The distr Package

March 30, 2005

Title Object orientated implementation of distributions

Version 1.5

Depends R(>= 2.0.0), methods, graphics, setRNG

Imports stats

LazyLoad yes

SaveImage no

Author Florian Camphausen, Matthias Kohl, Peter Ruckdeschel, Thomas Stabla

Description Object orientated implementation of distributions and some additional functionality

Maintainer Peter Ruckdeschel <Peter.Ruckdeschel@uni-bayreuth.de>

License GPL (version 2 or later)

URL http://www.uni-bayreuth.de/departments/math/org/mathe7/DISTR/

R topics documented:

(INTRO.DISTR) *DISTR*

Description

"distr" is a package for R from version 1.8.1 onwards that is distributed under GPL license 2.0. Its own current version is 1.5. It requires package **setRNG** by Paul Gilbert, ⟨pgilbert@bank-banque-canada.ca⟩, to be installed from

http://cran.r-project.org/mirrors.html

The aim of this package is to provide a conceptual treatment of random variables (r.v.'s) by means of S4 classes. A mother class `Distribution` is introduced with slots for a parameter and — most important— for the four constitutive methods `r`, `d`, `p`, and `q` for simulation respectively for evaluation of density / c.d.f. and quantile function of the corresponding distribution. All distributions of the **base** package for which corresponding `r`-, `d`-, `p`-, and `q`-functions exist (like normal, Poisson, etc.) are implemented as subclasses of either `AbscontDistribution` or `DiscreteDistribution`, which themselves are again subclasses of `Distribution`.

This approach seems very appealing to us from a conceptual viewpoint:

Just pass an object of some derived distribution class to a generic function as argument and let the dispatching mechanism decide what to do on run-time.

As an example, we may automatically generate new objects of these classes with corresponding `r`, `d`, `p`, and `q`-slots for the laws of r.v.'s under standard mathematical univariate transformations and under convolution of independent r.v.'s. For `Distribution` objects `X` and `Y` expressions like `3*X+sin(exp(-Y/4+3))` have their natural interpretation as corresponding image distributions.

Additionally, we also provide classes for a standardized treatment of simulations (also under contaminations) and evaluations of statistical procedures on such simulations. A somewhat longer and more detailed manual can be obtained under

http://www.uni-bayreuth.de/departments/math/org/mathe7/DISTR/

Usage

```
library("distr")
```

AbscontDistribution-class *Class "AbscontDistribution"*

Description

The `AbscontDistribution`-class is the mother-class of the classes `Beta`, `Cauchy`, `Chisq`, `Exp`, `F`, `Gammad`, `Lnorm`, `Logis`, `Norm`, `T`, `Unif` and `Weibull`. Further absolutely continuous distributions can be defined either by declaration of own random number generator, density, cumulative distribution and quantile functions, or as result of a convolution of two absolutely continuous distributions or by application of a mathematical operator to an absolutely continuous distribution. An additional way is, to specify only the random number generator. The function `RtoDPQ` then approximates the three remaining slots d, p and q by random sampling.

Objects from the Class

Objects can be created by calls of the form `new("AbscontDistribution", r, d, p, q)`. The result of this call is an absolutely continuous distribution.

Slots

`img`: Object of class `"Reals"`: the space of the image of this distribution which has dimension 1 and the name "Real Space"

`param`: Object of class `"Parameter"`: the parameter of this distribution, having only the slot name "Parameter of an absolutely continuous distribution"

`r`: Object of class `"function"`: generates random numbers

`d`: Object of class `"function"`: density function

`p`: Object of class `"function"`: cumulative distribution function

`q`: Object of class `"function"`: quantile function

Extends

Class `"UnivariateDistribution"`, directly.
Class `"Distribution"`, by class `"UnivariateDistribution"`.

Methods

initialize `signature(.Object = "AbscontDistribution")`: initialize method

Math `signature(x = "AbscontDistribution")`: application of a mathematical function, e.g. `sin` or `exp` (does not work with `log`!), to this absolutely continouos distribution

`-` `signature(e1 = "AbscontDistribution")`: application of '-' to this absolutely continuous distribution

`*` `signature(e1 = "AbscontDistribution", e2 = "numeric")`: multiplication of this absolutely continuous distribution by an object of class 'numeric'

`/` `signature(e1 = "AbscontDistribution", e2 = "numeric")`: division of this absolutely continuous distribution by an object of class 'numeric'

`+` `signature(e1 = "AbscontDistribution", e2 = "numeric")`: addition of this absolutely continuous distribution to an object of class 'numeric'

`-` `signature(e1 = "AbscontDistribution", e2 = "numeric")`: subtraction of an object of class 'numeric' from this absolutely continuous distribution

`*` `signature(e1 = "numeric", e2 = "AbscontDistribution")`: multiplication of this absolutely continuous distribution by an object of class 'numeric'

`+` `signature(e1 = "numeric", e2 = "AbscontDistribution")`: addition of this absolutely continuous distribution to an object of class 'numeric'

`-` `signature(e1 = "numeric", e2 = "AbscontDistribution")`: subtraction of this absolutely continuous distribution from an object of class 'numeric'

`+` `signature(e1 = "AbscontDistribution", e2 = "AbscontDistribution")`: Convolution of two absolutely continuous distributions. The slots p, d and q are approximated by grids.

`-` `signature(e1 = "AbscontDistribution", e2 = "AbscontDistribution")`: Convolution of two absolutely continuous distributions. The slots p, d and q are approximated by grids.

plot `signature(object = "AbscontDistribution")`: plots density, cumulative distribution and quantile function

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

Parameter-class UnivariateDistribution-class Beta-class Cauchy-class Chisq-class Exp-class Fd-class Gammad-class Lnorm-class Logis-class Norm-class Td-class Unif-class Weibull-class DiscreteDistribution-class Reals-class RtoDPQ

Examples

```
N = Norm() # N is a normal distribution with mean=0 and sd=1.
E = Exp() # E is an exponential distribution with rate=1.
A1 = E+1 # a new absolutely continuous distributions with exact slots d, p, q
A2 = A1*3 # a new absolutely continuous distributions with exact slots d, p, q
A3 = N*0.9 + E*0.1 # a new absolutely continuous distribution with approximated slots d,
r(A3)(1) # one random number generated from this distribution, e.g. -0.7150937
d(A3)(0) # The (approximated) density for x=0 is 0.4379882.
p(A3)(0) # The (approximated) probability that x <= 0 is 0.4562021.
q(A3)(.1) # The (approximated) 10 percent quantile is 0.1.
```

Beta-class *Class "Beta"*

Description

The Beta distribution with parameters `shape1` $= a$ and `shape2` $= b$ has density

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^a (1-x)^b$$

for $a > 0$, $b > 0$ and $0 \leq x \leq 1$ where the boundary values at $x = 0$ or $x = 1$ are defined as by continuity (as limits).

C.f. rbeta

Objects from the Class

Objects can be created by calls of the form `Beta(shape1, shape2)`. This object is a beta distribution.

Slots

img: Object of class `"Reals"`: The space of the image of this distribution has got dimension 1 and the name "Real Space".

param: Object of class `"BetaParameter"`: the parameter of this distribution (shape1 and shape2), declared at its instantiation

r: Object of class `"function"`: generates random numbers (calls function rbeta)

d: Object of class `"function"`: density function (calls function dbeta)

p: Object of class `"function"`: cumulative function (calls function pbeta)

q: Object of class `"function"`: inverse of the cumulative function (calls function qbeta)

Extends

Class `"AbscontDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`.
Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

initialize `signature(.Object = "Beta")`: initialize method

shape1 `signature(object = "Beta")`: returns the slot shape1 of the parameter of the distribution

shape1`<- signature(object = "Beta")`: modifies the slot shape1 of the parameter of the distribution

shape2 `signature(object = "Beta")`: returns the slot shape2 of the parameter of the distribution

shape2`<- signature(object = "Beta")`: modifies the slot shape2 of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`BetaParameter-class` `AbscontDistribution-class` `Reals-class` `rbeta`

Examples

```
B <- Beta(shape1 = 1, shape2 = 1)
# B is a beta distribution with shape1 = 1 and shape2 = 1.
r(B)(1) # one random number generated from this distribution, e.g. 0.6979795
d(B)(1) # Density of this distribution is 1 for x=1.
p(B)(1) # Probability that x < 1 is 1.
q(B)(.1) # Probability that x < 0.1 is 0.1.
shape1(B) # shape1 of this distribution is 1.
shape1(B) <- 2 # shape1 of this distribution is now 2.
```

`BetaParameter-class`
Class "BetaParameter"

Description

The parameter of a beta distribution, used by `Beta-class`

Objects from the Class

Objects can be created by calls of the form `new("BetaParameter", shape1, shape2)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Beta is instantiated.

Slots

shape1: Object of class `"numeric"`: the shape1 of a beta distribution
shape2: Object of class `"numeric"`: the shape2 of a beta distribution
name: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "BetaParameter")`: initialize method
shape1 `signature(object = "BetaParameter")`: returns the slot shape1 of the parameter of the distribution
shape1<- `signature(object = "BetaParameter")`: modifies the slot shape1 of the parameter of the distribution
shape2 `signature(object = "BetaParameter")`: returns the slot shape2 of the parameter of the distribution
shape2<- `signature(object = "BetaParameter")`: modifies the slot shape2 of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Beta-class` `Parameter-class`

Examples

```
W <- new("BetaParameter", shape1 = 1, shape2 = 1)
shape2(W) # shape2 of this distribution is 1.
shape2(W) <- 2 # shape2 of this distribution is now 2.
```

`Binom-class` *Class "Binom"*

Description

The binomial distribution with `size` $= n$, by default $= 1$, and `prob` $= p$, by default $= 0.5$, has density

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

for $x = 0, \ldots, n$.

C.f.`rbinom`

Objects from the Class

Objects can be created by calls of the form `Binom(prob, size)`. This object is a binomial distribution.

Slots

`img`: Object of class `"Naturals"`: The space of the image of this distribution has got dimension 1 and the name "Natural Space".

`param`: Object of class `"BinomParameter"`: the parameter of this distribution (`prob`, `size`), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function `rbinom`)

`d`: Object of class `"function"`: density function (calls function `dbinom`)

`p`: Object of class `"function"`: cumulative function (calls function `pbinom`)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function `qbinom`). The quantile is defined as the smallest value x such that F(x) >= p, where F is the cumulative function.

`support`: Object of class `"numeric"`: a (sorted) vector containing the support of the discrete density function

Extends

Class `"DiscreteDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"DiscreteDistribution"`.
Class `"Distribution"`, by class `"DiscreteDistribution"`.

Methods

`+ signature(e1 = "Binom", e2 = "Binom")`: For two binomial distributions with equal probabilities the exact convolution formula is implemented thereby improving the general numerical approximation.

initialize `signature(.Object = "Binom")`: initialize method

prob `signature(object = "Binom")`: returns the slot `prob` of the parameter of the distribution

prob`<- signature(object = "Binom")`: modifies the slot `prob` of the parameter of the distribution

size `signature(object = "Binom")`: returns the slot `size` of the parameter of the distribution

size<- `signature(object = "Binom")`: modifies the slot `size` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`BinomParameter-class` `DiscreteDistribution-class` `Naturals-class` `rbinom`

Examples

```
B=Binom(prob=0.5,size=1) # B is a binomial distribution with prob=0.5 and size=1.
r(B)(1) # # one random number generated from this distribution, e.g. 1
d(B)(1) # Density of this distribution is  0.5 for x=1.
p(B)(0.4) # Probability that x<0.4 is 0.5.
q(B)(.1) # x=0 is the smallest value x such that p(B)(x)>=0.1.
size(B) # size of this distribution is 1.
size(B)=2 # size of this distribution is now 2.
C=Binom(prob=0.5,size=1) # C is a binomial distribution with prob=0.5 and size=1.
D=Binom(prob=0.6,size=1) # D is a binomial distribution with prob=0.6 and size=1.
E=B+C # E is a binomial distribution with prob=0.5 and size=3.
F=B+D # F is an object of class DiscreteDistribution.
```

BinomParameter-class

Class "BinomParameter"

Description

The parameter of a binomial distribution, used by Binom-class

Objects from the Class

Objects can be created by calls of the form `new("BinomParameter", prob, size)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Binom is instantiated.

Slots

`prob`: Object of class `"numeric"`: the probability of a binomial distribution

`size`: Object of class `"numeric"`: the size of a binomial distribution

`name`: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "BinomParameter")`: initialize method

prob `signature(object = "BinomParameter")`: returns the slot `prob` of the parameter of the distribution

prob<- `signature(object = "BinomParameter")`: modifies the slot `prob` of the parameter of the distribution

size `signature(object = "BinomParameter")`: returns the slot `size` of the parameter of the distribution

size<- `signature(object = "BinomParameter")`: modifies the slot `size` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Binom-class` `Parameter-class`

Examples

```
W=new("BinomParameter",prob=0.5,size=1)
size(W) # size of this distribution is 1.
size(W)=2 # size of this distribution is now 2.
```

`Cauchy-class` *Class "Cauchy"*

Description

The Cauchy distribution with location l, by default $= 0$, and scale s , by default $= 1$,has density

$$f(x) = \frac{1}{\pi s}\left(1 + \left(\frac{x-l}{s}\right)^2\right)^{-1}$$

for all x. C.f. `rcauchy`

Objects from the Class

Objects can be created by calls of the form `Cauchy(location, scale)`. This object is a Cauchy distribution.

Slots

img: Object of class "Reals": The domain of this distribution has got dimension 1 and the name "Real Space".

param: Object of class "CauchyParameter": the parameter of this distribution (location and scale), declared at its instantiation

r: Object of class "function": generates random numbers (calls function rcauchy)

d: Object of class "function": density function (calls function dcauchy)

p: Object of class "function": cumulative function (calls function pcauchy)

q: Object of class "function": inverse of the cumulative function (calls function qcauchy)

Extends

Class "AbscontDistribution", directly.
Class "UnivariateDistribution", by class "AbscontDistribution".
Class "Distribution", by class "AbscontDistribution".

Methods

initialize signature(.Object = "Cauchy"): initialize method

location signature(object = "Cauchy"): returns the slot location of the parameter of the distribution

location<- signature(object = "Cauchy"): modifies the slot location of the parameter of the distribution

scale signature(object = "Cauchy"): returns the slot scale of the parameter of the distribution

scale<- signature(object = "Cauchy"): modifies the slot scale of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

CauchyParameter-class AbscontDistribution-class Reals-class rcauchy

Examples

```
C=Cauchy(location=1,scale=1) # C is a Cauchy distribution with location=1 and scale=1.
r(C)(1) # one random number generated from this distribution, e.g. 4.104603
d(C)(1) # Density of this distribution is 0.3183099 for x=1.
p(C)(1) # Probability that x<1 is 0.5.
q(C)(.1) # Probability that x<-2.077684 is 0.1.
location(C) # location of this distribution is 1.
location(C)=2 # location of this distribution is now 2.
```

`CauchyParameter-class`
Class "CauchyParameter"

Description

The parameter of a Cauchy distribution, used by Cauchy-class

Objects from the Class

Objects can be created by calls of the form `new("CauchyParameter", location, scale)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Cauchy is instantiated.

