSE, simplify = TRUE)
- cdf($x_1, \ldots$, lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- quantile($p, \ldots$, lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- rand($n$, simplify = TRUE)
- mean()
- variance()
- stdev()
- prec()
- cor()
- skewness()
- kurtosis(excess = TRUE)
- entropy(base = 2)
- mgf($t$)
- cf($t$)
- pgf($z$)
- median()
- iqr()
- mode(which = "all")

## Parameter Methods

- parameters(id)
- getParameterValue(id, error = "warn")
- setParameterValue(..., lst = NULL, error = "warn")

## Validation Methods

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

## Representation Methods

- strprint($n = 2$)
- print($n = 2$)
- summary(full = T)

---

**Link**

pdf

cdf

quantile.Distribution

rand

mean.Distribution

variance

stdev

prec

cor

skewness

kurtosis

entropy

mgf

cf

pgf

median.Distribution

iqr

mode

parameters

getParameterValue

setParameterValue

liesInSupport

liesInType

strprint

print

summary.Distribution
References


See Also

listDistributions for all available distributions. CoreStatistics for numerical results.

Examples

```r
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)
```

---

SDistribution

Abstract Special Distribution Class

Description

Abstract class that cannot be constructed directly.

Value

Returns error. Abstract classes cannot be constructed directly.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>
Public Methods

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

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

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

**Link**
- decorators
decorators
- traits
- valueSupport
- variateForm
type
- properties
- support
- symmetry
- sup
- inf
dmax
- dmin
- skewnessType
- kurtosisType
- pdf
- cdf
- quantile.Distribution
- rand
- mean.Distribution
- variance
- stdev
- prec
cor
- skewness
- kurtosis
- entropy
- mgf
cf
- pgf
- median.Distribution
- iqr
- mode
- parameters
- getParameterValue
- setParameterValue
**setParameterValue**

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.

---

**Validation Methods**

- `liesInSupport(x, all = TRUE, bound = FALSE)`
- `liesInType(x, all = TRUE, bound = FALSE)`

**Representation Methods**

- `strprint(n = 2)`
- `print(n = 2)`
- `summary(full = T)`

**Active Bindings**

<table>
<thead>
<tr>
<th>Active Binding</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isPdf</code></td>
<td><code>isPdf</code></td>
</tr>
<tr>
<td><code>isCdf</code></td>
<td><code>isCdf</code></td>
</tr>
<tr>
<td><code>isQuantile</code></td>
<td><code>isQuantile</code></td>
</tr>
<tr>
<td><code>isRand</code></td>
<td><code>isRand</code></td>
</tr>
</tbody>
</table>

**Description**

Sets the value of the given parameter.
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

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

See Also

`parameters` and `setParameterValue`

Examples

```r
ps <- Normal$new()$parameters()
ps$setParameterValue(mean = 2, var = 5)$print()

ps <- MultivariateNormal$new()$parameters()
ps$setParameterValue(lst = list(mean = c(1,1)))$print()
```

---

**Sigmoid**

**Sigmoid Kernel**

Description

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

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

over the support \( x \in \mathbb{R} \).

Details

The cdf and quantile functions are omitted as no closed form analytic expressions could be found, decorate with `FunctionImputation` for numeric results.

Value

Returns an R6 object inheriting from class Kernel.
Constructor

Sigmoid$new(decorators = NULL)

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality.</td>
</tr>
</tbody>
</table>

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

**Accessor Methods**

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

**Link**

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

**d/p/q/r Methods**

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

**Link**

- pdf
- cdf
- quantile.Distribution
- rand
Silverman

Silverman Kernel

Description

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

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

over the support \( x \in \mathbb{R} \).
Details

The cdf and quantile functions are omitted as no closed form analytic expressions could be found, decorate with FunctionImputation for numeric results.

Value

Returns an R6 object inheriting from class Kernel.

Constructor

Silverman$new(decorators = NULL)

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality.</td>
</tr>
</tbody>
</table>

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

<table>
<thead>
<tr>
<th>Accessor Methods</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
</tr>
</tbody>
</table>
**d/p/q/r Methods**
- pdf(x1, ..., log = FALSE, simplify = TRUE)
- cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- rand(n, simplify = TRUE)

**Statistical Methods**
- squared2Norm()
- prec()
- stdev()
- mode(which = "all")
- mean()
- median()
- iqr()
- correlation()

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

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

**Representation Methods**
- strprint(n = 2)
- print(n = 2)
- summary(full = T)

---

**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**

```r
simulateEmpiricalDistribution(EmpiricalDist, n, seed = NULL)
```

**Arguments**

- `EmpiricalDist`: Empirical Distribution
- `n`: Number of samples to generate. See Details.
- `seed`: Numeric passed to `set.seed`. See Details.

**Details**

This function can only be used to sample from the `Empirical` distribution without replacement, and will return an error for other distributions.

The `seed` param ensures that the same samples can be reproduced and is more convenient than using the `set.seed` function each time before use. If `set.seed` is `NULL` then the seed is left unchanged (NULL is not passed to the `set.seed` function).

If `n` is of length greater than one, then `n` is taken to be the length of `n`. If `n` is greater than the number of observations in the Empirical distribution, then `n` is taken to be the number of observations in the distribution.

**Value**

A vector of length `n` with elements drawn without replacement from the given Empirical distribution.

**See Also**

`set.seed`, `rand`, and `Empirical`

---

**skewness**

*Distribution Skewness*

**Description**

Skewness of a distribution

**Usage**

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

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

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

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

$\text{skewness}()$

See Also

CoreStatistics and decorate

<table>
<thead>
<tr>
<th>skewnessType</th>
<th>Type of Skewness Accessor</th>
</tr>
</thead>
</table>

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

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)
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

$\text{squared2Norm}(lower = \text{NULL}, upper = \text{NULL})$

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

```r
Triangular$new()$strprint()
Triangular$new()$strprint(1)
```

---

### StudentT

#### Student's T Distribution Class

**Description**

Mathematical and statistical functions for the Student’s T distribution, which is commonly used to estimate the mean of populations with unknown variance from a small sample size, as well as in t-testing for difference of means and regression analysis.

**Details**

The Student’s T distribution parameterised with degrees of freedom, $\nu$, is defined by the pdf,

$$
  f(x) = \frac{\Gamma((\nu + 1)/2)/((\nu \pi)^{1/2}\Gamma(\nu/2)) \times \left(1 + \frac{x^2}{\nu}\right)^\left(-\frac{\nu + 1}{2}\right)}
$$

for $\nu > 0$.

