Package ‘distributions3’

September 3, 2019

Title Probability Distributions as S3 Objects

Version 0.1.1

Description Tools to create and manipulate probability
distributions using S3. Generics random(), pdf(), cdf() and
quantile() provide replacements for base R's r/d/p/q style functions.
Functions and arguments have been named carefully to minimize
confusion for students in intro stats courses. The documentation for
each distribution contains detailed mathematical notes.

License MIT + file LICENSE

URL https://github.com/alexpghayes/distributions3

BugReports https://github.com/alexpghayes/distributions3/issues

Imports ellipsis, glue

Suggests covr, cowplot, ggplot2, knitr, rmarkdown, testthat (>= 2.1.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Alex Hayes [aut, cre] (<https://orcid.org/0000-0002-4985-5160>),
Ralph Moller-Trane [aut],
Emil Hvitfeldt [ctb] (<https://orcid.org/0000-0002-0679-1945>),
Daniel Jordan [ctb],
Bruna Wundervald [ctb]

Maintainer Alex Hayes <alexpghayes@gmail.com>

Repository CRAN

Date/Publication 2019-09-03 16:10:06 UTC

R topics documented:

`Bernoulli` *Create a Bernoulli distribution*

Description

Bernoulli distributions are used to represent events like coin flips when there is single trial that is either successful or unsuccessful. The Bernoulli distribution is a special case of the `Binomial()` distribution with `n = 1`.

Usage

```
Bernoulli(p = 0.5)
```

Arguments

p The success probability for the distribution. p can be any value in [0,1], and defaults to 0.5.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail.

In the following, let X be a Bernoulli random variable with parameter p = p. Some textbooks also define $q = 1 - p$, or use π instead of p.

The Bernoulli probability distribution is widely used to model binary variables, such as 'failure' and 'success'. The most typical example is the flip of a coin, when p is thought as the probability of flipping a head, and $q = 1 - p$ is the probability of flipping a tail.

Support: $\{0, 1\}$

Mean: p

Variance: $p \cdot (1 - p) = p \cdot q$

Probability mass function (p.m.f):

$$P(X = x) = p^x (1 - p)^{1-x} = p^x q^{1-x}$$

Cumulative distribution function (c.d.f):

$$P(X \le x) = \begin{cases} 0 & x < 0 \\ 1 - p & 0 \le x < 1 \\ 1 & x \ge 1 \end{cases}$$

Moment generating function (m.g.f):

$$E(e^{tX}) = (1 - p) + pe^t$$

Value

A Bernoulli object.

See Also

Other discrete distributions: Binomial, Categorical, Geometric, HyperGeometric, Multinomial, NegativeBinomial, Poisson

Examples

```
set.seed(27)

X <- Bernoulli(0.7)
X
```

```
random(X, 10)
pdf(X, 1)
log_pdf(X, 1)
cdf(X, 0)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

Beta *Create a Beta distribution*

Description

Create a Beta distribution

Usage

```
Beta(alpha = 1, beta = 1)
```

Arguments

alpha	The alpha parameter. alpha can be any value strictly greater than zero. Defaults to 1.
beta	The beta parameter. beta can be any value strictly greater than zero. Defaults to 1.

Value

A beta object.

See Also

Other continuous distributions: Cauchy, ChiSquare, Exponential, FisherF, Gamma, LogNormal, Logistic, Normal, StudentsT, Tukey, Uniform, Weibull

Examples

```
set.seed(27)

X <- Beta(1, 2)
X

random(X, 10)

pdf(X, 0.7)
log_pdf(X, 0.7)
```

```
cdf(X, 0.7)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

`Binomial` *Create a Binomial distribution*

Description

Binomial distributions are used to represent situations can that can be thought as the result of n Bernoulli experiments (here the n is defined as the `size` of the experiment). The classical example is n independent coin flips, where each coin flip has probability `p` of success. In this case, the individual probability of flipping heads or tails is given by the Bernoulli(p) distribution, and the probability of having x equal results (x heads, for example), in n trials is given by the Binomial(n, p) distribution. The equation of the Binomial distribution is directly derived from the equation of the Bernoulli distribution.