Slots

`location`: Object of class `"numeric"`: the location of a Cauchy distribution
`scale`: Object of class `"numeric"`: the scale of a Cauchy distribution
`name`: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "CauchyParameter")`: initialize method
scale `signature(object = "CauchyParameter")`: returns the slot `scale` of the parameter of the distribution
scale<- `signature(object = "CauchyParameter")`: modifies the slot `scale` of the parameter of the distribution
location `signature(object = "CauchyParameter")`: returns the slot `location` of the parameter of the distribution
location<- `signature(object = "CauchyParameter")`: modifies the slot `location` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Cauchy-class` `Parameter-class`

Examples

```
W=new("CauchyParameter",location=1,scale=1)
location(W) # location of this distribution is 1.
location(W)=2 # location of this distribution is now 2.
```

`Chisq-class` *Class "Chisq"*

Description

The chi-squared distribution with `df`$= n$ degrees of freedom has density

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

for $x > 0$. The mean and variance are n and $2n$.

The non-central chi-squared distribution with `df`$= n$ degrees of freedom and non-centrality parameter `ncp` $= \lambda$ has density

$$f(x) = e^{-\lambda/2} \sum_{r=0}^{\infty} \frac{(\lambda/2)^r}{r!} f_{n+2r}(x)$$

for $x \geq 0$. For integer n, this is the distribution of the sum of squares of n normals each with variance one, λ being the sum of squares of the normal means.

C.f. `rchisq`

Objects from the Class

Objects can be created by calls of the form `Chisq(df, ncp)`. This object is a chi-squared distribution.

Slots

`img`: Object of class `"Reals"`: The space of the image of this distribution has got dimension 1 and the name "Real Space".

`param`: Object of class `"ChisqParameter"`: the parameter of this distribution (df and ncp), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function rchisq)

`d`: Object of class `"function"`: density function (calls function dchisq)

`p`: Object of class `"function"`: cumulative function (calls function pchisq)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function qchisq)

Extends

Class `"AbscontDistribution"`, directly. Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`. Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

initialize `signature(.Object = "Chisq")`: initialize method

df `signature(object = "Chisq")`: returns the slot df of the parameter of the distribution

df<- `signature(object = "Chisq")`: modifies the slot df of the parameter of the distribution

ncp `signature(object = "Chisq")`: returns the slot ncp of the parameter of the distribution

ncp<- `signature(object = "Chisq")`: modifies the slot ncp of the parameter of the distribution

Note

Warning: The code for pchisq and qchisq is unreliable for values of ncp above approximately 290.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`ChisqParameter-class` `AbscontDistribution-class` `Reals-class` `rchisq`

Examples

```
C <- Chisq(df = 1, ncp = 1) # C is a chi-squared distribution with df=1 and ncp=1.
r(C)(1) # one random number generated from this distribution, e.g. 0.2557184
d(C)(1) # Density of this distribution is 0.2264666 for x = 1.
p(C)(1) # Probability that x < 1 is 0.4772499.
q(C)(.1) # Probability that x < 0.04270125 is 0.1.
df(C) # df of this distribution is 1.
df(C) <- 2 # df of this distribution is now 2.
```

ChisqParameter-class

Class "ChisqParameter"

Description

The parameter of a chi-squared distribution, used by Chisq-class

Objects from the Class

Objects can be created by calls of the form `new("ChisqParameter", ncp, df)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Chisq is instantiated.

Slots

`ncp`: Object of class `"numeric"`: the ncp of a chi-squared distribution

`df`: Object of class `"numeric"`: the df of a chi-squared distribution

`name`: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "ChisqParameter")`: initialize method

df `signature(object = "ChisqParameter")`: returns the slot `df` of the parameter of the distribution

df<- `signature(object = "ChisqParameter")`: modifies the slot `df` of the parameter of the distribution

ncp `signature(object = "ChisqParameter")`: returns the slot `ncp` of the parameter of the distribution

ncp<- `signature(object = "ChisqParameter")`: modifies the slot `ncp` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Chisq-class` `Parameter-class`

Examples

```
W=new("ChisqParameter",df=1,ncp=1)
ncp(W) # ncp of this distribution is 1.
ncp(W)=2 # ncp of this distribution is now 2.
```

Contsimulation-class

Class "Contsimulation"

Description

In an object of type `Contsimulation` data can be simulated in any distribution and size. One part (usually the largest) of the random numbers stems from an ideal distribution, the rest is contaminated.

Objects from the Class

Objects can be created by calls of the form `Contsimulation(filename, runs, samplesize, seed, distribution.id, distribution.c, rate)`. A `Contsimulation`-object includes a filename, the number of runs, the size of the sample, the seed, the distribution of the ideal and the contaminated data and the contamination rate. The slot Data stays empty until the method simulate has been used.

Slots

ind: Object of class "vectororNULL": Indicator of the same length as the data; saves whether each element of the data vector is contaminated or not

Data.id: Object of class "vectororNULL": – the ideal data

Data.c: Object of class "vectororNULL": – the contaminated data

rate: Object of class "numeric": the contamination rate, so the probability for each random number to be contaminated

distribution.c: Object of class "UnivariateDistribution": the distribution of the ideal data

distribution.id: Object of class "UnivariateDistribution": the distribution of the contaminated data

seed: Object of class "list": the seed the simulation has been generated with

filename: Object of class "character": the filename the Contsimulation shall be saved

Data: Object of class "vectororNULL": the simulated data

runs: Object of class "numeric": the number of runs of the data

samplesize: Object of class "numeric": the size of the sample, so the dimension of the data

Extends

Class "Dataclass", directly.

Methods

Data.c signature(object = "Contsimulation"): returns the contaminated data

Data.id signature(object = "Contsimulation"): returns the ideal data

Data<- signature(object = "Contsimulation"): ERROR: A modification of simulated data is not allowed.

distribution.c signature(object = "Contsimulation"): returns the distribution of the contaminated data

distribution.c<- signature(object = "Contsimulation"): changes the distribution of the contaminated data

distribution.id signature(object = "Contsimulation"): returns the distribution of the ideal data

distribution.id<- signature(object = "Contsimulation"): changes the distribution of the ideal data

seed signature(object = "Contsimulation"): returns the seed

seed<- signature(object = "Contsimulation"): changes the seed

ind signature(object = "Contsimulation"): returns the indicator which saves which data is contaminated

initialize signature(.Object = "Contsimulation"): initialize method

rate signature(object = "Contsimulation"): returns the contamination rate

rate<- signature(object = "Contsimulation"): changes the contamination rate

runs<- signature(object = "Contsimulation"): changes the number of runs

samplesize<- signature(object = "Contsimulation"): changes the size of the sample

savedata `signature(object = "Contsimulation")`: saves the simulation in the directory of R

simulate `signature(x = "Contsimulation")`: generates the random numbers for the simulation

plot `signature(x = "Contsimulation")`: produces a plot of the real data matrix

print `signature(x = "Contsimulation")`: returns filename, seed, number of runs, the size of the sample, the rate and the distributions

summary `signature(object = "Contsimulation")`: returns filename, seed, number of runs, the size of the sample, the rate and a statistical summary for each run of the real data

Note

Changing distributions, seed, runs, samplesize or rate deletes possibly simulated data, as it would not fit to the new parameters.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Dataclass-class` `Simulation-class` `savedata-methods` `plot-methods` `simulate-methods` `summary-methods`

Examples

```
N <- Norm() # N is a standard normal distribution.
C <- Cauchy() # C is a Cauchy distribution
cs <- Contsimulation(filename = "csim",
                     runs = 10,
                     samplesize = 3,
                     seed = setRNG(),
                     distribution.id = N,
                     distribution.c = C,
                     rate = 0.1)
simulate(cs)
# Each of the 30 random numbers is ideal (N-distributed) with
# probability 0.9 and contaminated (C-distributed) with
# probability = 0.1
summary(cs)
Data(cs) # different data
savedata(cs) # saves the object in the working directory of R...
load("csim") # loads it again...
Data(cs) # ...without the data - use simulate to return it!
```

Data-methods *Methods for Function Data in Package 'distr'*

Description

Methods

Data `signature(object = "Dataclass")`: returns the data

Data<- `signature(object = "Dataclass")`: changes the data (does not work with a simulation or a contsimulation object)

Data<- `signature(object = "Simulation")`: ERROR: A change of the data is not allowed.

Data<- `signature(object = "Contsimulation")`: ERROR: A change of the data is not allowed.

Data.c-methods *Methods for Function Data.c in Package 'distr'*

Description

Data.c-methods

Methods

Data.c `signature(object = "Dataclass")`: returns the contaminated data

Data.id-methods *Methods for Function Data.id in Package 'distr'*

Description

Data.id-methods

Methods

Data.id `signature(object = "Contsimulation")`: returns the ideal data

`Dataclass-class` *Class "Dataclass"*

Description

In an object of type "Dataclass" data can be saved containing any number of runs in any dimension. All information about the data is stored in a unified way.

Objects from the Class

Objects can be created by calls of the form `Dataclass(filename, Data)`. A Dataclass-object includes, aside from the actual data, a filename and the number of runs and the size of the sample, which give the number of rows and columns of the data matrix.

Slots

`filename`: Object of class `"character"`: the filename the data shall be saved

`Data`: Object of class `"vectororNULL"`: the actual data, either of type "NULL" (means no data) or "vector"

`runs`: Object of class `"numeric"`: the number of runs of the data

`samplesize`: Object of class `"numeric"`: the size of the sample

Methods

Data `signature(object = "Dataclass")`: returns the actual data

Data<- `signature(object = "Dataclass")`: changes the data

evaluate `signature(object = "Dataclass", estimator = "function")`: creates an object of type "Evaluation", see there for further information

filename `signature(object = "Dataclass")`: returns the the filename

filename<- `signature(object = "Dataclass")`: changes the the filename

initialize `signature(.Object = "Dataclass")`: initialize method

runs `signature(object = "Dataclass")`: returns the number of runs

samplesize `signature(object = "Dataclass")`: returns the size of the sample

savedata `signature(object = "Dataclass")`: saves the object in the directory of R and also a copy without data

plot `signature(x = "Dataclass")`: produces a plot of the data matrix

print `signature(x = "Dataclass")`: returns filename, number of runs and the size of the sample

summary `signature(object = "Dataclass")`: returns the same information as print, moreover a statistical summary for each run

Note

The saved Dataclass can be loaded with the usual load-command, the saved comment with the function cload.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Simulation-class` `Contsimulation-class` `Evaluation-class` `plot-methods` `print-methods` `summary-methods` `load` `cload` `savedata-methods`

Examples

```
D <- Dataclass(Data = matrix(c(1,2,3,4,5,6),2),
               filename = "xyz.sav")
# A new object of type "Dataclass" is created.
savedata(D)
# creates a file called "xyz.sav" where the information is saved and a
# copy "xyz.sav.comment" without data
Data(D) <- matrix(c(11,12,13,14,15,16),2) # changes the data of D
cload("xyz.sav") # loads the object without data - it is called "D.comment"
D.comment
load("xyz.sav") # loads the original object "D"
Data(D) # the original data: matrix(c(1,2,3,4,5,6),2)
evaluate(object = D, estimator = mean) # returns the mean of each variable
```

`Dirac-class` *Class "Dirac"*

Description

The Dirac distribution with location l, by default $= 0$, has density $d(x) = 1$ for $x = l$, 0 else.

Objects from the Class

Objects can be created by calls of the form `Dirac(location)`. This object is a `Dirac` distribution.

Slots

`img`: Object of class `"Naturals"`: The space of the image of this distribution has got dimension 1 and the name "Real Space".

`param`: Object of class `"DiracParameter"`: the parameter of this distribution (location), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers

`d`: Object of class `"function"`: density function

`p`: Object of class `"function"`: cumulative function

`q`: Object of class `"function"`: inverse of the cumulative function

`support`: Object of class `"numeric"`: a (sorted) vector containing the support of the discrete density function

Extends

Class "DiscreteDistribution", directly.
Class "UnivariateDistribution", by class "DiscreteDistribution".
Class "Distribution", by class "DiscreteDistribution".

Methods

initialize signature(.Object = "Dirac"): initialize method

location signature(object = "Dirac"): returns the slot location of the parameter of the distribution

location<- signature(object = "Dirac"): modifies the slot location of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

DiracParameter-class DiscreteDistribution-class Naturals-class

Examples

```
D <- Dirac(location = 0) # D is a Dirac distribution with location=0.
r(D)(1)
# r(D)(1) generates a pseudo-random-number according to a Dirac
# distribution with location = 0,
# which of course will take 0 as value almost surely.
d(D)(0) # Density of this distribution is 1 for x = 0.
p(D)(1) # Probability that x < 1 is 1.
q(D)(.1) # q(D)(x) is always 0 (= location).
location(D) # location of this distribution is 0.
location(D) <- 2 # location of this distribution is now 2.
```

DiracParameter-class

Class "DiracParameter"

Description

The parameter of a Dirac distribution, used by Dirac-class

Objects from the Class

Objects can be created by calls of the form new("DiracParameter", location). Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Dirac is instantiated.

Slots

location: Object of class "numeric": the location of a Dirac distribution

name: Object of class "character": a name / comment for the parameters

Extends

Class "Parameter", directly.

Methods

initialize signature(.Object = "DiracParameter"): initialize method

location signature(object = "DiracParameter"): returns the slot location of the parameter of the distribution

location<- signature(object = "DiracParameter"): modifies the slot location of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

Dirac-class Parameter-class

Examples

```
W=new("DiracParameter",location=1)
location(W) # location of this distribution is 1.
location(W)=2 # location of this distribution is now 2.
```

DiscreteDistribution-class

Class "DiscreteDistribution"

Description

The DiscreteDistribution-class is the mother-class of the classes Binom, Dirac, Geom, Hyper, Nbinom and Poisson. Further discrete distributions can be defined either by declaration of own random number generator, density and cumulative distribution and quantile functions, or as result of a convolution of two discrete distributions or by application of a mathematical operator to a discrete distribution. An additional way is, to specify only the random number generator. The function RtoDPQ.d then approximates the three remaining slots d, p and q by random sampling.

Objects from the Class

Objects can be created by calls of the form new("DiscreteDistribution", r, d, p, q). The result of this call is a discrete distribution.

Slots

img: Object of class "Reals": the space of the image of this distribution which has dimension 1 and the name "Real Space"

param: Object of class "Parameter": the parameter of this distribution, having only the slot name "Parameter of a discrete distribution"

r: Object of class "function": generates random numbers

d: Object of class "function": density/probability function

p: Object of class "function": cumulative distribution function

q: Object of class "function": quantile function

support: Object of class "numeric": a (sorted) vector containing the support of the discrete density function

Extends

Class "UnivariateDistribution", directly.
Class "Distribution", by class "UnivariateDistribution".

Methods

initialize signature(.Object = "DiscreteDistribution"): initialize method

Math signature(x = "DiscreteDistribution"): application of a mathematical function, e.g. sin or exp (does not work with log!), to this discrete distribution

- signature(e1 = "DiscreteDistribution"): application of '-' to this discrete distribution

* signature(e1 = "DiscreteDistribution", e2 = "numeric"): multiplication of this discrete distribution by an object of class 'numeric'

/ signature(e1 = "DiscreteDistribution", e2 = "numeric"): division of this discrete distribution by an object of class 'numeric'

+ signature(e1 = "DiscreteDistribution", e2 = "numeric"): addition of this discrete distribution to an object of class 'numeric'

- signature(e1 = "DiscreteDistribution", e2 = "numeric"): subtraction of an object of class 'numeric' from this discrete distribution

* signature(e1 = "numeric", e2 = "DiscreteDistribution"): multiplication of this discrete distribution by an object of class 'numeric'

+ signature(e1 = "numeric", e2 = "DiscreteDistribution"): addition of this discrete distribution to an object of class 'numeric'

- signature(e1 = "numeric", e2 = "DiscreteDistribution"): subtraction of this discrete distribution from an object of class 'numeric'

+ signature(e1 = "DiscreteDistribution", e2 = "DiscreteDistribution"): Convolution of two discrete distributions. The slots p, d and q are approximated by grids.