The distribution is supported on the Reals.

**Value**

Returns an R6 object inheriting from class SDistribution.

**Constructor**

```r
StudentT$new(df = 1, decorators = NULL, verbose = FALSE)
```

**Constructor Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>df</code></td>
<td>numeric</td>
<td>degrees of freedom.</td>
</tr>
<tr>
<td><code>decorators</code></td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td><code>verbose</code></td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>
Constructor Details

The Student's T distribution is parameterised with \( df \) as a positive numeric.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

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

**Statistical Methods**
- pdf(x1, ..., log = FALSE, simplify = TRUE)
- cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- rand(n, simplify = TRUE)
- mean()
- variance()
- stdev()
- prec()
- cor()
- skewness()
- kurtosis(excess = TRUE)
StudentT

entropy(base = 2)
mgf(t)
cf(t)
pgf(z)
median()
iqr()
mode(which = "all")

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

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

Author(s)
Chijing Zeng

References

See Also
listDistributions for all available distributions. Normal for the Normal distribution, StudentTNoncentral for the noncentral Student’s T 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)

---

**StudentTNoncentral**  
*Noncentral Student’s T Distribution Class*

---

**Description**
Mathematical and statistical functions for the Noncentral Student’s T distribution, which is commonly used to estimate the mean of populations with unknown variance from a small sample size, as well as in t-testing for difference of means and regression analysis.

**Details**
The Noncentral Student’s T distribution parameterised with degrees of freedom, $\nu$ and location, $\lambda$, is defined by the pdf,

$$f(x) = \frac{(\nu^{\nu/2} \exp\left(-\nu\lambda^2\right)/\left(2(x^2+\nu)\right)/\left(\sqrt{\pi}\Gamma\left(\nu/2\right)\right)^{(\nu-1)/2}\left(x^2+\nu\right)^{(\nu+1)/2}}{\int_0^\infty y^{\nu/2} \exp\left(-1/2 \left(y-x\lambda/\sqrt{x^2+\nu}\right)^2\right) dy}$$

for $\nu > 0$, $\lambda \in \mathbb{R}$.

The distribution is supported on the Reals.

skewness, kurtosis, mode, entropy, pgf, 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**
StudentTNoncentral$new(df = 1, location = 0, decorators = NULL, verbose = FALSE)

**Constructor Arguments**
**Argument**
- df: numeric degrees of freedom.
- location: numeric location (ncp in rstats).
- decorators: Decorator decorators to add functionality. See details.
- verbose: logical if TRUE parameterisation messages produced.

**Constructor Details**

The Noncentral Student’s T distribution is parameterised with df as positive numeric and location as real numeric.

**Public Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

**Public Methods**

**Accessor Methods**

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

**Statistical Methods**

- pdf(x1, ..., log = FALSE, simplify = TRUE)
- cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- rand(n, simplify = TRUE)
mean()  
variance()  
stddev()  
prec()  
cor()  
skewness()  
kurtosis(excess = TRUE)  
entropy(base = 2)  
gmf(t)  
cf(t)  
pgf(z)  
median()  
iqr()  
mode(which = "all")

\textbf{Parameter Methods}  
parameters(id)  
getParameterValue(id, error = "warn")  
setParameterValue(..., lst = NULL, error = "warn")

\textbf{Validation Methods}  
liesInSupport(x, all = TRUE, bound = FALSE)  
liesInType(x, all = TRUE, bound = FALSE)

\textbf{Representation Methods}  
strprint(n = 2)  
print(n = 2)  
summary(full = T)

\textbf{Author(s)}  
Jordan Deenichin

\textbf{References}  
See Also

listDistributions for all available distributions. Normal for the Normal distribution, StudentT for the central Student’s T distribution. CoreStatistics for numerical results.

Examples

x = StudentTNoncentral$new(df = 2, location = 3)

# 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)
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
**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,

\[
\text{Supp}(X) = \{ x \in \mathbb{R} : f_X(x) > 0 \}
\]

where \( f_X \) is the pmf if distribution \( X \) is discrete, otherwise the pdf.

**Value**

An R6 object of class `set6::Set`.

**R6 Usage**

$support()$

**See Also**

`set6::Set` 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

```r
$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

```r
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

\[ \text{as}(a, b) = \int_a^b S_X(x) \, dx \]

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 \texttt{ExoticStatistics}.

Value

Antiderivative of the survival function evaluated between limits as a numeric.

R6 Usage

\$\text{survivalAntiDeriv}(lower = \text{NULL}, \text{upper} = \text{NULL})$

See Also

\texttt{ExoticStatistics} and \texttt{decorate}

---

\texttt{survivalPNorm} \hspace{1cm} \textit{Survival Function P-Norm}

Description

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

Usage

\texttt{survivalPNorm(object, p = 2, lower = \text{NULL}, upper = \text{NULL})}

Arguments

\begin{itemize}
\item \texttt{object} \hspace{1cm} Distribution.
\item \texttt{p} \hspace{1cm} p-norm to calculate.
\item \texttt{lower} \hspace{1cm} lower limit for integration, default is infimum.
\item \texttt{upper} \hspace{1cm} upper limit for integration, default is supremum.
\end{itemize}

Details

The p-norm of the survival function is defined by

\[ \left( \int_a^b |S_X|^p \, d\mu \right)^{1/p} \]

where \( X \) is the distribution, \( S_X \) is the survival function and \( a, b \) are the limits of integration.

Returns NULL if distribution is not continuous.

Can only be used after decorating with \texttt{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
Description

Validation checks to test if Distribution is continuous.