- signature(e1 = "DiscreteDistribution", e2 = "DiscreteDistribution"): Convolution of two discrete distributions. The slots p, d and q are approximated by grids.

support signature(object = "DiscreteDistribution"): returns the support

plot signature(object = "DiscreteDistribution"): plots density, cumulative distribution and quantile function

Note

Working with a computer, we use a finite interval as support which carries at least mass $1 - TruncQuantile$.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Parameter-class` `UnivariateDistribution-class` `Binom-class` `Dirac-class` `Geom-class` `Hyper-class` `Nbinom-class` `Pois-class` `AbscontDistribution-class` `Reals-class` `RtoDPQ.d`

Examples

```
B = Binom(prob=0.1,size=10) # B is a Binomial distribution with prob=0.1 and size=10.
P = Pois(lambda=1) # P is a Poisson distribution with lambda=1.
D1 = B+1 # a new discrete distributions with exact slots d, p, q
D2 = D1*3 # a new discrete distributions with exact slots d, p, q
D3 = B+P # a new discrete distributions with approximated slots d, p, q
D4 = D1+P # a new discrete distributions with approximated slots d, p, q
support(D4) # the (approximated) support of this distribution is 1, 2, ..., 21
r(D4)(1) # one random number generated from this distribution, e.g. 4
d(D4)(1) # The (approximated) density for x=1 is 0.1282716.
p(D4)(1) # The (approximated) probability that x<=1 is 0.1282716.
q(D4)(.5) # The (approximated) 50 percent quantile is 3.
```

`Distribution-class` *Class "Distribution"*

Description

The `Distribution-class` is the mother-class of the class `UnivariateDistribution`.

Objects from the Class

Objects can be created by calls of the form `new("Distribution")`.

Slots

`img`: Object of class `"Parameter"`: the space of the image

`param`: Object of class `"function"`: the parameter

`r`: Object of class `"function"`: generates random numbers

`d`: Object of class `"OptionalFunction"`: density function

`p`: Object of class `"OptionalFunction"`: cumulative distribution function

`q`: Object of class `"OptionalFunction"`: quantile function

Methods

img `signature(object = "Distribution")`: returns the space of the image

param `signature(object = "Distribution")`: returns the parameter

r `signature(object = "Distribution")`: returns the random number generator

d `signature(object = "Distribution")`: returns the density function

p `signature(object = "Distribution")`: returns the cumulative distribution function

q `signature(object = "Distribution")`: returns the quantile function

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`UnivariateDistribution-class` `Parameter-class`

`EuclideanSpace-class`

Class "EuclideanSpace"

Description

The distribution-classes contain a slot where the sample space is stored. One typical sample space is the Euclidean Space in dimension k.

Objects from the Class

Objects could theoretically be created by calls of the form `new("EuclideanSpace", dimension, name)`. Usually an object of this class is not needed on its own. `EuclideanSpace` is the mother-class of the class `Reals`, which is generated automatically when a univariate absolutly continuous distribution is instantiated.

Slots

`dimension`: Object of class `"numeric"`: the dimension of the space, by default = 1

`name`: Object of class `"character"`: the name of the space, by default = "Euclidean Space"

Extends

Class `"rSpace"`, directly.

Methods

initialize `signature(.Object = "EuclideanSpace")`: initialize method

liesIn `signature(object = "EuclideanSpace", x = "numeric")`: Does a particular vector lie in this space or not?

dimension `signature(object = "EuclideanSpace")`: returns the dimension of the space

dimension<- `signature(object = "EuclideanSpace")`: modifies the dimension of the space

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`rSpace-class` `Reals-class` `Distribution-class` `liesIn-methods`

Examples

```
E=new("EuclideanSpace",dimension=2)
dimension(E) # The dimension of this space is 2.
dimension(E)=3 # The dimension of this space is now 3.
liesIn(E,c(0,0,0)) # TRUE
liesIn(E,c(0,0)) # FALSE
```

`Evaluation-class` *Class "Evaluation"*

Description

When an estimator is used to data of the type "Dataclass" with the method evaluate, the result is an object of class "Evaluation".

Objects from the Class

Objects could be created by calls of the form `new("Evaluation", name, filename, call.ev, result, estimator)`. It does not seem to be very useful to generate a new object this way, however. Use "evaluate" with a Dataclass object!

Slots

`name`: Object of class `"character"`: the name of the Dataclass object, which was called by evaluate

`filename`: Object of class `"character"`: the filename of the evaluated object

`call.ev`: Object of class `"call.ev"` the call which created the object, e.g. “evaluate(Dataclassobject)”

`result`: Object of class `"vector"`: the result of the evaluation of the estimation on data

`estimator`: Object of class `"OptionalFunction"`: estimation function used

Methods

call.ev `signature(object = "Evaluation")`: returns the call which created the object

estimator `signature(object = "Evaluation")`: returns the estimator

filename `signature(object = "Evaluation")`: returns the filename

initialize `signature(.Object = "Evaluation")`: initialize method

name `signature(object = "Evaluation")`: returns the name of the data object

result `signature(object = "Evaluation")`: returns the result

plot `signature(object = "Evaluation")`: returns a boxplot of the result

print `signature(object = "Evaluation")`: returns the name of the data object, its filename, the estimator used and the result

savedata `signature(object = "Evaluation")`: saves the object in two files in the directory of R - one with data, one without as comment file (see example)

summary `signature(object = "Evaluation")`: returns the name of the data object, its filename, the estimator used and a statistical summary of the result

Note

The saved "evaluation" can be loaded with the usual load-command, the saved comment with the function cload.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Dataclass-class` `Simulation-class` `Contsimulation-class` `load` `cload` `savedata-methods` `plot-methods` `simulate-methods` `summary-methods`

Examples

```
N <- Norm() # N is a standard normal distribution.
C <- Cauchy() # C is a Cauchy distribution
cs <- Contsimulation(filename = "csim",
                     runs = 5,
                     samplesize=5000,
                     seed=setRNG(),
                     distribution.id = N,
                     distribution.c = C,
                     rate = 0.1)
simulate(cs)
# Each of the 25000 random numbers is ideal (N-distributed) with
# probability 0.9 and contaminated (C-distributed) with probability = 0.1
summary(cs)
ev1 <- evaluate(cs, mean) # estimates the data with mean
ev1 # bad results
ev2 <- evaluate(cs,median) # estimates the data with median
ev2 # better results because median is robust
```

```
savedata(ev1)
# saves the evaluation with result as "csim.mean" and without result as
# "csim.mean.comment" in the working directory # of R - "csim" is the
# filename of the Contsimulation object, mean the name of the estimator
rm(ev1)
cload("csim.mean")
# loads the evaluation without result - the object is called ev1.comment
ev1.comment
load("csim.mean") # loads the evaluation with result
ev1
```

`Exp-class` *Class "Exp"*

Description

The exponential distribution with rate λ has density

$$f(x) = \lambda e^{-\lambda x}$$

for $x \geq 0$.

C.f. `rexp`

Objects from the Class

Objects can be created by calls of the form `Exp(rate)`. This object is a exponential distribution.

Slots

`img`: Object of class `"Reals"`: The space of the image of this distribution has got dimension 1 and the name "Real Space".

`param`: Object of class `"ExpParameter"`: the parameter of this distribution (rate), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function rexp)

`d`: Object of class `"function"`: density function (calls function dexp)

`p`: Object of class `"function"`: cumulative function (calls function pexp)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function qexp)

Extends

Class `"AbscontDistribution"`, directly. Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`. Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

initialize `signature(.Object = "Exp")`: initialize method

rate `signature(object = "Exp")`: returns the slot rate of the parameter of the distribution

rate<- `signature(object = "Exp")`: modifies the slot rate of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`ExpParameter-class` `AbscontDistribution-class` `Reals-class` `rexp`

Examples

```
E <- Exp(rate = 1) # E is a exp distribution with rate = 1.
r(E)(1) # one random number generated from this distribution, e.g. 0.4190765
d(E)(1) # Density of this distribution is 0.3678794 for x = 1.
p(E)(1) # Probability that x < 1 is 0.6321206.
q(E)(.1) # Probability that x < 0.1053605 is 0.1.
rate(E) # rate of this distribution is 1.
rate(E) <- 2 # rate of this distribution is now 2.
```

`ExpParameter-class` *Class "ExpParameter"*

Description

The parameter of an exponential distribution, used by Exp-class

Objects from the Class

Objects can be created by calls of the form `new("ExpParameter", rate)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Exp is instantiated.

Slots

`rate`: Object of class `"numeric"`: the rate of an exponential distribution

`name`: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "ExpParameter")`: initialize method

rate `signature(object = "ExpParameter")`: returns the slot `rate` of the parameter of the distribution

rate<- `signature(object = "ExpParameter")`: modifies the slot `rate` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Exp-class` `Parameter-class`

Examples

```
W=new("ExpParameter", rate=1)
rate(W) # rate of this distribution is 1.
rate(W)=2 # rate of this distribution is now 2.
```

FParameter-class *Class "FParameter"*

Description

The parameter of a F distribution, used by Fd-class

Objects from the Class

Objects can be created by calls of the form `new("FParameter", df1, df2)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Fd is instantiated.

Slots

`df1`: Object of class `"numeric"`: the df1 of a F distribution

`df2`: Object of class `"numeric"`: the df2 of a F distribution

`name`: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "FParameter")`: initialize method

df1 `signature(object = "FParameter")`: returns the slot df1 of the parameter of the distribution

df1<- `signature(object = "FParameter")`: modifies the slot df1 of the parameter of the distribution

df2 `signature(object = "FParameter")`: returns the slot df2 of the parameter of the distribution

df2<- `signature(object = "FParameter")`: modifies the slot df2 of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Fd-class` `Parameter-class`

Examples

```
W <- new("FParameter", df1 = 1, df2 = 1)
df2(W) # df2 of this distribution is 1.
df2(W) <- 2 # df2 of this distribution is now 2.
```

`Fd-class` *Class "Fd"*

Description

The F distribution with `df1` $= n_1$, by default `= 1`, and `df2` $= n_2$, by default `= 1`, degrees of freedom has density

$$d(x) = \frac{\Gamma(n_1/2 + n_2/2)}{\Gamma(n_1/2)\Gamma(n_2/2)} \left(\frac{n_1}{n_2}\right)^{n_1/2} x^{n_1/2-1} \left(1 + \frac{n_1 x}{n_2}\right)^{-(n_1+n_2)/2}$$

for $x > 0$.

C.f. `rf`

Objects from the Class

Objects can be created by calls of the form `Fd(df1, df2)`. This object is a F distribution.

Slots

`img`: Object of class `"Reals"`: The space of the image of this distribution has got dimension 1 and the name "Real Space".

`param`: Object of class `"FParameter"`: the parameter of this distribution (df1 and df2), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function rf)

`d`: Object of class `"function"`: density function (calls function df)

`p`: Object of class `"function"`: cumulative function (calls function pf)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function qf)

Extends

Class `"AbscontDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`.
Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

initialize `signature(.Object = "Fd")`: initialize method

df1 `signature(object = "Fd")`: returns the slot `df1` of the parameter of the distribution

df1<- `signature(object = "Fd")`: modifies the slot `df1` of the parameter of the distribution

df2 `signature(object = "Fd")`: returns the slot `df2` of the parameter of the distribution

df2<- `signature(object = "Fd")`: modifies the slot `df2` of the parameter of the distribution

Note

It is the distribution of the ratio of the mean squares of n1 and n2 independent standard normals, and hence of the ratio of two independent chi-squared variates each divided by its degrees of freedom. Since the ratio of a normal and the root mean-square of m independent normals has a Student's t_m distribution, the square of a t_m variate has a F distribution on 1 and m degrees of freedom.

The non-central F distribution is again the ratio of mean squares of independent normals of unit variance, but those in the numerator are allowed to have non-zero means and ncp is the sum of squares of the means.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`FParameter-class` `AbscontDistribution-class` `Reals-class` `rf`

Examples

```
F=Fd(df1=1,df2=1) # F is a F distribution with df=1 and df2=1.
r(F)(1) # one random number generated from this distribution, e.g. 29.37863
d(F)(1) # Density of this distribution is 0.1591549 for x=1 .
p(F)(1) # Probability that x<1 is 0.5.
q(F)(.1) # Probability that x<0.02508563 is 0.1.
df1(F) # df1 of this distribution is 1.
df1(F)=2 # df1 of this distribution is now 2.
```

GammaParameter-class

Class "GammaParameter"

Description

The parameter of a gamma distribution, used by `Gammad`-class

Objects from the Class

Objects can be created by calls of the form `new("GammaParameter", shape, scale)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class `Gammad` is instantiated.

Slots

shape: Object of class `"numeric"`: the shape of a Gamma distribution

scale: Object of class `"numeric"`: the scale of a Gamma distribution

name: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "GammaParameter")`: initialize method

scale `signature(object = "GammaParameter")`: returns the slot `scale` of a parameter of a Gamma distribution

scale<- `signature(object = "GammaParameter")`: modifies the slot `scale` of a parameter of a Gamma distribution

shape `signature(object = "GammaParameter")`: returns the slot `shape` of a parameter of a Gamma distribution

shape<- `signature(object = "GammaParameter")`: modifies the slot `shape` of a parameter of a Gamma distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Gammad-class` `Parameter-class`

Examples

```
W=new("GammaParameter",scale=1,shape=1)
shape(W) # shape of this distribution is 1.
shape(W)=2 # shape of this distribution is now 2.
```

`Gammad-class` *Class "Gammad"*

Description

The Gammad distribution with parameters `shape` $= \alpha$, by default `= 1`, and `scale` $= \sigma$, by default `= 1`, has density

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

for $x > 0$, $\alpha > 0$ and $\sigma > 0$. The mean and variance are $E(X) = \alpha\sigma$ and $Var(X) = \alpha\sigma^2$. C.f. `rgamma`

Objects from the Class

Objects can be created by calls of the form `Gammad(scale, shape)`. This object is a gamma distribution.

Slots

`img`: Object of class `"Reals"`: The space of the image of this distribution has got dimension 1 and the name "Real Space".

`param`: Object of class `"GammaParameter"`: the parameter of this distribution (scale and shape), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function rgamma)

`d`: Object of class `"function"`: density function (calls function dgamma)

`p`: Object of class `"function"`: cumulative function (calls function pgamma)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function qgamma)

Extends

Class `"AbscontDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`.
Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

initialize `signature(.Object = "Gammad")`: initialize method

scale `signature(object = "Gammad")`: returns the slot `scale` of the parameter of the distribution

scale<- `signature(object = "Gammad")`: modifies the slot `scale` of the parameter of the distribution

shape `signature(object = "Gammad")`: returns the slot `shape` of the parameter of the distribution

shape<- `signature(object = "Gammad")`: modifies the slot `shape` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`GammaParameter-class` `AbscontDistribution-class` `Reals-class` `rgamma`

Examples

```
G=Gammad(scale=1,shape=1) # G is a gamma distribution with scale=1 and shape=1.
r(G)(1) # one random number generated from this distribution, e.g. 0.1304441
d(G)(1) # Density of this distribution is 0.3678794 for x=1.
p(G)(1) # Probability that x<1 is 0.6321206.
q(G)(.1) # Probability that x<0.1053605 is 0.1.
scale(G) # scale of this distribution is 1.
scale(G)=2 # scale of this distribution is now 2.
```

`Geom-class` *Class "Geom"*

Description

The geometric distribution with `prob` $= p$ has density

$$p(x) = p(1-p)^x$$

for $x = 0, 1, 2, \ldots$

C.f. `rgeom`

Objects from the Class

Objects can be created by calls of the form `Geom(prob)`. This object is a geometric distribution.

Slots

`img`: Object of class `"Naturals"`: The space of the image of this distribution has got dimension 1 and the name "Natural Space".

`param`: Object of class `"GeomParameter"`: the parameter of this distribution (prob), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function rgeom)

`d`: Object of class `"function"`: density function (calls function dgeom)

`p`: Object of class `"function"`: cumulative function (calls function pgeom)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function qgeom). The quantile is defined as the smallest value x such that $F(x) \geq p$, where F is the distribution function.

`support`: Object of class `"numeric"`: a (sorted) vector containing the support of the discrete density function

Extends

Class "DiscreteDistribution", directly. Class "UnivariateDistribution", by class "DiscreteDistribution". Class "Distribution", by class "DiscreteDistribution".

Methods

initialize signature(.Object = "Geom"): initialize method

prob signature(object = "Geom"): returns the slot prob of the parameter of the distribution

prob<- signature(object = "Geom"): modifies the slot prob of the parameter of the distribution

Note

Working with a computer, we use a finite interval as support which carries at least mass (1-TruncQuantile).

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

GeomParameter-class DiscreteDistribution-class Naturals-class rgeom

Examples

```
G <- Geom(prob = 0.5) # G is a geometric distribution with prob = 0.5.
r(G)(1) # one random number generated from this distribution, e.g. 0
d(G)(1) # Density of this distribution is 0.25 for x = 1.
p(G)(1) # Probability that x<1 is 0.75.
q(G)(.1) # x = 0 is the smallest value x such that p(G)(x) >= 0.1.
prob(G) # prob of this distribution is 0.5.
prob(G) <- 0.6 # prob of this distribution is now 0.6.
```

GeomParameter-class

Class "GeomParameter"

Description

The parameter of a geometric distribution, used by Geom-class

Objects from the Class

Objects can be created by calls of the form new("GeomParameter", prob). Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Geom is instantiated.

Slots

prob: Object of class `"numeric"`: the probability of a geometric distribution

name: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "GeomParameter")`: initialize method

prob `signature(object = "GeomParameter")`: returns the slot `prob` of the parameter of the distribution

prob<- `signature(object = "GeomParameter")`: modifies the slot `prob` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Geom-class` `Parameter-class`

Examples

```
W=new("GeomParameter",prob=0.5)
prob(W) # prob of this distribution is 0.5.
prob(W)=0.4 # prob of this distribution is now 0.4.
```

`Hyper-class` *Class "Hyper"*

Description

The hypergeometric distribution is used for sampling *without* replacement. The density of this distribution with parameters `m`, `n` and `k` (named Np, $N - Np$, and n, respectively in the reference below) is given by

$$p(x) = \binom{m}{x}\binom{n}{k-x} \Big/ \binom{m+n}{k}$$

for $x = 0, \ldots, k$. C.f. `rhyper`

Objects from the Class

Objects can be created by calls of the form `Hyper(m, n, k)`. This object is a hypergeometric distribution.

Slots

img: Object of class `"Naturals"`: The space of the image of this distribution has got dimension 1 and the name "Natural Space".

param: Object of class `"HyperParameter"`: the parameter of this distribution (`m`, `n`, `k`), declared at its instantiation

r: Object of class `"function"`: generates random numbers (calls function `rhyper`)

d: Object of class `"function"`: density function (calls function `dhyper`)

p: Object of class `"function"`: cumulative function (calls function `phyper`)

q: Object of class `"function"`: inverse of the cumulative function (calls function `qhyper`). The α-quantile is defined as the smallest value x such that $p(x) \geq \alpha]$, where p is the cumulative function.

support: Object of class `"numeric"`: a (sorted) vector containing the support of the discrete density function

Extends

Class `"DiscreteDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"DiscreteDistribution"`.
Class `"Distribution"`, by class `"DiscreteDistribution"`.

Methods

initialize `signature(.Object = "Hyper")`: initialize method

m `signature(object = "Hyper")`: returns the slot `m` of the parameter of the distribution

m<- `signature(object = "Hyper")`: modifies the slot `m` of the parameter of the distribution

n `signature(object = "Hyper")`: returns the slot `n` of the parameter of the distribution

n<- `signature(object = "Hyper")`: modifies the slot `n` of the parameter of the distribution

k `signature(object = "Hyper")`: returns the slot `k` of the parameter of the distribution

k<- `signature(object = "Hyper")`: modifies the slot `k` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`HyperParameter-class` `DiscreteDistribution-class` `Naturals-class` `rhyper`

Examples

```
H=Hyper(m=3,n=3,k=3) # H is a hypergeometric distribution with m=3,n=3,k=3.
r(H)(1) # one random number generated from this distribution, e.g. 2
d(H)(1) # Density of this distribution is  0.45 for x=1.
p(H)(1) # Probability that x<1 is 0.5.
q(H)(.1) # x=1 is the smallest value x such that p(H)(x)>=0.1.
m(H) # m of this distribution is 3.
m(H)=2 # m of this distribution is now 2.
```

HyperParameter-class

Class "HyperParameter"

Description

The parameter of a hypergeometric distribution, used by `Hyper`-class

Objects from the Class

Objects can be created by calls of the form `new("HyperParameter", k, m, n)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class `Hyper` is instantiated.

Slots

k: Object of class `"numeric"`: k of a hypergeometric distribution

m: Object of class `"numeric"`: m of a hypergeometric distribution

n: Object of class `"numeric"`: n of a hypergeometric distribution

name: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "HyperParameter")`: initialize method

k `signature(object = "HyperParameter")`: returns the slot `k` of the parameter of the distribution

k<- `signature(object = "HyperParameter")`: modifies the slot `k` of the parameter of the distribution

m `signature(object = "HyperParameter")`: returns the slot `m` of the parameter of the distribution

m<- `signature(object = "HyperParameter")`: modifies the slot `m` of the parameter of the distribution

n `signature(object = "HyperParameter")`: returns the slot `n` of the parameter of the distribution

n<- `signature(object = "HyperParameter")`: modifies the slot `n` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Hyper-class` `Parameter-class`

Examples

```
W=new("HyperParameter",k=3, m=3, n=3)
m(W) # m of this distribution is 3.
m(W)=2 # m of this distribution is now 2.
```

`Lnorm-class` *Class "Lnorm"*

Description

The log normal distribution has density

$$d(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-(\log(x)-\mu)^2/2\sigma^2}$$

where μ, by default $= 0$, and σ, by default $= 1$, are the mean and standard deviation of the logarithm. C.f. `rlnorm`

Objects from the Class

Objects can be created by calls of the form `Lnorm(meanlog, sdlog)`. This object is a log normal distribution.

Slots

`img`: Object of class `"Reals"`: The space of the image of this distribution has got dimension 1 and the name "Real Space".

`param`: Object of class `"LnormParameter"`: the parameter of this distribution (meanlog and sdlog), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function `rlnorm`)

`d`: Object of class `"function"`: density function (calls function `dlnorm`)

`p`: Object of class `"function"`: cumulative function (calls function `plnorm`)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function `qlnorm`)

Extends

Class `"AbscontDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`.
Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

initialize `signature(.Object = "Lnorm")`: initialize method

meanlog `signature(object = "Lnorm")`: returns the slot `meanlog` of the parameter of the distribution

meanlog<- `signature(object = "Lnorm")`: modifies the slot `meanlog` of the parameter of the distribution

sdlog `signature(object = "Lnorm")`: returns the slot `sdlog` of the parameter of the distribution

sdlog<- `signature(object = "Lnorm")`: modifies the slot `sdlog` of the parameter of the distribution

Note

The mean is $E(X) = exp(\mu + 1/2\sigma^2)$, and the variance $Var(X) = exp(2\mu + \sigma^2)(exp(\sigma^2) - 1)$ and hence the coefficient of variation is $\sqrt{exp(\sigma^2) - 1}$ which is approximately σ when that is small (e.g., $\sigma < 1/2$).

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`LnormParameter-class` `AbscontDistribution-class` `Reals-class` `rlnorm`

Examples

```
L=Lnorm(meanlog=1,sdlog=1) # L is a lnorm distribution with mean=1 and sd=1.
r(L)(1) # one random number generated from this distribution, e.g. 3.608011
d(L)(1) # Density of this distribution is 0.2419707 for x=1.
p(L)(1) # Probability that x<1 is 0.1586553.
q(L)(.1) # Probability that x<0.754612 is 0.1.
meanlog(L) # meanlog of this distribution is 1.
meanlog(L)=2 # meanlog of this distribution is now 2.
```

LnormParameter-class

Class "LnormParameter"

Description

The parameter of a log normal distribution, used by Lnorm-class

Objects from the Class

Objects can be created by calls of the form `new("LnormParameter", meanlog, sdlog)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Lnorm is instantiated.

Slots

meanlog: Object of class `"numeric"`: the mean of a log normal distribution

sdlog: Object of class `"numeric"`: the sd of a log normal distribution

name: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "LnormParameter")`: initialize method

sdlog `signature(object = "LnormParameter")`: returns the slot sdlog of the parameter of the distribution

sdlog<- `signature(object = "LnormParameter")`: modifies the slot sdlog of the parameter of the distribution

meanlog `signature(object = "LnormParameter")`: returns the slot meanlog of the parameter of the distribution

meanlog<- `signature(object = "LnormParameter")`: modifies the slot meanlog of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Lnorm-class` `Parameter-class`

Examples

```
W=new("LnormParameter",sdlog=1,meanlog=0)
meanlog(W) # meanlog of this distribution is 0.
meanlog(W)=2 # meanlog of this distribution is now 2.
```

`Logis-class` *Class "Logis"*

Description

The Logistic distribution with `location` $= \mu$, by default `= 0`, and `scale` $= \sigma$, by default `= 1`, has distribution function

$$p(x) = \frac{1}{1 + e^{-(x-\mu)/\sigma}}$$

and density

$$d(x) = \frac{1}{\sigma} \frac{e^{(x-\mu)/\sigma}}{(1 + e^{(x-\mu)/\sigma})^2}$$

It is a long-tailed distribution with mean μ and variance $\pi^2/3\sigma^2$. C.f. `rlogis`

Objects from the Class

Objects can be created by calls of the form `Logis(location, scale)`. This object is a logistic distribution.

Slots

`img`: Object of class `"Reals"`: The space of the image of this distribution has got dimension 1 and the name "Real Space".

`param`: Object of class `"LogisParameter"`: the parameter of this distribution (location and scale), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function rlogis)

`d`: Object of class `"function"`: density function (calls function dlogis)

`p`: Object of class `"function"`: cumulative function (calls function plogis)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function qlogis)

Extends

Class `"AbscontDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`.
Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

initialize `signature(.Object = "Logis")`: initialize method

location `signature(object = "Logis")`: returns the slot `location` of the parameter of the distribution

location<- `signature(object = "Logis")`: modifies the slot `location` of the parameter of the distribution

scale `signature(object = "Logis")`: returns the slot `scale` of the parameter of the distribution

scale<- `signature(object = "Logis")`: modifies the slot `scale` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`LogisParameter-class` `AbscontDistribution-class` `Reals-class` `rlogis`

Examples

```
L <- Logis(location = 1,scale = 1)
# L is a logistic distribution with  location = 1 and scale = 1.
r(L)(1) # one random number generated from this distribution, e.g. 5.87557
d(L)(1) # Density of this distribution is 0.25 for x = 1.
p(L)(1) # Probability that x < 1 is 0.5.
```

```
q(L)(.1) # Probability that x < -1.197225 is 0.1.
location(L) # location of this distribution is 1.
location(L) <- 2 # location of this distribution is now 2.
```

`LogisParameter-class`
Class "LogisParameter"

Description

The parameter of a logistic distribution, used by `Logis`-class

Objects from the Class

Objects can be created by calls of the form `new("LogisParameter", scale, location)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class `Logis` is instantiated.

Slots

`scale`: Object of class `"numeric"`: the scale of a logistic distribution

`location`: Object of class `"numeric"`: the location of a logistic distribution

`name`: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "LogisParameter")`: initialize method

location `signature(object = "LogisParameter")`: returns the slot `location` of the parameter of the distribution

location`<- signature(object = "LogisParameter")`: modifies the slot `location` of the parameter of the distribution

scale `signature(object = "LogisParameter")`: returns the slot `scale` of the parameter of the distribution

scale`<- signature(object = "LogisParameter")`: modifies the slot `scale` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Logis-class` `Parameter-class`

Examples

```
W=new("LogisParameter",location=0,scale=1)
scale(W) # scale of this distribution is 1.
scale(W)=2 # scale of this distribution is now 2.
```

Math-methods *Methods for Function Math in Package 'distr'*

Description

Math-methods

Methods

Math `signature(x = "AbscontDistribution")`: application of a mathematical function, e.g. sin or exp (does not work with log!), to this absolutely continouos distribution

Math `signature(x = "AbscontDistribution")`: application of a mathematical function, e.g. sin or exp (does not work with log!), to this discrete distribution

---

Max-methods *Methods for Function Max in Package 'distr'*

Description

Max-methods

Methods

Max `signature(object = "UnifParameter")`: returns the slot `Max` of the parameter of the distribution

Max<- `signature(object = "UnifParameter")`: modifies the slot `Max` of the parameter of the distribution

Max `signature(object = "Unif")`: returns the slot `Max` of the parameter of the distribution

Max<- `signature(object = "Unif")`: modifies the slot `Max` of the parameter of the distribution

Min-methods *Methods for Function Min in Package 'distr'*

Description

Min-methods

Methods

Min `signature(object = "UnifParameter")`: returns the slot `Min` of the parameter of the distribution

Min<- `signature(object = "UnifParameter")`: modifies the slot `Min` of the parameter of the distribution

Min `signature(object = "Unif")`: returns the slot `Min` of the parameter of the distribution

Min<- `signature(object = "Unif")`: modifies the slot `Min` of the parameter of the distribution

Naturals-class *Class "Naturals"*

Description

The distribution-classes contain a slot where the sample space is stored. Typically, discrete random variables take naturals as values.

Objects from the Class

Objects could theoretically be created by calls of the form `new("Naturals", dimension, name)`. Usually an object of this class is not needed on its own. It is generated automatically when a univariate discrete distribution is instantiated.

Slots

`dimension`: Object of class `"character"`: the dimension of the space, by default = 1

`name`: Object of class `"character"`: the name of the space, by default = "Natural Space"

Extends

Class `"Reals"`, directly.
Class `"EuclideanSpace"`, by class `"Reals"`.
Class `"rSpace"`, by class `"Reals"`.

Methods

initialize `signature(.Object = "Naturals")`: initialize method

liesIn `signature(object = "Naturals", x = "numeric")`: Does a particular vector only contain naturals?

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Reals-class` `DiscreteDistribution-class`

Examples

```
N=new("Naturals")
liesIn(N,1) # TRUE
liesIn(N,c(0,1)) # FALSE
liesIn(N,0.1) # FALSE
```

`Nbinom-class` *Class "Nbinom"*

Description

The negative binomial distribution with `size` $= n$, by default $= 1$, and `prob` $= p$, by default $= 0.5$, has density

$$d(x) = \frac{\Gamma(x+n)}{\Gamma(n)x!}p^n(1-p)^x$$

for $x = 0, 1, 2, \ldots$

This represents the number of failures which occur in a sequence of Bernoulli trials before a target number of successes is reached. C.f. `rnbinom`

Objects from the Class

Objects can be created by calls of the form `Nbinom(prob, size)`. This object is a negative binomial distribution.