Usage

testContinuous(
  object,
  errmsg = paste(object$short_name, "is not continuous")
)

cmpCheckContinuous(
  object,
  errmsg = paste(object$short_name, "is not continuous")
)

assertContinuous(
  object,
  errmsg = paste(object$short_name, "is not continuous")
)

Arguments

object Distribution
errmsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testContinuous(Binomial$new()) # FALSE
### testDiscrete

**Description**
Validation checks to test if Distribution is discrete.

**Usage**

```r
testDiscrete(object, errormsg = paste(object$short_name, "is not discrete"))
checkDiscrete(object, errormsg = paste(object$short_name, "is not discrete"))
assertDiscrete(object, errormsg = paste(object$short_name, "is not discrete"))
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Distribution</td>
</tr>
<tr>
<td>errormsg</td>
<td>custom error message to return if assert/check fails</td>
</tr>
</tbody>
</table>

**Value**
If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

**Examples**

```r
testDiscrete(Binomial$new()) # FALSE
```

---

### testDistribution

**Description**
Validation checks to test if a given object is an R6 Distribution.

**Usage**

```r
testDistribution(
  object,
  errormsg = paste(object, "is not an R6 Distribution object")
)
checkDistribution(
  object,
```
assertDistribution(
  object,
  errmsg = paste(object, "is not an R6 Distribution object")
)

Arguments

object          object to test
errmsg          custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

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,
  errmsg = "One or more items in the list are not Distributions"
)

checkDistributionList(
  object,
  errmsg = "One or more items in the list are not Distributions"
)

assertDistributionList(
  object,
  errmsg = "One or more items in the list are not Distributions"
)
Arguments

object  object to test
errmsg  custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testDistributionList(list(Binomial$new(),5)) # FALSE  
testDistributionList(list(Binomial$new(),Exponential$new())) # TRUE

description

Validation checks to test if Distribution is leptokurtic.

Usage

testLeptokurtic(
  object,
  errmsg = paste(object$short_name, "is not leptokurtic")
)
checkLeptokurtic(
  object,
  errmsg = paste(object$short_name, "is not leptokurtic")
)
assertLeptokurtic(
  object,
  errmsg = paste(object$short_name, "is not leptokurtic")
)

Arguments

object  Distribution
errmsg  custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.
Examples

testLeptokurtic(Binomial$new())

description

Validation checks to test if Distribution is matrixvariate.

Usage

testMatrixvariate(
  object,
  errormsg = paste(object$short_name, "is not matrixvariate")
)

checkMatrixvariate(
  object,
  errormsg = paste(object$short_name, "is not matrixvariate")
)

assertMatrixvariate(
  object,
  errormsg = paste(object$short_name, "is not matrixvariate")
)

Arguments

object Distribution

errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testMatrixvariate(Binomial$new()) # FALSE
Description

Validation checks to test if Distribution is mesokurtic.

Usage

testMesokurtic(
  object,
  errormsg = paste(object$short_name, "is not mesokurtic")
)

checkMesokurtic(
  object,
  errormsg = paste(object$short_name, "is not mesokurtic")
)

assertMesokurtic(
  object,
  errormsg = paste(object$short_name, "is not mesokurtic")
)

Arguments

object Distribution
errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testMesokurtic(Binomial$new())
testMixture

Description
Validation checks to test if Distribution is mixture.

Usage
```r
testMixture(object, errormsg = paste(object$short_name, "is not mixture"))
checkMixture(object, errormsg = paste(object$short_name, "is not mixture"))
assertMixture(object, errormsg = paste(object$short_name, "is not mixture"))
```

Arguments
- `object`: Distribution
- `errormsg`: custom error message to return if assert/check fails

Value
If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples
```r
testMixture(Binomial$new()) # FALSE
```

testMultivariate

Description
Validation checks to test if Distribution is multivariate.

Usage
```r
testMultivariate(
    object,
    errmsg = paste(object$short_name, "is not multivariate")
)
checkMultivariate(
    object,
```
errmsg = paste(object$short_name, "is not multivariate")

assertMultivariate(
    object,
    errmsg = paste(object$short_name, "is not multivariate")
)

**Arguments**

- `object` Distribution
- `errmsg` custom error message to return if assert/check fails

**Value**

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

**Examples**

testMultivariate(Binomial$new()) # FALSE

testNegativeSkew

**Description**

Validation checks to test if Distribution is negative skew.

**Usage**

testNegativeSkew(
    object,
    errmsg = paste(object$short_name, "is not negative skew")
)

checkNegativeSkew(
    object,
    errmsg = paste(object$short_name, "is not negative skew")
)

assertNegativeSkew(
    object,
    errmsg = paste(object$short_name, "is not negative skew")
)
testNoSkew

Arguments

- `object`: Distribution
- `errmsg`: custom error message to return if assert/check fails

Value

If check passes then `assert` returns invisibly and `test/check` return `TRUE`. If check fails, `assert` stops code with error, `check` returns an error message as string, `test` returns `FALSE`.

Examples

```r
testNegativeSkew(Binomial$new())
```

---

testNoSkew `assert/check/test/NoSkew`

Description

Validation checks to test if Distribution is no skew.

Usage

```r
testNoSkew(object, errmsg = paste(object$short_name, "is not no skew"))
checkNoSkew(object, errmsg = paste(object$short_name, "is not no skew"))
assertNoSkew(object, errmsg = paste(object$short_name, "is not no skew"))
```

Arguments

- `object`: Distribution
- `errmsg`: custom error message to return if assert/check fails

Value

If check passes then `assert` returns invisibly and `test/check` return `TRUE`. If check fails, `assert` stops code with error, `check` returns an error message as string, `test` returns `FALSE`.

Examples