Slots

`img`: Object of class `"Naturals"`: The space of the image of this distribution has got dimension 1 and the name "Natural Space".

`param`: Object of class `"NbinomParameter"`: the parameter of this distribution (prob, size), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function `rnbinom`)

`d`: Object of class `"function"`: density function (calls function `dnbinom`)

`p`: Object of class `"function"`: cumulative function (calls function `pnbinom`)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function `qnbinom`). The quantile is defined as the smallest value x such that $F(x) \geq p$, where F is the distribution function.

`support`: Object of class `"numeric"`: a (sorted) vector containing the support of the discrete density function

Extends

Class `"DiscreteDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"DiscreteDistribution"`.
Class `"Distribution"`, by class `"DiscreteDistribution"`.

Methods

initialize `signature(.Object = "Nbinom")`: initialize method

prob `signature(object = "Nbinom")`: returns the slot `prob` of the parameter of the distribution

prob<- `signature(object = "Nbinom")`: modifies the slot `prob` of the parameter of the distribution

size `signature(object = "Nbinom")`: returns the slot `size` of the parameter of the distribution

size<- `signature(object = "Nbinom")`: modifies the slot `size` of the parameter of the distribution

Note

Working with a computer, we use a finite interval as support which carries at least mass $1 - TruncQuantile$.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`NbinomParameter-class` `DiscreteDistribution-class` `Naturals-class` `rnbinom`

Examples

```
N=Nbinom(prob=0.5,size=1) # N is a binomial distribution with prob=0.5 and size=1.
r(N)(1) # one random number generated from this distribution, e.g. 3
d(N)(1) # Density of this distribution is  0.25 for x=1.
p(N)(0.4) # Probability that x<0.4 is 0.5.
q(N)(.1) # x=0 is the smallest value x such that p(B)(x)>=0.1.
size(N) # size of this distribution is 1.
size(N)=2 # size of this distribution is now 2.
```

`NbinomParameter-class`
Class "NbinomParameter"

Description

The parameter of a negative binomial distribution, used by Nbinom-class

Objects from the Class

Objects can be created by calls of the form `new("NbinomParameter", prob, size)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Nbinom is prepared.

Slots

prob: Object of class `"numeric"`: the probability of a negative binomial distribution
size: Object of class `"numeric"`: the size of a negative binomial distribution
name: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "NbinomParameter")`: initialize method
prob `signature(object = "NbinomParameter")`: returns the slot `prob` of the parameter of the distribution
prob<- `signature(object = "NbinomParameter")`: modifies the slot `prob` of the parameter of the distribution
size `signature(object = "NbinomParameter")`: returns the slot `size` of the parameter of the distribution
size<- `signature(object = "NbinomParameter")`: modifies the slot `size` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Nbinom-class` `Parameter-class`

Examples

```
W=new("NbinomParameter",prob=0.5,size=1)
size(W) # size of this distribution is 1.
size(W)=2 # size of this distribution is now 2.
```

`Norm-class` *Class "Norm"*

Description

The normal distribution has density

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}$$

where μ is the mean of the distribution and σ the standard deviation. C.f. `rnorm`

Objects from the Class

Objects can be created by calls of the form `Norm(mean, sd)`. This object is a normal distribution.

Slots

`img`: Object of class `"Reals"`: The domain of this distribution has got dimension 1 and the name "Real Space".

`param`: Object of class `"UniNormParameter"`: the parameter of this distribution (mean and sd), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function `rnorm`)

`d`: Object of class `"function"`: density function (calls function `dnorm`)

`p`: Object of class `"function"`: cumulative function (calls function `pnorm`)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function `qnorm`)

Extends

Class `"AbscontDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`.
Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

`* signature(e1 = "numeric", e2 = "Norm")`: multiplication of this normal distribution by an object of class 'numeric'

`+ signature(e1 = "numeric", e2 = "Norm")`: addition of this normal distribution to an object of class 'numeric'

`- signature(e1 = "numeric", e2 = "Norm")`: subtraction of this normal distribution from an object of class 'numeric'

`* signature(e1 = "Norm", e2 = "numeric")`: multiplication of this normal distribution by an object of class 'numeric'

`+ signature(e1 = "Norm", e2 = "numeric")`: addition of this normal distribution to an object of class 'numeric'

`- signature(e1 = "Norm", e2 = "numeric")`: subtraction of an object of class 'numeric' from this normal distribution

`/ signature(e1 = "Norm", e2 = "numeric")`: division of this normal distribution by an object of class 'numeric'

`- signature(e1 = "Norm", e2 = "Norm")`

`+ signature(e1 = "Norm", e2 = "Norm")`: For the normal distribution the exact convolution formulas are implemented thereby improving the general numerical approximation.

initialize `signature(.Object = "Norm")`: initialize method

mean `signature(object = "Norm")`: returns the slot `mean` of the parameter of the distribution

mean<- `signature(object = "Norm")`: modifies the slot `mean` of the parameter of the distribution

sd `signature(object = "Norm")`: returns the slot `sd` of the parameter of the distribution

sd<- `signature(object = "Norm")`: modifies the slot `sd` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`UniNormParameter-class` `AbscontDistribution-class` `Reals-class` `rnorm`

Examples

```
N=Norm(mean=1,sd=1) # N is a normal distribution with mean=1 and sd=1.
r(N)(1) # one random number generated from this distribution, e.g. 2.257783
d(N)(1) # Density of this distribution is  0.3989423 for x=1.
p(N)(1) # Probability that x<1 is 0.5.
q(N)(.1) # Probability that x<-0.2815516 is 0.1.
mean(N) # mean of this distribution is 1.
sd(N)=2 # sd of this distribution is now 2.
M=Norm() # M is a normal distribution with mean=0 and sd=1.
O=M+N # O is a normal distribution with mean=1 (=1+0) and sd=sqrt(5) (=sqrt(2^2+1^2)).
```

NormParameter-class

Class "NormParameter"

Description

The parameter of a normal distribution, used by Norm-class

Objects from the Class

Objects can be created by calls of the form `new("NormParameter", sd, mean)`. Usually an object of this class is not needed on its own. It is the mother-class of the class UniNormParameter, which is generated automatically when such a distribution is instantiated.

Slots

sd: Object of class `"numeric"`: the sd of a normal distribution

mean: Object of class `"numeric"`: the mean of a normal distribution

name: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "NormParameter")`: initialize method

mean `signature(object = "NormParameter")`: returns the slot mean of the parameter of the distribution

mean<- `signature(object = "NormParameter")`: modifies the slot mean of the parameter of the distribution

sd `signature(object = "NormParameter")`: returns the slot sd of the parameter of the distribution

sd<- `signature(object = "NormParameter")`: modifies the slot sd of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Norm-class` `Parameter-class`

Examples

```
W <- new("NormParameter", mean = 0, sd = 1)
sd(W) # sd of this distribution is 1.
sd(W) <- 2 # sd of this distribution is now 2.
```

`OptionalParameter-class`

Class "OptionalParameter"

Description

auxiliary class; may contain either a Parameter or NULL, cf. J. Chambers, "green book".

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "OptionalParameter" in the signature.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

`Parameter-class` *Class "Parameter"*

Description

Parameter is the mother-class of all Parameter classes.

Objects from the Class

Objects can be created by calls of the form `new("Parameter")`.

Slots

`name:` Object of class `"character"`: a name / comment for the parameters

Methods

name `signature(object = "Parameter")`: returns the name of the parameter

name<- `signature(object = "Parameter")`: modifies the name of the parameter

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Distribution-class`

`Pois-class` *Class "Pois"*

Description

The Poisson distribution has density

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

for $x = 0, 1, 2, \ldots$. The mean and variance are $E(X) = Var(X) = \lambda$.

C.f. `rpois`

Objects from the Class

Objects can be created by calls of the form `Pois(lambda)`. This object is a Poisson distribution.

Slots

`img`: Object of class `"Naturals"`: The space of the image of this distribution has got dimension 1 and the name "Natural Space".

`param`: Object of class `"PoisParameter"`: the parameter of this distribution (lambda), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function rpois)

`d`: Object of class `"function"`: density function (calls function dpois)

`p`: Object of class `"function"`: cumulative function (calls function ppois)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function qpois). The quantile is defined as the smallest value x such that $F(x) \geq p$, where F is the distribution function.

`support`: Object of class `"numeric"`: a (sorted) vector containing the support of the discrete density function

Extends

Class `"DiscreteDistribution"`, directly. Class `"UnivariateDistribution"`, by class `"DiscreteDistribution"`. Class `"Distribution"`, by class `"DiscreteDistribution"`.

Methods

`+ signature(e1 = "Pois", e2 = "Pois")`: For the Poisson distribution the exact convolution formula is implemented thereby improving the general numerical approximation.

initialize `signature(.Object = "Pois")`: initialize method

lambda `signature(object = "Pois")`: returns the slot lambda of the parameter of the distribution

lambda<- `signature(object = "Pois")`: modifies the slot lambda of the parameter of the distribution

Note

Working with a computer, we use a finite interval as support which carries at least mass (1-TruncQuantile).

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`PoisParameter-class` `DiscreteDistribution-class` `Naturals-class` `rpois`

Examples

```
P <- Pois(lambda = 1) # P is a Poisson distribution with lambda = 1.
r(P)(1) # one random number generated from this distribution, e.g. 1
d(P)(1) # Density of this distribution is 0.3678794 for x = 1.
p(P)(0.4) # Probability that x < 0.4 is 0.3678794.
q(P)(.1) # x = 0 is the smallest value x such that p(B)(x) >= 0.1.
lambda(P) # lambda of this distribution is 1.
lambda(P) <- 2 # lambda of this distribution is now 2.
R <- Pois(lambda = 3) # R is a Poisson distribution with lambda = 2.
S <- P + R # R is a Poisson distribution with lambda = 5(=2+3).
```

PoisParameter-class

Class "PoisParameter"

Description

The parameter of a Poisson distribution, used by Pois-class

Objects from the Class

Objects can be created by calls of the form `new("PoisParameter", lambda)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Pois is prepared.

Slots

`lambda`: Object of class `"numeric"`: the lambda of a Poisson distribution

`name`: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "PoisParameter")`: initialize method

lambda `signature(object = "PoisParameter")`: returns the slot lambda of the parameter of the distribution

lambda<- `signature(object = "PoisParameter")`: modifies the slot lambda of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Pois-class` `Parameter-class`

Examples

```
W <- new("PoisParameter",lambda = 1)
lambda(W) # lambda of this distribution is 1.
lambda(W) <- 2 # lambda of this distribution is now 2.
```

`Reals-class` *Class "Reals"*

Description

Particular case of a one-dimensional Euclidean Space

Objects from the Class

Objects could theoretically be created by calls of the form `new("Reals", dimension, name)`. Usually an object of this class is not needed on its own. It is generated automatically when a univariate absolutly continuous distribution is instantiated.

Slots

`dimension`: Object of class `"character"`: the dimension of the space, by default = 1

`name`: Object of class `"character"`: the name of the space, by default = "Real Space"

Extends

Class `"EuclideanSpace"`, directly.
Class `"rSpace"`, by class `"EuclideanSpace"`.

Methods

initialize `signature(.Object = "Reals")`: initialize method

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

EuclideanSpace-class Naturals-class AbscontDistribution-class

Examples

```
R=new("Reals")
liesIn(R,c(0,0)) # FALSE
```

RtoDPQ *RtoDPQ*

Description

function to do get empirical density, cumulative distribution and quantile function from random numbers

Usage

```
RtoDPQ(r, e = RtoDPQ.e, n = DefaultNrGridPoints)
```

Arguments

r	the random number generator
e	10^e numbers are generated, a higher number leads to a better result.
n	The number of grid points used to create the approximated functions, a higher number leads to a better result.

Details

RtoDPQ generates 10^e random numbers, by default

$$e = RtoDPQ.e$$

. The density is formed on the basis of n points using approxfun and density, by default

$$n = DefaultNrGridPoints$$

. The cumulative distribution function and the quantile function are also created on the basis of n points using `approxfun` and `ecdf`. Of course, the results are usually not exact as they rely on random numbers.

Value

`RtoDPQ` returns a list of functions.

dfun	density
pfun	cumulative distribution function
qfun	quantile function

Note

Use `RtoDPQ` for absolutely continuous and `RtoDPQ.d` for discrete distributions.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`UnivariateDistribution-class`

Examples

```
rn2 <- function(n){rnorm(n)^2}
x <- RtoDPQ(r = rn2, e = 4, n = 512)
# returns density, cumulative distribution and quantile function of
# squared standard normal distribution
x$dfun(4)
RtoDPQ(r = rn2, e = 5, n = 1024) # for a better result

rp2 <- function(n){rpois(n, lambda = 1)^2}
x <- RtoDPQ.d(r = rp2, e = 5)
# returns density, cumulative distribution and quantile function of
# squared Poisson distribution with parameter lambda=1
```

`RtoDPQ.d` *RtoDPQ.d*

Description

function to do get empirical density, cumulative distribution and quantile function from random numbers

Usage

```
RtoDPQ.d(r, e = RtoDPQ.e)
```

Arguments

`r`	the random number generator
`e`	10^e numbers are generated, a higher number leads to a better result.

Details

`RtoDPQ.d` generates 10^e random numbers, by default $e =$ `RtoDPQ.e` which are used to produce a density, cdf and quantile function. Of course, the results are usually not exact as they rely on random numbers.

Value

RtoDPQ returns a list of functions.

`dfun`	density
`pfun`	cumulative distribution function
`qfun`	quantile function

Note

Use RtoDPQ for absolutely continuous and RtoDPQ.d for discrete distributions.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`UnivariateDistribution-class`

Examples

```
rn2 <- function(n){rnorm(n)^2}
x <- RtoDPQ(r = rn2, e = 4, n = 512)
# returns density, cumulative distribution and quantile function of
# squared standard  normal distribution

x$dfun(4)
RtoDPQ(r = rn2, e = 5, n = 1024) # for a better result

rp2 <- function(n){rpois(n, lambda = 1)^2}
x <- RtoDPQ.d(r = rp2, e = 5)
# returns density, cumulative distribution and quantile function of
# squared Poisson distribution with parameter lambda=1
```

`Simulation-class` *Class "Simulation"*

Description

In an object of type `Simulation` data can be simulated in any distribution and size.

Objects from the Class

Objects can be created by calls of the form `Simulation(filename, runs, samplesize, seed, distribution)`. A `Simulation`-object includes a filename, the number of runs, the size of the sample, the seed and the distribution of the random numbers. The slot `Data` stays empty until the method simulate has been used.

Slots

`seed`: Object of class `"list"`: the seed the simulation has been generated with

`distribution`: Object of class `"UnivariateDistribution"`: the distribution of the random numbers

`filename`: Object of class `"character"`: the filename the simulation shall be saved

`Data`: Object of class `"vectororNULL"`: the simulated data

`runs`: Object of class `"numeric"`: the number of runs of the data

`samplesize`: Object of class `"numeric"`: the size of the sample, so the dimension of the data

Extends

Class `"Dataclass"`, directly.

Methods

Data<- `signature(object = "Contsimulation")`: ERROR: A modification of simulated data is not allowed.

distribution `signature(object = "Simulation")`: returns the distribution

distribution<- `signature(object = "Simulation")`: changes the distribution

seed `signature(object = "Simulation")`: returns the seed

seed<- `signature(object = "Simulation")`: changes the seed

runs<- `signature(object = "Simulation")`: changes the number of runs

samplesize<- `signature(object = "Simulation")`: changes the size of the sample

savedata `signature(object = "Simulation")`: saves the object without the data in the directory of R (After loading the data can be reproduced by using simulate.)

initialize `signature(.Object = "Simulation")`: initialize method

plot `signature(x = "Simulation")`: produces a plot of the data matrix

print `signature(x = "Simulation")`: returns filename, seed, number of runs, the size of the sample and the distribution

simulate `signature(x = "Simulation")`: generates the random numbers for the simulation

summary `signature(object = "Simulation")`: returns filename, seed, number of runs, the size of the sample and a statistical summary for each run

Note

Changing distribution, seed, runs or samplesize deletes possibly simulated data, as it would not fit to the new parameters.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Dataclass-class` `Contsimulation-class` `plot-methods` `print-methods` `summary-methods` `simulate-methods`

Examples

```
N=Norm() # N is a standard normal distribution.
S=Simulation(filename="xyz",runs=10,samplesize=3,seed=setRNG(),distribution=N)
Data(S) # no data yet
simulate(S)
Data(S) # now there are random numbers
Data(S) # the same data as before because the seed has not changed
seed(S)=setRNG()
```

```
simulate(S)
Data(S) # different data
savedata(S) # saves the object in the directory of R...
load("xyz") # loads it again...
Data(S) # ...without the data - use simulate to return it!
```

`TParameter-class` *Class "TParameter"*

Description

The parameter of a t distribution, used by Td-class

Objects from the Class

Objects can be created by calls of the form `new("TParameter", df)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class `Td` is instantiated.

Slots

`df`: Object of class `"numeric"`: the df of a T distribution

`name`: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "TParameter")`: initialize method

df `signature(object = "TParameter")`: returns the slot `df` of the parameter of the distribution

df<- `signature(object = "TParameter")`: modifies the slot `df` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Td-class` `Parameter-class`

Examples

```
W=new("TParameter",df=1)
df(W) # df of this distribution is 1.
df(W)=2 # df of this distribution is now 2.
```

`Td-class` *Class "Td"*

Description

The t distribution with `df` $= \nu$ degrees of freedom has density

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

for all real x. It has mean 0 (for $\nu > 1$) and variance $\frac{\nu}{\nu-2}$ (for $\nu > 2$). C.f. `rt`

Objects from the Class

Objects can be created by calls of the form `Td(df)`. This object is a t distribution.

Slots

`img`: Object of class `"Reals"`: The domain of this distribution has got dimension 1 and the name "Real Space".

`param`: Object of class `"TParameter"`: the parameter of this distribution (df), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function `rt`)

`d`: Object of class `"function"`: density function (calls function `dt`)

`p`: Object of class `"function"`: cumulative function (calls function `pt`)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function `qt`)

Extends

Class `"AbscontDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`.
Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

initialize `signature(.Object = "Td")`: initialize method

df `signature(object = "Td")`: returns the slot df of the parameter of the distribution

df<- `signature(object = "Td")`: modifies the slot df of the parameter of the distribution

Note

The general *non-central* t with parameters (ν, δ) `= (df, ncp)` is defined as a the distribution of $T_\nu(\delta) := \frac{U+\delta}{\chi_\nu/\sqrt{\nu}}$ where U and χ_ν are independent random variables, $U \sim \mathcal{N}(0,1)$, and χ^2_ν is chi-squared, see `rchisq`.

The most used applications are power calculations for t-tests:
Let $T = \frac{\bar{X}-\mu_0}{S/\sqrt{n}}$ where $\bar{X}$ is the `mean` and S the sample standard deviation (`sd`) of $X_1, X_2, \ldots, X_n$ which are i.i.d. $N(\mu, \sigma^2)$. Then T is distributed as non-centrally t with `df`$= n - 1$ degrees of freedom and **n**on-**c**entrality **p**arameter `ncp`$= (\mu - \mu_0)\sqrt{n}/\sigma$.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`TParameter-class` `AbscontDistribution-class` `Reals-class` `rt`

Examples

```
T <- Td(df = 1) # T is a t distribution with df = 1.
r(T)(1) # one random number generated from this distribution, e.g. -0.09697573
d(T)(1) # Density of this distribution is 0.1591549 for x = 1.
p(T)(1) # Probability that x < 1 is 0.75.
q(T)(.1) # Probability that x < -3.077684 is 0.1.
df(T) # df of this distribution is 1.
df(T) <- 2 # df of this distribution is now 2.
```

UniNormParameter-class

Class "UniNormParameter"

Description

The parameter of a univariate normal distribution, used by Norm-class

Objects from the Class

Objects can be created by calls of the form `new("NormParameter", sd, mean)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Norm is instantiated.

Slots

sd: Object of class `"numeric"`: the sd of a univariate normal distribution

mean: Object of class `"numeric"`: the mean of a univariate normal distribution

name: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"NormParameter"`, directly. Class `"Parameter"`, by class `"NormParameter"`.

Methods

initialize `signature(.Object = "UniNormParameter")`: initialize method

mean `signature(object = "UniNormParameter")`: returns the slot mean of the parameter of the distribution

mean<- `signature(object = "UniNormParameter")`: modifies the slot mean of the parameter of the distribution

sd `signature(object = "UniNormParameter")`: returns the slot sd of the parameter of the distribution

sd<- `signature(object = "UniNormParameter")`: modifies the slot sd of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Norm-class` `NormParameter-class` `Parameter-class`

Examples

```
W <- new("UniNormParameter", mean = 0, sd = 1)
sd(W) # sd of this distribution is 1
sd(W) <- 2 # sd of this distribution is now 2
```

`Unif-class` *Class "Unif"*

Description

The uniform distribution has density

$$d(x) = \frac{1}{max - min}$$

for min, by default $= 0$, $\leq x \leq max$, by default $= 1$. C.f. `runif`

Objects from the Class

Objects can be created by calls of the form `Unif(Min, Max)`. This object is a uniform distribution.

Slots

`*` `signature(e1 = "Unif", e2 = "numeric")`: multiplication of this uniform distribution by an object of class 'numeric'

`+` `signature(e1 = "Unif", e2 = "numeric")`: addition of this uniform distribution to an object of class 'numeric'

`img`: Object of class `"Reals"`: The space of the image of this distribution has got dimension 1 and the name "Real Space".

`param`: Object of class `"UnifParameter"`: the parameter of this distribution (Min and Max), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function `runif`)

`d`: Object of class `"function"`: density function (calls function `dunif`)

`p`: Object of class `"function"`: cumulative function (calls function `punif`)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function `qunif`)

Extends

Class "AbscontDistribution", directly.
Class "UnivariateDistribution", by class "AbscontDistribution".
Class "Distribution", by class "AbscontDistribution".

Methods

initialize signature(.Object = "Unif"): initialize method

Min signature(object = "Unif"): returns the slot Min of the parameter of the distribution

Min<- signature(object = "Unif"): modifies the slot Min of the parameter of the distribution

Max signature(object = "Unif"): returns the slot Max of the parameter of the distribution

Max<- signature(object = "Unif"): modifies the slot Max of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

UnifParameter-class AbscontDistribution-class Reals-class runif

Examples

```
U=Unif(Min=0,Max=2) # U is a uniform distribution with Min=0 and Max=2.
r(U)(1) # one random number generated from this distribution, e.g. 1.984357
d(U)(1) # Density of this distribution is 0.5 for x=1.
p(U)(1) # Probability that x<1 is 0.5.
q(U)(.1) # Probability that x<0.2 is 0.1.
Min(U) # Min of this distribution is 0.
Min(U)=1 # Min of this distribution is now 1.
```

UnifParameter-class

Class "UnifParameter"

Description

The parameter of a uniform distribution, used by Unif-class

Objects from the Class

Objects can be created by calls of the form new("UnifParameter", Max, Min). Usually an object of this class is not needed on its own, it is generated automatically when an object of the class Unif is instantiated.

Slots

Max: Object of class `"numeric"`: the Max of a uniform distribution

Min: Object of class `"numeric"`: the Min of a uniform distribution

name: Object of class `"character"`: a name / comment for the parameters

Extends

Class `"Parameter"`, directly.

Methods

initialize `signature(.Object = "UnifParameter")`: initialize method

Min `signature(object = "UnifParameter")`: returns the slot `Min` of the parameter of the distribution

Min<- `signature(object = "UnifParameter")`: modifies the slot `Min` of the parameter of the distribution

Max `signature(object = "UnifParameter")`: returns the slot `Max` of the parameter of the distribution

Max<- `signature(object = "UnifParameter")`: modifies the slot `Max` of the parameter of the distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Unif-class` `Parameter-class`

Examples

```
W=new("UnifParameter",Min=0,Max=1)
Max(W) # Max of this distribution is 1.
Max(W)=2 # Max of this distribution is now 2.
```

`UnivariateDistribution-class`

Class "UnivariateDistribution"

Description

The UnivariateDistribution-class is the mother-class of the classes AbscontDistribution and DiscreteDistribution.

Objects from the Class

Objects can be created by calls of the form `new("UnivariateDistribution")`.

Slots

img: Object of class "Reals": the space of the image of this distribution which has dimension 1 and the name "Real Space"

param: Object of class "Parameter": the parameter of this distribution

r: Object of class "function": generates random numbers

d: Object of class "function": density function

p: Object of class "function": cumulative distribution function

q: Object of class "function": quantile function

Extends

Class "Distribution", directly.

Methods

initialize signature(.Object = "UnivariateDistribution"):
initialize method

\- signature(e1 = "UnivariateDistribution"):
application of '-' to this univariate distribution

* signature(e1 = "UnivariateDistribution", e2 = "numeric"):
multiplication of this univariate distribution by an object of class 'numeric'

/ signature(e1 = "UnivariateDistribution", e2 = "numeric"):
division of this univariate distribution by an object of class 'numeric'

\+ signature(e1 = "UnivariateDistribution", e2 = "numeric"):
addition of this univariate distribution to an object of class 'numeric'

\- signature(e1 = "UnivariateDistribution", e2 = "numeric"):
subtraction of an object of class 'numeric' from this univariate distribution

* signature(e1 = "numeric", e2 = "UnivariateDistribution"):
multiplication of this univariate distribution by an object of class 'numeric'

\+ signature(e1 = "numeric", e2 = "UnivariateDistribution"):
addition of this univariate distribution to an object of class 'numeric'

\- signature(e1 = "numeric", e2 = "UnivariateDistribution"):
subtraction of this univariate distribution from an object of class 'numeric'

\+ signature(e1 = "UnivariateDistribution", e2 = "UnivariateDistribution"):
Convolution of two univariate distributions. The slots p, d and q are approximated by grids.

\- signature(e1 = "UnivariateDistribution", e2 = "UnivariateDistribution"):
Convolution of two univariate distributions. The slots p, d and q are approximated by grids.

simplifyr signature(object = "UnivariateDistribution"):
simplifies the r-method of a distribution, see there for further information

print signature(object = "UnivariateDistribution"):
returns the class of the object and its parameters

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Parameter-class` `Distribution-class` `AbscontDistribution-class` `DiscreteDistribution-class` `Reals-class` `RtoDPQ` `simplifyr-methods`

`Weibull-class` *Class "Weibull"*

Description

The Weibull distribution with `shape` parameter a, by default $= 1$, and `scale` parameter σ has density given by, by default $= 1$,

$$d(x) = (a/\sigma)(x/\sigma)^{a-1}\exp(-(x/\sigma)^a)$$

for $x > 0$.

C.f. `rweibull`

Objects from the Class

Objects can be created by calls of the form `Weibull(shape, scale)`. This object is a Weibull distribution.

Slots

`img`: Object of class `"Reals"`: The space of the image of this distribution has got dimension 1 and the name "Real Space".

`param`: Object of class `"WeibullParameter"`: the parameter of this distribution (shape and scale), declared at its instantiation

`r`: Object of class `"function"`: generates random numbers (calls function `rweibull`)

`d`: Object of class `"function"`: density function (calls function `dweibull`)

`p`: Object of class `"function"`: cumulative function (calls function `pweibull`)

`q`: Object of class `"function"`: inverse of the cumulative function (calls function `qweibull`)

Extends

Class `"AbscontDistribution"`, directly.
Class `"UnivariateDistribution"`, by class `"AbscontDistribution"`.
Class `"Distribution"`, by class `"AbscontDistribution"`.

Methods

initialize `signature(.Object = "Weibull")`: initialize method

scale `signature(object = "Weibull")`: returns the slot `scale` of the parameter of the distribution

scale<- `signature(object = "Weibull")`: modifies the slot `scale` of the parameter of the distribution

shape `signature(object = "Weibull")`: returns the slot `shape` of the parameter of the distribution

shape<- `signature(object = "Weibull")`: modifies the slot `shape` of the parameter of the distribution

Note

The density is $d(x) = 0$ for $x < 0$.
The cumulative is $p(x) = 1 - \exp(-(x/\sigma)^a)$,
the mean is $E(X) = \sigma\Gamma(1 + 1/a)$,
and the $Var(X) = \sigma^2(\Gamma(1 + 2/a) - (\Gamma(1 + 1/a))^2)$.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`WeibullParameter-class` `AbscontDistribution-class` `Reals-class` `rweibull`

Examples

```
W=Weibull(shape=1,scale=1) # W is a Weibull distribution with shape=1 and scale=1.
r(W)(1) # one random number generated from this distribution, e.g. 0.5204105
d(W)(1) # Density of this distribution is 0.3678794 for x=1.
p(W)(1) # Probability that x<1 is 0.6321206.
q(W)(.1) # Probability that x<0.1053605 is 0.1.
shape(W) # shape of this distribution is 1.
shape(W)=2 # shape of this distribution is now 2.
```

`WeibullParameter-class`

Class "WeibullParameter"

Description

The parameter of a Weibull distribution, used by `Weibull`-class

Objects from the Class

Objects can be created by calls of the form `new("WeibullParameter", shape, scale)`. Usually an object of this class is not needed on its own, it is generated automatically when an object of the class `Weibull` is instantiated.

Slots

shape: Object of class "numeric": the shape of a Weibull distribution

scale: Object of class "numeric": the scale of a Weibull distribution

name: Object of class "character": a name / comment for the parameters

Extends

Class "Parameter", directly.