```r
testNoSkew(Binomial$new())
```
Description

Validation checks to test if Distribution is platykurtic.

Usage

testPlatykurtic(
  object,
  errmsg = paste(object$short_name, "is not platykurtic")
)

cHECKPlatykurtic(
  object,
  errmsg = paste(object$short_name, "is not platykurtic")
)

assertPlatykurtic(
  object,
  errmsg = paste(object$short_name, "is not platykurtic")
)

Arguments

  object                  Distribution
  errmsg                  custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testPlatykurtic(Binomial$new())
Description

Validation checks to test if Distribution is positive skew.

Usage

testPositiveSkew(
    object,
    errormsg = paste(object$short_name, "is not positive skew")
)

checkPositiveSkew(
    object,
    errormsg = paste(object$short_name, "is not positive skew")
)

assertPositiveSkew(
    object,
    errormsg = paste(object$short_name, "is not positive skew")
)

Arguments

object    Distribution
errormsg  custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testPositiveSkew(Binomial$new())
Description

Validation checks to test if Distribution is symmetric.

Usage

testSymmetric(object, errormsg = paste(object$short_name, "is not symmetric"))

checkSymmetric(object, errormsg = paste(object$short_name, "is not symmetric"))

assertSymmetric(
    object,
    errormsg = paste(object$short_name, "is not symmetric")
)

Arguments

object Distribution
errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testSymmetric(Binomial$new()) # FALSE

testUnivariate  assert/check/test/Univariate

Description

Validation checks to test if Distribution is univariate.
traits

Usage

testUnivariate(
  object,
  errormsg = paste(object$short_name, "is not univariate")
)

checkUnivariate(
  object,
  errormsg = paste(object$short_name, "is not univariate")
)

assertUnivariate(
  object,
  errormsg = paste(object$short_name, "is not univariate")
)

Arguments

object Distribution
errormsg custom error message to return if assert/check fails

Value

If check passes then assert returns invisibly and test/check return TRUE. If check fails, assert stops code with error, check returns an error message as string, test returns FALSE.

Examples

testUnivariate(Binomial$new()) # TRUE

traits

Traits Accessor

Description

Returns the traits of the distribution.

Usage

traits(object)

Arguments

object Distribution.
**Triangular**

**Value**

List of traits.

**R6 Usage**

`$traits()`

---

**Triangular Distribution Class**

**Description**

Mathematical and statistical functions for the Triangular distribution, which is commonly used to model population data where only the minimum, mode and maximum are known (or can be reliably estimated), also to model the sum of standard uniform distributions.

**Details**

The Triangular distribution parameterised with lower limit, $a$, upper limit, $b$, and mode, $c$, is defined by the pdf.

\[
\begin{align*}
    f(x) &= 0, \quad x < a \\
    f(x) &= \frac{2(x-a)}{(b-a)(c-a)}, \quad a \leq x < c \\
    f(x) &= \frac{2}{b-a}, \quad x = c \\
    f(x) &= \frac{2(b-x)}{(b-a)(b-c)}, \quad c < x \leq b \\
    f(x) &= 0, \quad x > b \quad \text{for} \quad a, b, c \in \mathbb{R}, \quad a \leq c \leq b.
\end{align*}
\]

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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>lower</td>
<td>numeric</td>
<td>lower limit.</td>
</tr>
<tr>
<td>upper</td>
<td>numeric</td>
<td>upper limit.</td>
</tr>
<tr>
<td>mode</td>
<td>numeric</td>
<td>mode.</td>
</tr>
<tr>
<td>symmetric</td>
<td>logical</td>
<td>see details.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>
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

\[ mode = \frac{\text{lower} + \text{upper}}{2} \]

this cannot be changed after construction. If symmetric = FALSE (default) then mode can be updated after construction.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

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

Statistical Methods

- pdf(x1, ..., log = FALSE, simplify = TRUE)
- cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
- rand(n, simplify = TRUE)
- mean()
### Parameter Methods

- `parameters(id)`
- `getParameterValue(id, error = "warn")`
- `setParameterValue(..., lst = NULL, error = "warn")`

### Validation Methods

- `liesInSupport(x, all = TRUE, bound = FALSE)`
- `liesInType(x, all = TRUE, bound = FALSE)`

### Representation Methods

- `strprint(n = 2)`
- `print(n = 2)`
- `summary(full = T)`

### References


### See Also

- `listDistributions` for all available distributions. `Uniform` for the Uniform distribution.

### Examples

```r
Triangular$new(lower = 2, upper = 5, symmetric = TRUE)
```
TriangularKernel

```r
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**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality.</td>
</tr>
</tbody>
</table>
Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

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

**Link**
- decorators
- traits
- valueSupport
- variateForm
- type
- properties
- support
- symmetry
- sup
- inf
- dmax
- dmin
- skewnessType
- kurtosisType

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

**Link**
- pdf
- cdf
- quantile.Distribution
- rand

**Statistical Methods**
- squared2Norm()
- prec()
- stdev()
- mode(which = "all")
- mean()
- median()
- iqr()

**Link**
- squared2Norm
- prec
- stdev
- mode
- mean.Distribution
- median.Distribution
- iqr
correlation()

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

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

Tricube

Tricube Kernel

Description
Mathematical and statistical functions for the Tricube kernel defined by the pdf,
\[ f(x) = \frac{70}{81}(1 - |x|^3)^3 \]
over the support \( x \in (-1, 1) \).

Details
The 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)
Tricube

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality.</td>
</tr>
</tbody>
</table>

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

**Accessor Methods**

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

**d/p/q/r Methods**

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

**Statistical Methods**

- Link
**Triweight Kernel**

**Description**

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

\[ f(x) = \frac{35}{32}(1 - x^2)^3 \]

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

**Details**

The quantile function is omitted as no closed form analytic expression could be found, decorate with FunctionImputation for numeric results.
Value

Returns an R6 object inheriting from class Kernel.

Constructor

Triweight$new(decorators = NULL)

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality.</td>
</tr>
</tbody>
</table>

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

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

d/p/q/r Methods

pdf(x1, ..., log = FALSE, simplify = TRUE)
**truncate**

**Description**

S3 functionality to truncate an R6 distribution.
TruncatedDistribution

Usage

truncate(x, lower = NULL, upper = NULL)

Arguments

x  Distribution.
lower  lower limit for truncation.
upper  upper limit for truncation.

See Also

TruncatedDistribution

Description

A wrapper for truncating any probability distribution at given limits.

Details

Truncates a distribution at lower and upper limits, using the formulae

\[
\begin{align*}
  f_T(x) &= f_X(x)/(F_X(\text{upper}) - F_X(\text{lower})) \\
  F_T(x) &= (F_X(x) - F_X(\text{lower}))/\left((F_X(\text{upper}) - F_X(\text{lower}))\right)
\end{align*}
\]

where \(f_T/F_T\) is the pdf/cdf of the truncated distribution \(T = \text{Truncate}(X, \text{lower}, \text{upper})\) and \(f_X, F_X\) is the pdf/cdf of the original distribution.

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>distribution</td>
<td>distribution</td>
<td>Distribution to truncate.</td>
</tr>
<tr>
<td>lower</td>
<td>numeric</td>
<td>Lower limit for truncation.</td>
</tr>
<tr>
<td>upper</td>
<td>numeric</td>
<td>Upper limit for truncation.</td>
</tr>
</tbody>
</table>
Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

**Accessor Methods**

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

**d/p/q/r Methods**

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

**Statistical Methods**

- prec()
- stdev()
- median()
- iqr()
- cor()
Parameter Methods
parameters(id)
getParameterValue(id, error = "warn")
setParameterValue(..., lst = NULL, error = "warn")

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

See Also
listWrappers, FunctionImputation, truncate

Examples
truncBin <- TruncatedDistribution$new(
    Binomial$new(prob = 0.5, size = 10),
    lower = 2, upper = 4)
truncBin$getParameterValue("prob")

type

<table>
<thead>
<tr>
<th>Type Accessor</th>
</tr>
</thead>
</table>

Description

Returns the scientific type of the distribution.