Methods

initialize signature(.Object = "WeibullParameter"): initialize method

scale signature(object = "WeibullParameter"): returns the slot scale of a parameter of a Weibull distribution

scale<- signature(object = "WeibullParameter"): modifies the slot scale of a parameter of a Weibull distribution

shape signature(object = "WeibullParameter"): returns the slot shape of a parameter of a Weibull distribution

shape<- signature(object = "WeibullParameter"): modifies the slot shape of a parameter of a Weibull distribution

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

Weibull-class Parameter-class

Examples

```
W=new("WeibullParameter",shape=1,scale=1)
shape(W) # shape of this distribution is 1.
shape(W)=2 # shape of this distribution is now 2.
```

---

call.ev-methods *Methods for Function call.ev in Package 'distr'*

Description

call.ev-methods

Methods

call.ev signature(object = "Evaluation"): returns the call which created the object

cload *cload*

Description

loads the comment file from a saved `Dataclass` or `Evaluation` object

Usage

```
cload(filename)
```

Arguments

`filename`	the filename which was declared at the instantiaton of the Dataclass

Details

Value

no value is returned

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`Evaluation-class` `Dataclass-class` `load` `savedata-methods`

Examples

```
# see Dataclass and Evaluation for examples
## The function is currently defined as
function(filename){
  eval.parent(parse(text=paste("load(\"",filename,".comment\")", sep = "")))
  }
```

d-methods *Methods for Function d in Package 'distr'*

Description

d-methods

Methods

d `signature(object = "Distribution")`: returns the density function

df-methods *Methods for Function df in Package 'distr'*

Description

df-methods

Methods

df `signature(object = "TParameter")`: returns the slot `df` of the parameter of the distribution

df<- `signature(object = "TParameter")`: modifies the slot `df` of the parameter of the distribution

df `signature(object = "Td")`: returns the slot `df` of the parameter of the distribution

df<- `signature(object = "Td")`: modifies the slot `df` of the parameter of the distribution

df `signature(object = "ChisqParameter")`: returns the slot `df` of the parameter of the distribution

df<- `signature(object = "ChisqParameter")`: modifies the slot `df` of the parameter of the distribution

df `signature(object = "Chisq")`: returns the slot `df` of the parameter of the distribution

df<- `signature(object = "Chisq")`: modifies the slot `df` of the parameter of the distribution

df1-methods *Methods for Function df1 in Package 'distr'*

Description

df-methods

Methods

df1 `signature(object = "FParameter")`: returns the slot `df1` of the parameter of an F-distribution

df1<- `signature(object = "FParameter")`: modifies the slot `df1` of the parameter of an F-distribution

df1 `signature(object = "Fd")`: returns the slot `df1` of the slot `param` of the distribution

df1<- `signature(object = "Fd")`: modifies the slot `df1` of the slot `param` of the distribution

df2-methods *Methods for Function df2 in Package 'distr'*

Description

df-methods

Methods

df2 `signature(object = "FParameter")`: returns the slot `df2` of the parameter of an F-distribution

df2<- `signature(object = "FParameter")`: modifies the slot `df2` of the parameter of an F-distribution

df2 `signature(object = "Fd")`: returns the slot `df2` of the slot `param` of the distribution

df2<- `signature(object = "Fd")`: modifies the slot `df2` of the slot `param` of the distribution

dimension-methods *Methods for Function dimension in Package 'distr'*

Description

dimension-methods

Methods

dimension `signature(object = "EuclideanSpace")`: returns the dimension of the space

dimension<- `signature(object = "EuclideanSpace")`: modifies the dimension of the space

distribution-methods *Methods for Function distribution in Package 'distr'*

Description

distribution-methods

Methods

distribution `signature(object = "Simulation")`: returns the slot `distribution` of the simulation

distribution<- `signature(object = "Simulation")`: changes the slot `distribution` of the simulation

distribution.c-methods

Methods for Function distribution.c in Package 'distr'

Description

distribution-methods

Methods

distribution.c `signature(object = "Contsimulation")`: returns the distribution of the contaminated data

distribution.c<- `signature(object = "Contsimulation")`: changes the distribution of the contaminated data

distribution.id-methods

Methods for Function distribution.id in Package 'distr'

Description

distribution-methods

Methods

distribution.id `signature(object = "Contsimulation")`: returns the distribution of the ideal data

distribution.id<- `signature(object = "Contsimulation")`: changes the distribution of the ideal data

distroptions

function to change the global variables of the package 'distr'

Description

With `distroptions` you may inspect and change the global variables used by the package **'distr'**

Usage

```
distroptions(arg = "missing", value = -1)
```

Arguments

`arg`	the global variable to be shown or changed
`value`	the new value of the global variable

Value

`distroptions()` returns a list of the global variables. `distroptions(arg=x)` returns the global variable *x*. `distroptions(arg=x,value=y)` sets the value of the global variable *x* to *y*.

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

Examples

```
distroptions("RtoDPQ.e") # returns the value of RtoDPQ.e, by default = 5
distroptions("RtoDPQ.e",6) # now RtoDPQ.e = 6

## The function is currently defined as
function (arg = "missing", value = -1)
{
    globals <- list(DefaultNrFFTGridPointsExponent = DefaultNrFFTGridPointsExponent,
        DefaultNrGridPoints = DefaultNrGridPoints, DistrResolution = DistrResolution,
        RtoDPQ.e = RtoDPQ.e, TruncQuantile = TruncQuantile)
    if (arg == "missing") {
        print(globals)
        return(invisible())
    }
    if (!any(arg == names(globals)))
        stop(paste("No such variable:", arg))
    if (value == -1)
        switch(arg, DefaultNrGridPoints = DefaultNrGridPoints,
            DistrResolution = DistrResolution, TruncQuantile = TruncQuantile,
            DefaultNrFFTGridPointsExponent = DefaultNrFFTGridPointsExponent,
            RtoDPQ.e = RtoDPQ.e)
    else eval.parent(parse(text = paste("assignInNamespace(\"",
        arg, "\",", value, ", \"distr\")", sep = "")))
  }
```

---

estimator-methods *Methods for Function estimator in Package 'distr'*

Description

estimator-methods

Methods

estimator `signature(object = "Evaluation")`: returns the estimator

evaluate-methods *Methods for Function evaluate in Package 'distr'*

Description

evaluate-methods

Methods

evaluate `signature(object = "Dataclass", estimator = "function")`: creates an object of type "Evaluation", see there for further information

See Also

`Evaluation-class`

filename-methods *Methods for Function filename in Package 'distr'*

Description

filename-methods

Methods

filename `signature(object = "Dataclass")`: returns the filename

filename<- `signature(object = "Dataclass")`: changes the filename

filename `signature(object = "Evaluation")`: returns the filename of the evaluated object

img-methods *Methods for Function img in Package 'distr'*

Description

img-methods

Methods

img `signature(object = "Distribution")`: returns the image space / domain of the distribution

ind-methods *Methods for Function ind in Package 'distr'*

Description

ind-methods

Methods

ind `signature(object = "Contsimulation")`: returns an indicator showing which data is contaminated

k-methods *Methods for Function k in Package 'distr'*

Description

k-methods

Methods

k `signature(object = "HyperParameter")`: returns the slot `k` of the parameter of the distribution

k<- `signature(object = "HyperParameter")`: modifies the slot `k` of the parameter of the distribution

k `signature(object = "Hyper")`: returns the slot `k` of the parameter of the distribution

k<- `signature(object = "Hyper")`: modifies the slot `k` of the parameter of the distribution

lambda-methods *Methods for Function lambda in Package 'distr'*

Description

lambda-methods

Methods

lambda `signature(object = "PoisParameter")`: returns the slot `lambda` of the parameter of the distribution

lambda<- `signature(object = "PoisParameter")`: modifies the slot `lambda` of the parameter of the distribution

lambda `signature(object = "Pois")`: returns the slot `lambda` of the parameter of the distribution

lambda<- `signature(object = "Pois")`: modifies the slot `lambda` of the parameter of the distribution

liesIn-methods *Methods for Function liesIn in Package 'distr'*

Description

liesIn-methods

Methods

liesIn `signature(object = "EuclideanSpace", x = "numeric")`:
Does a particular vector lie in this space or not?

liesIn `signature(object = "Naturals", x = "numeric")`:
Does a particular vector only contain naturals?

location-methods *Methods for Function location in Package 'distr'*

Description

location-methods

Methods

location `signature(object = "LogisParameter")`: returns the slot `location` of the parameter of the distribution

location<- `signature(object = "LogisParameter")`: modifies the slot `location` of the parameter of the distribution

location `signature(object = "Logis")`: returns the slot `location` of the parameter of the distribution

location<- `signature(object = "Logis")`: modifies the slot `location` of the parameter of the distribution

location `signature(object = "CauchyParameter")`: returns the slot `location` of the parameter of the distribution

location<- `signature(object = "CauchyParameter")`: modifies the slot `location` of the parameter of the distribution

location `signature(object = "Cauchy")`: returns the slot `location` of the parameter of the distribution

location<- `signature(object = "Cauchy")`: modifies the slot `location` of the parameter of the distribution

location `signature(object = "DiracParameter")`: returns the slot `location` of the parameter of the distribution

location<- `signature(object = "DiracParameter")`: modifies the slot `location` of the parameter of the distribution

location `signature(object = "Dirac")`: returns the slot `location` of the parameter of the distribution

location<- `signature(object = "Dirac")`: modifies the slot `location` of the parameter of the distribution

m-methods *Methods for Function m in Package 'distr'*

Description

m-methods

Methods

m `signature(object = "HyperParameter")`: returns the slot m of the parameter of the distribution

m<- `signature(object = "HyperParameter")`: modifies the slot m of the parameter of the distribution

m `signature(object = "Hyper")`: returns the slot m of the parameter of the distribution

m<- `signature(object = "Hyper")`: modifies the slot m of the parameter of the distribution

mean-methods *Methods for Function mean in Package 'distr'*

Description

mean-methods

Methods

mean `signature(object = "NormParameter")`: returns the slot mean of the parameter of the distribution

mean<- `signature(object = "NormParameter")`: modifies the slot mean of the parameter of the distribution

mean `signature(object = "Norm")`: returns the slot mean of the parameter of the distribution

mean<- `signature(object = "Norm")`: modifies the slot mean of the parameter of the distribution

meanlog-methods *Methods for Function meanlog in Package 'distr'*

Description

meanlog-methods

Methods

meanlog `signature(object = "LnormParameter")`: returns the slot meanlog of the parameter of the distribution

meanlog<- `signature(object = "LnormParameter")`: modifies the slot meanlog of the parameter of the distribution

meanlog `signature(object = "Lnorm")`: returns the slot meanlog of the parameter of the distribution

meanlog<- `signature(object = "Lnorm")`: modifies the slot meanlog of the parameter of the distribution

n-methods *Methods for Function n in Package 'distr'*

Description

n-methods

Methods

n `signature(object = "HyperParameter")`: returns the slot `n` of the parameter of the distribution

n<- `signature(object = "HyperParameter")`: modifies the slot `n` of the parameter of the distribution

n `signature(object = "Hyper")`: returns the slot `n` of the parameter of the distribution

n<- `signature(object = "Hyper")`: modifies the slot `n` of the parameter of the distribution

name-methods *Methods for Function name in Package 'distr'*

Description

name-methods

Methods

name `signature(object = "Parameter")`: returns the slot name of the parameter

name<- `signature(object = "Parameter")`: modifies the slot name of the parameter

name `signature(object = "rSpace")`: returns the slot name of the space

name<- `signature(object = "rSpace")`: modifies the slot name of the space

name `signature(object = "Evaluation")`: returns the slot name of data object

ncp-methods *Methods for Function ncp in Package 'distr'*

Description

ncp-methods

Methods

ncp `signature(object = "ChisqParameter")`: returns the slot `ncp` of the parameter of the distribution

ncp<- `signature(object = "ChisqParameter")`: modifies the slot `ncp` of the parameter of the distribution

ncp `signature(object = "Chisq")`: returns the slot `ncp` of the parameter of the distribution

ncp<- `signature(object = "Chisq")`: modifies the slot `ncp` of the parameter of the distribution

operators-methods *Methods for operators +,-,*,/ in Package 'distr'*

Description

operator-methods

Methods

`- signature(e1 = "UnivariateDistribution")`:
application of '-' to this univariate distribution

`* signature(e1 = "UnivariateDistribution", e2 = "numeric")`:
multiplication of this univariate distribution by an object of class 'numeric'

`/ signature(e1 = "UnivariateDistribution", e2 = "numeric")`:
division of this univariate distribution by an object of class 'numeric'

`+ signature(e1 = "UnivariateDistribution", e2 = "numeric")`:
addition of this univariate distribution to an object of class 'numeric'

`- signature(e1 = "UnivariateDistribution", e2 = "numeric")`:
subtraction of an object of class 'numeric' from this univariate distribution

`* signature(e1 = "numeric", e2 = "UnivariateDistribution")`:
multiplication of this univariate distribution by an object of class 'numeric'

`+ signature(e1 = "numeric", e2 = "UnivariateDistribution")`:
addition of this univariate distribution to an object of class 'numeric'

`- signature(e1 = "numeric", e2 = "UnivariateDistribution")`:
subtraction of this univariate distribution from an object of class 'numeric'

`+ signature(e1 = "UnivariateDistribution", e2 = "UnivariateDistribution")`:
Convolution of two univariate distributions. The slots p, d and q are approximated by grids.

`- signature(e1 = "UnivariateDistribution", e2 = "UnivariateDistribution")`:
Convolution of two univariate distributions. The slots p, d and q are approximated by grids.

`- signature(e1 = "AbscontDistribution")`:
application of '-' to this absolutely continuous distribution

`* signature(e1 = "AbscontDistribution", e2 = "numeric")`:
multiplication of this absolutely continuous distribution by an object of class 'numeric'

`/ signature(e1 = "AbscontDistribution", e2 = "numeric")`:
division of this absolutely continuous distribution by an object of class 'numeric'

`+ signature(e1 = "AbscontDistribution", e2 = "numeric")`:
addition of this absolutely continuous distribution to an object of class 'numeric'

`- signature(e1 = "AbscontDistribution", e2 = "numeric")`:
subtraction of an object of class 'numeric' from this absolutely continuous distribution

`* signature(e1 = "numeric", e2 = "AbscontDistribution")`:
multiplication of this absolutely continuous distribution by an object of class 'numeric'

`+ signature(e1 = "numeric", e2 = "AbscontDistribution")`:
addition of this absolutely continuous distribution to an object of class 'numeric'

`- signature(e1 = "numeric", e2 = "AbscontDistribution")`:
subtraction of this absolutely continuous distribution from an object of class 'numeric'

`+ signature(e1 = "AbscontDistribution", e2 = "AbscontDistribution")`:
Convolution of two absolutely continuous distributions. The slots p, d and q are approximated by grids.

`- signature(e1 = "AbscontDistribution", e2 = "AbscontDistribution")`:
Convolution of two absolutely continuous distributions. The slots p, d and q are approximated by grids.

`- signature(e1 = "DiscreteDistribution")`:
application of '-' to this discrete distribution

`* signature(e1 = "DiscreteDistribution", e2 = "numeric")`:
multiplication of this discrete distribution by an object of class 'numeric'

`/ signature(e1 = "DiscreteDistribution", e2 = "numeric")`:
division of this discrete distribution by an object of class 'numeric'

`+ signature(e1 = "DiscreteDistribution", e2 = "numeric")`:
addition of this discrete distribution to an object of class 'numeric'

`- signature(e1 = "DiscreteDistribution", e2 = "numeric")`:
subtraction of an object of class 'numeric' from this discrete distribution

`* signature(e1 = "numeric", e2 = "DiscreteDistribution")`:
multiplication of this discrete distribution by an object of class 'numeric'

`+ signature(e1 = "numeric", e2 = "DiscreteDistribution")`:
addition of this discrete distribution to an object of class 'numeric'

`- signature(e1 = "numeric", e2 = "DiscreteDistribution")`:
subtraction of this discrete distribution from an object of class 'numeric'

`+ signature(e1 = "DiscreteDistribution", e2 = "DiscreteDistribution")`:
Convolution of two discrete distributions. The slots p, d and q are approximated by grids.

`- signature(e1 = "DiscreteDistribution", e2 = "DiscreteDistribution")`:
Convolution of two discrete distributions. The slots p, d and q are approximated by grids.

`* signature(e1 = "numeric", e2 = "Norm")`:
multiplication of this normal distribution by an object of class 'numeric'

`+ signature(e1 = "numeric", e2 = "Norm")`:
addition of this normal distribution to an object of class 'numeric'

`- signature(e1 = "numeric", e2 = "Norm")`:
subtraction of this normal distribution from an object of class 'numeric'

`* signature(e1 = "Norm", e2 = "numeric")`:
multiplication of this normal distribution by an object of class 'numeric'

`+ signature(e1 = "Norm", e2 = "numeric")`:
addition of this normal distribution to an object of class 'numeric'

`- signature(e1 = "Norm", e2 = "numeric")`:
subtraction of an object of class 'numeric' from this normal distribution

`/ signature(e1 = "Norm", e2 = "numeric")`:
division of this normal distribution by an object of class 'numeric'

`- signature(e1 = "Norm", e2 = "Norm")`
`+ signature(e1 = "Norm", e2 = "Norm")`:
For the normal distribution the exact convolution formulas are implemented thereby improving the general numerical approximation.

`* signature(e1 = "Unif", e2 = "numeric")`:
multiplication of this uniform distribution by an object of class 'numeric'

`+ signature(e1 = "Unif", e2 = "numeric")`:
addition of this uniform distribution to an object of class 'numeric'

`+ signature(e1 = "Binom", e2 = "Binom")`:
For two binomial distributions with the same probabilities the exact convolution formula is implemented thereby improving the general numerical approximation.

`+ signature(e1 = "Pois", e2 = "Pois")`:
For the Poisson distribution the exact convolution formula is implemented thereby improving the general numerical approximation.

See Also

`UnivariateDistribution-class` `AbscontDistribution-class` `DiscreteDistribution-class` `Norm-class` `Binom-class` `Pois-class`

`p-methods` *Methods for Function p in Package 'distr'*

Description

p-methods

Methods

p `signature(object = "Distribution")`: returns the cumulative distribution function

param-methods *Methods for Function param in Package 'distr'*

Description

param-methods

Methods

param `signature(object = "Distribution")`: returns the parameter

plot-methods *Methods for Function plot in Package 'distr'*

Description

plot-methods

Methods

plot `signature(object = "AbscontDistribution")`: plots density, cumulative and the inverse of the cumulative function

plot `signature(object = "DiscreteDistribution")`: plots density, cumulative and the inverse of the cumulative function

plot `signature(object = "Dataclass")`: produces a plot of the data matrix

plot `signature(object = "Simulation")`: produces a plot of the data matrix

plot `signature(object = "Contsimulation")`: produces a plot of the real data matrix

plot `signature(object = "Evaluation")`: returns a boxplot of the result

print-methods *Methods for Function print in Package 'distr'*

Description

print-methods

Methods

print `signature(x = "UnivariateDistribution")`: returns the class of the object and its parameters

print `signature(x = "Dataclass")`: returns filename, number of runs and the size of the sample

print `signature(x = "Simulation")`: returns filename, seed, number of runs, the size of the sample and the distribution

print `signature(x = "Contsimulation")`: returns filename, seed, number of runs, the size of the sample, the rate and the distributions

print `signature(x = "Evaluation")`: returns the name of the data object, its filename, the estimator used and the result

prob-methods *Methods for Function prob in Package 'distr'*

Description

prob-methods

Methods

prob `signature(object = "BinomParameter")`: returns the slot `prop` of the parameter of the distribution

prob<- `signature(object = "BinomParameter")`: modifies the slot `prob` of the parameter of the distribution

prob `signature(object = "Binom")`: returns the slot `prop` of the parameter of the distribution

prob<- `signature(object = "Binom")`: modifies the slot `prob` of the parameter of the distribution

prob `signature(object = "NbinomParameter")`: returns the slot `prop` of the parameter of the distribution

prob<- `signature(object = "NbinomParameter")`: modifies the slot `prob` of the parameter of the distribution

prob `signature(object = "Nbinom")`: returns the slot `prop` of the parameter of the distribution

prob<- `signature(object = "Nbinom")`: modifies the slot `prob` of the parameter of the distribution

prob `signature(object = "GeomParameter")`: returns the slot `prop` of the parameter of the distribution

prob<- `signature(object = "GeomParameter")`: modifies the slot `prob` of the parameter of the distribution

prob `signature(object = "Geom")`: returns the slot `prop` of the parameter of the distribution

prob<- `signature(object = "Geom")`: modifies the slot `prob` of the parameter of the distribution

q-methods *Methods for Function q in Package 'distr'*

Description

q-methods

Methods

q `signature(object = "Distribution")`: returns the quantile function

r-methods *Methods for Function r in Package 'distr'*

Description

r-methods

Methods

p `signature(object = "Distribution")`: returns the cumulative distribution function

rSpace-class *Class "rSpace"*

Description

The distribution-classes contain a slot where the sample space is stored. Typically, discrete random variables take naturals as values. rSpace is the mother-class of the class EuclideanSpace.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

name: Object of class `"character"`: the name of the space

Methods

name `signature(object = "rSpace")`: returns the name of the space

name<- `signature(object = "rSpace")`: changes the name of the space

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

See Also

`EuclideanSpace-class` `Distribution-class`

rate-methods *Methods for Function rate in Package 'distr'*

Description

rate-methods

Methods

rate `signature(object = "ExpParameter")`: returns the slot `rate` of the parameter of the distribution

rate<- `signature(object = "ExpParameter")`: modifies the slot `rate` of the parameter of the distribution

rate `signature(object = "Exp")`: returns the slot `rate` of the parameter of the distribution

rate<- `signature(object = "Exp")`: modifies the slot `rate` of the parameter of the distribution

rate `signature(object = "Contsimulation")`: returns the contamination rate

rate<- `signature(object = "Contsimulation")`: modifies the contamination rate

result-methods *Methods for Function result in Package 'distr'*

Description

result-methods

Methods

result `signature(object = "Evaluation")`: returns the result of an evaluation

runs-methods *Methods for Function runs in Package 'distr'*

Description

runs-methods

Methods

runs `signature(object = "Dataclass")`: returns the number of runs

runs<- `signature(object = "Simulation")`: changes the number of runs

runs<- `signature(object = "Contsimulation")`: changes the number of runs

samplesize-methods *Methods for Function samplesize in Package 'distr'*

Description

samplesize-methods

Methods

samplesize `signature(object = "Dataclass")`: returns the size of the sample

samplesize<- `signature(object = "Simulation")`: changes the size of the sample

samplesize<- `signature(object = "Contsimulation")`: changes the size of the sample

savedata-methods *Methods for Function savedata in Package 'distr'*

Description

savedata-methods

Methods

savedata `signature(object = "Dataclass")`: saves the object (with the data) in the directory of R

savedata `signature(object = "Simulation")`: saves the object without the data in the directory of R (After loading the data can be reproduced by using simulate.)

savedata `signature(object = "Contsimulation")`: saves the object without the data in the directory of R (After loading the data can be reproduced by using simulate.)

savedata `signature(object = "Evaluation")`: saves the object in two files in the directory of R - one with data, one without as comment file

See Also

`Dataclass-class` `Simulation-class` `Contsimulation-class` `Evaluation-class`

`scale-methods` *Methods for Function scale in Package 'distr'*

Description

scale-methods

Methods

scale `signature(object = "GammaParameter")`: returns the slot `scale` of the parameter of the distribution

scale<- `signature(object = "GammaParameter")`: modifies the slot `scale` of the parameter of the distribution

scale `signature(object = "Gammad")`: returns the slot `scale` of the parameter of the distribution

scale<- `signature(object = "Gammad")`: modifies the slot `scale` of the parameter of the distribution

scale `signature(object = "LogisParameter")`: returns the slot `scale` of the parameter of the distribution

scale<- `signature(object = "LogisParameter")`: modifies the slot `scale` of the parameter of the distribution

scale `signature(object = "Logis")`: returns the slot `scale` of the parameter of the distribution

scale<- `signature(object = "Logis")`: modifies the slot `scale` of the parameter of the distribution

scale `signature(object = "WeibullParameter")`: returns the slot `scale` of the parameter of the distribution

scale<- `signature(object = "WeibullParameter")`: modifies the slot `scale` of the parameter of the distribution

scale `signature(object = "Weibull")`: returns the slot `scale` of the parameter of the distribution

scale<- `signature(object = "Weibull")`: modifies the slot `scale` of the parameter of the distribution

scale `signature(object = "CauchyParameter")`: returns the slot `scale` of the parameter of the distribution

scale<- `signature(object = "CauchyParameter")`: modifies the slot `scale` of the parameter of the distribution

scale `signature(object = "Cauchy")`: returns the slot `scale` of the parameter of the distribution

scale<- `signature(object = "Cauchy")`: modifies the slot `scale` of the parameter of the distribution

sd-methods *Methods for Function sd in Package 'distr'*

Description

sd-methods

Methods

sd `signature(object = "NormParameter")`: returns the slot sd of the parameter of the distribution

sd<- `signature(object = "NormParameter")`: modifies the slot sd of the parameter of the distribution

sd `signature(object = "Norm")`: returns the slot sd of the parameter of the distribution

sd<- `signature(object = "Norm")`: modifies the slot sd of the parameter of the distribution

sdlog-methods *Methods for Function sdlog in Package 'distr'*

Description

Methods

sdlog `signature(object = "LnormParameter")`: returns the slot `sdlog` of the parameter of the distribution

sdlog<- `signature(object = "LnormParameter")`: modifies the slot `sdlog` of the parameter of the distribution

sdlog `signature(object = "Lnorm")`: returns the slot `sdlog` of the parameter of the distribution

sdlog<- `signature(object = "Lnorm")`: modifies the slot `sdlog` of the parameter of the distribution

seed-methods *Methods for Function seed in Package 'distr'*

Description

seed-methods

Methods

seed `signature(object = "Simulation")`: returns the slot `seed` of an object of class `"Simulation"`

seed<- `signature(object = "Simulation")`: changes the slot `seed` of an object of class `"Simulation"`

seed `signature(object = "Contsimulation")`: returns the slot `seed` of an object of class `"Contsimulation"`

seed<- `signature(object = "Contsimulation")`: changes the slot `seed` of an object of class `"Contsimulation"`

Note

The value to which the seed is set has to be consistent with the **setRNG**-package; to draw a "new" simulation, use something like `seed(X)<-setRNG();simulate(X)`; cf. manual to this package, p. 9

shape-methods *Methods for Function shape in Package 'distr'*

Description

shape-methods

Methods

shape `signature(object = "GammaParameter")`: returns the slot `shape` of a parameter of a Gamma distribution

shape<- `signature(object = "GammaParameter")`: modifies the slot `shape` of a parameter of a Gamma distribution

shape `signature(object = "Gammad")`: returns the slot `shape` of the parameter slot of a Gamma distribution

shape<- `signature(object = "Gammad")`: modifies the slot shape of the parameter slot of a Gamma distribution

shape `signature(object = "WeibullParameter")`: returns the slot `shape` of a parameter of a Weibull distribution

shape<- `signature(object = "WeibullParameter")`: modifies the slot `shape` of a parameter of a Weibull distribution

shape `signature(object = "Weibull")`: returns the slot `shape` of the parameter slot of the distribution

shape<- `signature(object = "Weibull")`: modifies the slot `shape` of the parameter slot of the distribution

shape1-methods *Methods for Function shape1 in Package 'distr'*

Description

shape-methods

Methods

shape1 `signature(object = "BetaParameter")`: returns the slot `shape1` of the parameter of the distribution

shape1<- `signature(object = "BetaParameter")`: modifies the slot `shape1` of the parameter of the distribution

shape1 `signature(object = "Beta")`: returns the slot `shape1` of the parameter of the distribution

shape1<- `signature(object = "Beta")`: modifies the slot `shape1` of the parameter of the distribution

shape2-methods *Methods for Function shape2 in Package 'distr'*

Description

shape-methods

Methods

shape2 `signature(object = "BetaParameter")`: returns the slot shape2 of the parameter of the distribution

shape2<- `signature(object = "BetaParameter")`: modifies the slot shape2 of the parameter of the distribution

shape2 `signature(object = "Beta")`: returns the slot shape2 of the parameter of the distribution

shape2<- `signature(object = "Beta")`: modifies the slot shape2 of the parameter of the distribution

simplifyr-methods *Methods for Function simplifyr in Package 'distr'*

Description

simplifyr-methods

Methods

simplifyr `signature(.Object = "UnivariateDistribution")`: After several transformations of a given distribution it may take quite a long time to generate random numbers from the resulting distribution. simplifyr generates a certain number, by default 10^5, of random numbers once. This pool of random numbers forms the basis for further uses of the r-method. That is, random numbers are generated by sampling with replacement out of this pool.

Note

If you want to generate many random numbers, you should use simplifyr with a big size to be sure, that your numbers are really random.

See Also

`Distribution-class`

Examples

```
F <- ( Norm() + Binom() + Pois() + Exp() ) * 2 - 10
system.time(r(F)(10^6))
simplifyr(F, size = 10^6)
system.time(r(F)(10^6))
```

simulate-methods *Methods for Function simulate in Package 'distr'*

Description

simulate-methods

Methods

simulate `signature(x = "Simulation")`: generates the random numbers for the simulation

simulate `signature(x = "Contsimulation")`: generates the random numbers for the simulation

size-methods *Methods for Function size in Package 'distr'*

Description

size-methods

Methods

size `signature(object = "BinomParameter")`: returns the slot `size` of the parameter of the distribution

size<- `signature(object = "BinomParameter")`: modifies the slot `size` of the parameter of the distribution

size `signature(object = "Binom")`: returns the slot `size` of the parameter of the distribution

size<- `signature(object = "Binom")`: modifies the slot `size` of the parameter of the distribution

size `signature(object = "NbinomParameter")`: returns the slot `size` of the parameter of the distribution

size<- `signature(object = "NbinomParameter")`: modifies the slot `size` of the parameter of the distribution

size `signature(object = "Nbinom")`: returns the slot `size` of the parameter of the distribution

size<- `signature(object = "Nbinom")`: modifies the slot `size` of the parameter of the distribution

standardMethods *standardMethods*

Description

Creates definitions for accessor and replacement functions of an given class.

Usage

```
standardMethods(class, writetofile = FALSE, directory)
```

Arguments

`class`	the class for which accessor and replacement functions are to be produced, given as a string
`writetofile`	logical value, indicating wheter output is to be written to a file
`directory`	if writetofile = TRUE, the output is written to a file in the given directory, the name of the file starting with "classname " and ending with "StandardMethods.txt"

Value

no value is returned

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩

Examples

```
setClass("testclass", representation(a = "numeric", b = "character"))
standardMethods("testclass")
#directory = "C:/Dokumente und Einstellungen/X/Eigene Dateien/Studium/R/SWP/"
#standardMethods("testclass", writetofile = TRUE, directory = directory)
```

---

summary-methods *Methods for Function summary in Package 'distr'*

Description

summary-methods

Methods

summary `signature(object = "Dataclass")`: returns the same information as print, moreover a statistical summary for each run

summary `signature(object = "Simulation")`: returns filename, seed, number of runs, the size of the sample and a statistical summary for each run

summary `signature(object = "Contsimulation")`: returns filename, seed, number of runs, the size of the sample, the rate and a statistical summary for each run of the real data

summary `signature(object = "Evaluation")`: returns the name of the data object, its filename, the estimator used and a statistical summary of the result

---

support-methods *Methods for Function support in Package 'distr'*

Description

support-methods

Methods

support `signature(object = "DiscreteDistributionContsimulation")`: returns the support

`vectororNULL-class` *Class "vectororNULL"*

Description

auxiliary class; may contain either a vector or NULL, cf. J. Chambers, "green book".

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "vectororNULL" in the signature.

Note

Dataclass-class can save data either of type "NULL" (means no data) or "vector"

Author(s)

Thomas Stabla ⟨Thomas.Stabla@uni-bayreuth.de⟩,
Florian Camphausen ⟨Florian.Camphausen@uni-bayreuth.de⟩,
Peter Ruckdeschel ⟨Peter.Ruckdeschel@uni-bayreuth.de⟩,
Matthias Kohl ⟨Matthias.Kohl@uni-bayreuth.de⟩

Index