Usage

type(object)
**Uniform**

Arguments

object Distribution.

Value

An R6 object of class \texttt{set6::Set}.

R6 Usage

$\texttt{\$type()}$

See Also

\texttt{set6::Set}

---

**Uniform Distribution Class**

Description

Mathematical and statistical functions for the Uniform distribution, which is commonly used to model continuous events occurring with equal probability, as an uninformed prior in Bayesian modelling, and for inverse transform sampling.

Details

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

$$f(x) = \frac{1}{b - a}$$

for $-\infty < a < b < \infty$.
The distribution is supported on $[a, b]$.

Value

Returns an R6 object inheriting from class \texttt{SDistribution}.

Constructor

\texttt{Uniform\$new(lower = 0, upper = 1, decorators = NULL, verbose = FALSE)}

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>lower</td>
<td>integer</td>
<td>lower distribution limit.</td>
</tr>
<tr>
<td>upper</td>
<td>integer</td>
<td>upper distribution limit.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>
Constructor Details

The Uniform distribution is parameterised with `lower` and `upper` as numerics.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

**Accessor Methods**

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

**Statistical Methods**

- `pdf(x1, ..., log = FALSE, simplify = TRUE)`
- `cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`
- `quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`
- `rand(n, simplify = TRUE)`
- `mean()`
- `variance()`
- `stdev()`
- `prec()`
- `cor()`
- `skewness()`
- `kurtosis(excess = TRUE)`

**Statistical Methods**

- `pdf` [link: decorators]
- `cdf` [link: traits]
- `quantile.Distribution` [link: valueSupport]
- `rand` [link: variateForm]
- `mean.Distribution` [link: type]
- `variance` [link: properties]
- `stdev` [link: support]
- `prec` [link: symmetry]
- `cor` [link: sup]
- `skewness` [link: inf]
- `kurtosis` [link: dmax]
Uniform

entropy(base = 2)
mgf(t)
cf(t)
pgf(z)
median()
iqr()
mode(which = "all")

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

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

Author(s)
Yumi Zhou

References

See Also
listDistributions for all available distributions.

Examples
x <- Uniform$new(lower = -10, upper = 5)

# Update parameters
UniformKernel

**Description**

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

\[ f(x) = \frac{1}{2} \]

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

**Value**

Returns an R6 object inheriting from class Kernel.

**Constructor**

```
UniformKernel$new(decorators = NULL)
```

**Constructor Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality.</td>
</tr>
</tbody>
</table>

**Public Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
</tbody>
</table>
description  Brief description of distribution.
package  The package d/p/q/r are implemented in.

Public Methods

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

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

**Statistical Methods**
- squared2Norm()
- prec()
- stdev()
- mode(which = "all")
- mean()
- median()
- iqr()
- correlation()

**Parameter Methods**
- parameters(id)
- getParameterValue(id, error = "warn")
update.ParameterSet

update(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.
valueSupport

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

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

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

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

Value
Variance as a numeric.

R6 Usage
\$variance()

See Also
\texttt{CoreStatistics}, \texttt{decorate} and \texttt{genExp}.

---

**Variate Form Accessor**

<table>
<thead>
<tr>
<th>variateForm</th>
<th>Variate Form Accessor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description
Returns the variateForm of the distribution.

Usage
\texttt{variateForm(object)}

Arguments
\texttt{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 distribution is intended to vectorize distributions more efficiently than storing a list of
distributions. To improve speed and reduce memory usage, distributions are only constructed when
methods (e.g. d/p/q/r) are called. Whilst it is recommended to first extract distributions using [ before
querying them for results, all common methods are available in VectorDistribution as they are wrapped in apply.

Value
Returns an R6 object of class VectorDistribution.

Constructor
VectorDistribution$new(distlist = NULL, distribution = NULL, params = NULL, name = NULL,
short_name = NULL, description = NULL, decorators = NULL)

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>distlist</td>
<td>list</td>
<td>List of distributions.</td>
</tr>
<tr>
<td>distribution</td>
<td>character</td>
<td>Distribution to wrap.</td>
</tr>
<tr>
<td>params</td>
<td>a R object</td>
<td>Either list of parameters or matrix-type frame, see examples.</td>
</tr>
<tr>
<td>shared_params</td>
<td>a R object</td>
<td>Either list of shared parameters or matrix-type frame, see examples.</td>
</tr>
<tr>
<td>name</td>
<td>list</td>
<td>Optional new name for distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>list</td>
<td>Optional new short_name for distribution.</td>
</tr>
<tr>
<td>description</td>
<td>list</td>
<td>Optional new description for distribution.</td>
</tr>
<tr>
<td>decorators</td>
<td>list</td>
<td>Decorators to pass to wrapped distributions on construction.</td>
</tr>
</tbody>
</table>

Constructor Details
A vector distribution can either be constructed by a list of distributions passed to distlist or by
passing the name of one or more distributions 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 wrapped distributions but the latter is vastly faster for distributions of class SDistribution or
custom distributions. The shared_params parameter decreases memory usage and improves speed
by storing any parameters shared between distributions only once (instead of repeated in a list).
Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

**Accessor Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>wrappedModels(model = NULL)</td>
<td>wrappedModels</td>
</tr>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
</tr>
<tr>
<td>kurtosisType()</td>
<td>kurtosisType</td>
</tr>
</tbody>
</table>

**d/p/q/r Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdf(x1, ..., log = FALSE, simplify = TRUE)</td>
<td>pdf</td>
</tr>
<tr>
<td>cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)</td>
<td>cdf</td>
</tr>
<tr>
<td>quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)</td>
<td>quantile.Distribution</td>
</tr>
<tr>
<td>rand(n, simplify = TRUE)</td>
<td>rand</td>
</tr>
</tbody>
</table>

**Statistical Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>prec()</td>
<td>prec</td>
</tr>
<tr>
<td>stdev()</td>
<td>stdev</td>
</tr>
<tr>
<td>median()</td>
<td>median.Distribution</td>
</tr>
<tr>
<td>iqr()</td>
<td>iqr</td>
</tr>
<tr>
<td>cor()</td>
<td>cor</td>
</tr>
</tbody>
</table>
Parameter Methods

- parameters(id)
- getParameterValue(id, error = "warn")
- setParameterValue(..., lst = NULL, error = "warn")

Validation Methods

- liesInSupport(x, all = TRUE, bound = FALSE)
- liesInType(x, all = TRUE, bound = FALSE)

Representation Methods

- strprint(n = 2)
- print(n = 2)
- summary(full = T)

See Also

- `listWrappers` and `ProductDistribution`

Examples

```r
# not run to save time
## Not run:
vecDist <- VectorDistribution$new(list(Binomial$new(prob = 0.5, size = 10), Normal$new(mean = 15)))
vecDist$pdf(x1 = 2, x2 =3)
# Equivalently
# Or to evaluate every distribution at the same point
vecDist$pdf(1)

# Same wrapping for statistical functions
vecDist$mean()
vecDist$entropy()
vecDist$cdf(1:5, 12:16)
vecDist$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)))
```
Wald

Wald Distribution Class

Description
Mathematical and statistical functions for the Wald distribution, which is commonly used for modelling the first passage time for Brownian motion.

Details
The Wald distribution parameterised with mean, $\mu$, and shape, $\lambda$, is defined by the pdf,

$$f(x) = \frac{\lambda}{(2x^3\pi)}^{1/2} \exp\left((-\lambda(x - \mu)^2)/(2\mu^2x)\right)$$

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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>numeric</td>
<td>location parameter.</td>
</tr>
<tr>
<td>shape</td>
<td>numeric</td>
<td>shape parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

The Wald distribution is parameterised with mean and shape as positive numerics.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

**Accessor Methods**

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

**Statistical Methods**

- pdf(x1, ..., log = FALSE, simplify = TRUE)
- cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)
rand(n, simplify = TRUE)
mean()
variance()
stdev()
pred()
cor()
skewness()
kurtosis(excess = TRUE)
entropy(base = 2)
mgf(t)
cf(t)
pgf(z)
median()
iqr()
mode(which = "all")

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

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)

References
Michael P. McLaughlin.

Weibull

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

```r
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 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, \( \alpha \), and scale, \( \beta \), 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, altscale = NULL, decorators = NULL, verbose = FALSE)`

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>shape</td>
<td>numeric</td>
<td>shape parameter.</td>
</tr>
<tr>
<td>scale</td>
<td>numeric</td>
<td>scale parameter.</td>
</tr>
<tr>
<td>altscale</td>
<td>numeric</td>
<td>alternative scale parameter.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

The Weibull distribution is parameterised with shape, scale, and altscale as positive numerics.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

<table>
<thead>
<tr>
<th>Link</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators</td>
<td>decorators</td>
</tr>
<tr>
<td>traits</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm</td>
<td>variateForm</td>
</tr>
<tr>
<td>type</td>
<td>type</td>
</tr>
<tr>
<td>properties</td>
<td>properties</td>
</tr>
<tr>
<td>support</td>
<td>support</td>
</tr>
<tr>
<td>symmetry</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup</td>
<td>sup</td>
</tr>
<tr>
<td>inf</td>
<td>inf</td>
</tr>
<tr>
<td>dmax</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType</td>
<td>skewnessType</td>
</tr>
<tr>
<td>kurtosisType</td>
<td>kurtosisType</td>
</tr>
</tbody>
</table>
**Statistical Methods**

- `pdf(x1, ..., log = FALSE, simplify = TRUE)`
- `cdf(x1, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`
- `quantile(p, ..., lower.tail = TRUE, log.p = FALSE, simplify = TRUE)`
- `rand(n, simplify = TRUE)`
- `mean()`
- `variance()`
- `stdev()`
- `prec()`
- `cor()`
- `skewness()`
- `kurtosis(excess = TRUE)`
- `entropy(base = 2)`
- `mgf(t)`
- `cf(t)`
- `pgf(z)`
- `median()`
- `iqr()`
- `mode(which = "all")`

**Parameter Methods**

- `parameters(id)`
- `getParameterValue(id, error = "warn")`
- `setParameterValue(..., lst = NULL, error = "warn")`

**Validation Methods**

- `liesInSupport(x, all = TRUE, bound = FALSE)`
- `liesInType(x, all = TRUE, bound = FALSE)`

**Representation Methods**

- `strprint(n = 2)`
- `print(n = 2)`
- `summary(full = T)`

**References**

### WeightedDiscrete Distribution Class

#### Description

Mathematical and statistical functions for the WeightedDiscrete distribution, which is commonly used in empirical estimators such as Kaplan-Meier.

#### Details

The WeightedDiscrete distribution is defined by the pmf,

\[ f(x_i) = p_i \]

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

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

Sampling from this distribution is performed with the `sample` function with the elements given as the \( x \) values and the pdf as the probabilities. The cdf and quantile 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

WeightedDiscrete$new(data, decorators = NULL, verbose = FALSE)

Constructor Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>data.frame</td>
<td>matrix-style object of observations and probabilities. See details.</td>
</tr>
<tr>
<td>decorators</td>
<td>Decorator</td>
<td>decorators to add functionality. See details.</td>
</tr>
<tr>
<td>verbose</td>
<td>logical</td>
<td>if TRUE parameterisation messages produced.</td>
</tr>
</tbody>
</table>

Constructor Details

The WeightedDiscrete distribution is parameterised with an object that can be coerced to a data.frame containing columns ‘sample’ and at least one of ‘pdf’ and ‘cdf’, see examples.

Public Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Name of distribution.</td>
</tr>
<tr>
<td>short_name</td>
<td>Id of distribution.</td>
</tr>
<tr>
<td>description</td>
<td>Brief description of distribution.</td>
</tr>
<tr>
<td>package</td>
<td>The package d/p/q/r are implemented in.</td>
</tr>
</tbody>
</table>

Public Methods

Accessor Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>decorators()</td>
<td>decorators</td>
</tr>
<tr>
<td>traits()</td>
<td>traits</td>
</tr>
<tr>
<td>valueSupport()</td>
<td>valueSupport</td>
</tr>
<tr>
<td>variateForm()</td>
<td>variateForm</td>
</tr>
<tr>
<td>type()</td>
<td>type</td>
</tr>
<tr>
<td>properties()</td>
<td>properties</td>
</tr>
<tr>
<td>support()</td>
<td>support</td>
</tr>
<tr>
<td>symmetry()</td>
<td>symmetry</td>
</tr>
<tr>
<td>sup()</td>
<td>sup</td>
</tr>
<tr>
<td>inf()</td>
<td>inf</td>
</tr>
<tr>
<td>dmax()</td>
<td>dmax</td>
</tr>
<tr>
<td>dmin()</td>
<td>dmin</td>
</tr>
<tr>
<td>skewnessType()</td>
<td>skewnessType</td>
</tr>
</tbody>
</table>
kurtosisType()

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

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

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

Representation Methods
strprint(n = 2)
print(n = 2)
summary(full = T)
wrappedModels

References


See Also

listDistributions for all available distributions. sample for the sampling function and Empirical for the closely related Empirical distribution.

Examples

```r
x = WeightedDiscrete$new(data = data.frame(x = 1:3, pdf = c(1/5, 3/5, 1/5)))
WeightedDiscrete$new(data = data.frame(x = 1:3, cdf = c(1/5, 4/5, 1))) # equivalently

# d/p/q/r
x$pdf(1:5)
x$cdf(1:5) # Assumes ordered in construction
x$quantile(0.42) # Assumes ordered in construction
x$rand(10)

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

summary(x)
```

wrappedModels

Gets Internally Wrapped Models

Description

Returns either a list of all the wrapped models or the models named by parameters.

Usage

```r
wrappedModels(object, model = NULL)
```

Arguments

- **object**: Distribution.
- **model**: character, see details.

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
```r
$wrappedModels(model = NULL)
```

See Also
`DistributionWrapper`
Index

[.VectorDistribution
  (Extract.VectorDistribution), 78

Arcsine, 8
ArrayDistribution, 32, 176, 193
as.data.table, 172
as.data.table.ParameterSet, 11
as.ParameterSet, 11
assertContinuous (testContinuous), 221
assertDiscrete (testDiscrete), 222
assertDistribution (testDistribution), 222
assertDistributionList
  (testDistributionList), 223
assertLeptokurtic (testLeptokurtic), 224
assertMatrixvariate
  (testMatrixvariate), 225
assertMesokurtic (testMesokurtic), 226
assertMixture (testMixture), 227
assertMultivariate (testMultivariate), 227
assertNegativeSkew (testNegativeSkew), 228
assertNoSkew (testNoSkew), 229
assertPlatykurtic (testPlatykurtic), 230
assertPositiveSkew (testPositiveSkew), 231
assertSymmetric (testSymmetric), 232
assertUnivariate (testUnivariate), 232

Bernoulli, 12
Beta, 15, 54
BetaNoncentral, 18
Binomial, 15, 21, 157, 164

c.Distribution, 24
Categorical, 25
Cauchy, 28

cdf, 9, 14, 17, 20, 23, 27, 30, 31, 36, 39, 45,
  50, 53, 56, 60, 63, 68, 71, 76, 80, 83,
  86, 88, 90, 95, 100, 103, 107, 110,
  114, 119, 125, 133, 136, 139, 142,
  145, 152, 156, 159, 163, 166, 169,
  174, 177, 182, 186, 190, 191, 193,
  195, 197, 200, 203, 210, 213, 235,
  238, 240, 243, 245, 248, 251, 256,
  259, 263, 266
cdfAntiDeriv, 32, 74
cdfPNorm, 33, 74
cf, 9, 14, 17, 20, 23, 27, 30, 34, 36, 39, 42, 50,
  53, 56, 68, 76, 80, 83, 86, 91, 96,
  100, 103, 111, 114, 125, 134, 137,
  142, 145, 156, 159, 163, 167, 174,
  182, 195, 197, 211, 214, 236, 249,
  260, 263, 266
checkContinuous (testContinuous), 221
checkDiscrete (testDiscrete), 222
checkDistribution (testDistribution), 222
checkDistributionList
  (testDistributionList), 223
checkLeptokurtic (testLeptokurtic), 224
checkMatrixvariate (testMatrixvariate), 225
checkMesokurtic (testMesokurtic), 226
checkMixture (testMixture), 227
checkMultivariate (testMultivariate), 227
checkNegativeSkew (testNegativeSkew), 228
checkNoSkew (testNoSkew), 229
checkPlatykurtic (testPlatykurtic), 230
checkPositiveSkew (testPositiveSkew), 231
checkSymmetric (testSymmetric), 232
checkUnivariate (testUnivariate), 232
ChiSquared, 35, 40, 81, 84

269
ChiSquaredNoncentral, 37, 38
chol, 158, 160
Convolution, 41
correlation, 43, 45, 60, 72, 120, 139, 170, 192, 201, 203, 239, 241, 243, 251
Cosine, 44
cumHazard, 46, 74
decorate, 32–34, 42, 43, 46, 47, 48, 59, 61, 62, 70, 73, 74, 88, 93, 105, 121, 122, 148, 151, 154, 177, 178, 190, 193, 205, 207, 218–220, 254
Degenerate, 48
Delta(Degenerate), 48
Dirac(Degenerate), 48
Dirichlet, 51
DiscreteUniform, 55
distr6 (distr6-package), 6
distr6-package, 6
distr6News, 58
Distribution, 58, 172, 185, 216
DistributionDecorator, 61, 129
DistributionWrapper, 62, 132, 185, 268
distrSimulate, 64
Empirical, 66, 189, 204, 267
entropy, 9, 14, 17, 20, 23, 27, 30, 36, 39, 42, 50, 53, 56, 68, 69, 76, 80, 83, 86, 91, 96, 100, 103, 111, 125, 133, 137, 142, 145, 156, 159, 163, 167, 174, 182, 195, 197, 211, 214, 236, 249, 260, 263, 266
Epanechnikov, 70
exkurosisType, 72, 206
ExoticStatistics, 33, 34, 46, 73, 105, 177, 207, 218–220
Exponential, 74
Extract.VectorDistribution, 78
FDistribution, 78, 84
FDistributionNoncentral, 81, 81
Fisk (Loglogistic), 140
Frechet, 84, 104, 264
Gamma, 89
gammaz, 104
Gaussian (Normal), 165
generalPNorm, 92
genExp, 43, 93, 148, 254
Geometric, 94, 164
getParameterSupport, 97, 172
getParameterValue, 10, 14, 17, 20, 23, 27, 30, 36, 40, 45, 50, 53, 57, 60, 63, 68, 72, 76, 80, 83, 86, 91, 96, 98, 101
makeUniqueDistributions, 147
mean, 93
median.Distribution, 9, 14, 17, 20, 23, 27, 30, 36, 39, 45, 50, 53, 56, 60, 63, 68, 72, 76, 80, 83, 86, 91, 96, 100, 103, 108, 111, 114, 119, 125, 134, 137, 139, 142, 145, 156, 159, 163, 166, 169, 174, 182, 192, 195, 197, 201, 203, 211, 214, 236, 238, 241, 243, 245, 249, 251, 256, 260, 263, 266
merge.ParameterSet, 149, 172
mgf, 9, 14, 17, 20, 23, 27, 30, 36, 39, 42, 50, 53, 56, 68, 72, 76, 80, 83, 86, 91, 96, 100, 103, 111, 114, 125, 133, 137, 142, 145, 150, 156, 159, 163, 167, 174, 182, 195, 197, 211, 214, 236, 249, 260, 263, 266
MixtureDistribution, 151
mode, 9, 14, 17, 20, 23, 27, 30, 36, 39, 43, 45, 50, 53, 56, 68, 72, 76, 80, 83, 86, 91, 96, 100, 103, 111, 114, 119, 125, 134, 137, 139, 142, 145, 153, 156, 160, 163, 167, 169, 175, 182, 192, 195, 197, 201, 203, 211, 214, 236, 238, 241, 243, 249, 251, 260, 263, 266
Multinomial, 154
MultivariateNormal, 157
NegativeBinomialNormal, 161
Normal, 37, 40, 81, 84, 146, 160, 165, 211, 215, 261
NormalKernel, 168
par, 179
ParameterSet, 11, 12, 59, 61, 150, 171, 184, 253
Pareto, 173
pdfPNorm, 74, 177
pgf, 9, 14, 17, 20, 23, 27, 30, 36, 39, 42, 50, 53, 56, 68, 76, 80, 83, 86, 91, 96, 100, 103, 111, 114, 125, 134, 137, 142, 145, 156, 160, 163, 167, 174, 178, 182, 195, 197, 211, 214, 236, 249, 260, 263, 266
plot.Distribution, 128, 179, 189
Poisson, 180
print.ParameterDistribution, 172, 184
ProductDistribution, 32, 176, 185, 190, 193, 257
properties, 9, 13, 16, 19, 22, 26, 29, 36, 39, 45, 50, 53, 56, 60, 62, 67, 71, 76, 79, 82, 86, 90, 95, 100, 103, 107, 110, 114, 119, 123, 124, 133, 136, 139,
survival, 9, 13, 16, 19, 22, 26, 29, 36, 39, 45,
50, 53, 56, 60, 62, 65–67, 71, 76, 79, 82, 86, 90, 95, 100, 103, 107, 110,
112, 114, 119, 124, 133, 136, 139,
141, 145, 152, 155, 159, 163, 166,
169, 174, 181, 186, 191, 194, 197, 200, 202, 210, 213, 216, 235, 238, 240,
242, 245, 248, 251, 256, 259, 262, 265

support, 9, 13, 16, 19, 22, 26, 29, 36, 39, 45,
50, 53, 56, 60, 62, 65–67, 71, 76, 79, 82, 86, 90, 95, 100, 103, 107, 110,
112, 114, 119, 124, 133, 136, 139,
141, 145, 152, 155, 159, 163, 166,
169, 174, 181, 186, 191, 194, 197, 200, 202, 210, 213, 216, 217, 235,
238, 240, 242, 245, 248, 251, 256,
259, 262, 265

symmetry, 9, 13, 16, 19, 22, 26, 29, 36, 39, 45,
50, 53, 56, 60, 62, 67, 71, 76, 79, 82, 86, 90, 95, 100, 103, 107, 110, 114,
119, 124, 133, 136, 139, 141, 145,
152, 155, 159, 163, 166, 169, 174,
181, 186, 191, 194, 197, 200, 202,
210, 213, 220, 235, 238, 240, 242,
245, 248, 251, 256, 259, 262, 265

testContinuous, 221
testDiscrete, 222
testDistribution, 222
testDistributionList, 223
testLeptokurtic, 224
testMatrixkurtic, 225
testMesokurtic, 226
testMixture, 227
testMultivariate, 227
testNegativeSkew, 228
testNoSkew, 229
testPlatykurtic, 230
testPositiveSkew, 231
testSymmetric, 232
testUnivariate, 232
traits, 9, 13, 16, 19, 22, 26, 29, 36, 39, 45,
49, 53, 56, 60, 62, 67, 71, 76, 79, 82,
85, 90, 95, 100, 103, 107, 110, 114,
119, 124, 133, 136, 139, 141, 145,
152, 155, 159, 163, 166, 169, 174,
181, 186, 191, 194, 197, 200, 202,
210, 213, 233, 235, 238, 240, 242,
245, 248, 251, 256, 259, 262, 265

TriangularKernel, 237
Tricube, 239
Triweight, 241
truncate, 243, 246
TruncatedDistribution, 244, 244

Uniform, 57, 236, 247
UniformKernel, 250
update.ParameterSet, 172, 252

valueSupport, 9, 13, 16, 19, 22, 26, 29, 36,
39, 45, 50, 53, 56, 60, 62, 67, 71, 76,
79, 82, 85, 90, 95, 100, 103, 107,
110, 114, 119, 124, 133, 136, 139,
141, 145, 152, 155, 159, 163, 166,
169, 174, 181, 186, 191, 194, 197,
200, 202, 210, 213, 235, 238, 240,
242, 245, 248, 251, 253, 256, 259,
262, 265

variance, 9, 14, 17, 20, 23, 27, 30, 36, 39, 43,
50, 53, 56, 68, 76, 80, 83, 86, 90, 95,
100, 103, 110, 114, 125, 133, 136,
142, 145, 156, 159, 163, 166, 174,
182, 184, 195, 197, 208, 210, 214,
216, 248, 253, 260, 263, 266

variateForm, 9, 13, 16, 19, 22, 26, 29, 36, 39,
45, 50, 53, 56, 60, 62, 67, 71, 76, 79,
82, 85, 90, 95, 100, 103, 107, 110,
114, 119, 124, 133, 136, 139, 141,
145, 152, 155, 159, 163, 166, 169,
174, 181, 186, 191, 194, 197, 200,
202, 210, 213, 235, 238, 240, 242,
245, 248, 251, 254, 256, 259, 262,
265

Wald, 258
Weibull, 87, 104, 261
INDEX

WeightedDiscrete, 69, 264
wrappedModels, 62, 107, 152, 186, 245, 256, 267