Usage

```
Binomial(size, p = 0.5)
```

Arguments

`size`	The number of trials. Must be an integer greater than or equal to one. When `size = 1L`, the Binomial distribution reduces to the bernoulli distribution. Often called `n` in textbooks.
`p`	The success probability for a given trial. `p` can be any value in `[0,1]`, and defaults to `0.5`.

Details

The Binomial distribution comes up when you are interested in the portion of people who do a thing. The Binomial distribution also comes up in the sign test, sometimes called the Binomial test (see `stats::binom.test()`), where you may need the Binomial C.D.F. to compute p-values.

We recommend reading this documentation on `https://alexpghayes.github.io/distributions3`, where the math will render with additional detail.

In the following, let X be a Binomial random variable with parameter `size` = n and `p` = p. Some textbooks define $q = 1 - p$, or called π instead of p.

Support: $\{0, 1, 2, ..., n\}$

Mean: np

Variance: $np \cdot (1 - p) = np \cdot q$

Probability mass function (p.m.f):

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Cumulative distribution function (c.d.f):

$$P(X \le k) = \sum_{i=0}^{\lfloor k \rfloor} \binom{n}{i} p^i (1-p)^{n-i}$$

Moment generating function (m.g.f):

$$E(e^{tX}) = (1 - p + pe^t)^n$$

Value

A `Binomial` object.

See Also

Other discrete distributions: `Bernoulli`, `Categorical`, `Geometric`, `HyperGeometric`, `Multinomial`, `NegativeBinomial`, `Poisson`

Examples

```
set.seed(27)

X <- Binomial(10, 0.2)
X

random(X, 10)

pdf(X, 2L)
log_pdf(X, 2L)

cdf(X, 4L)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`Categorical` *Create a Categorical distribution*

Description

Create a Categorical distribution

Usage

```
Categorical(outcomes, p = NULL)
```

Arguments

`outcomes`	A vector specifying the elements in the sample space. Can be numeric, factor, character, or logical.
`p`	A vector of success probabilities for each outcome. Each element of `p` can be any positive value – the vector gets normalized internally. Defaults to `NULL`, in which case the distribution is assumed to be uniform.

Value

A `Categorical` object.

See Also

Other discrete distributions: `Bernoulli`, `Binomial`, `Geometric`, `HyperGeometric`, `Multinomial`, `NegativeBinomial`, `Poisson`

Examples

```
set.seed(27)

X <- Categorical(1:3, p = c(0.4, 0.1, 0.5))
X

Y <- Categorical(LETTERS[1:4])
Y

random(X, 10)
random(Y, 10)

pdf(X, 1)
log_pdf(X, 1)

cdf(X, 1)
quantile(X, 0.5)
## Not run:
# cdfs are only defined for numeric sample spaces. this errors!
```

```
cdf(Y, "a")

# same for quantiles. this also errors!
quantile(Y, 0.7)

## End(Not run)
```

Cauchy *Create a Cauchy distribution*

Description

Note that the Cauchy distribution is the student's t distribution with one degree of freedom. The Cauchy distribution does not have a well defined mean or variance. Cauchy distributions often appear as priors in Bayesian contexts due to their heavy tails.

Usage

```
Cauchy(location = 0, scale = 1)
```

Arguments

`location`	The location parameter. Can be any real number. Defaults to `0`.
`scale`	The scale parameter. Must be greater than zero (?). Defaults to 1.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail and much greater clarity.

In the following, let X be a Cauchy variable with mean `location` = x_0 and `scale` = γ.

Support: R, the set of all real numbers

Mean: Undefined.

Variance: Undefined.

Probability density function (p.d.f):

$$f(x) = \frac{1}{\pi\gamma\left[1 + \left(\frac{x-x_0}{\gamma}\right)^2\right]}$$

Cumulative distribution function (c.d.f):

$$F(t) = \frac{1}{\pi}\arctan\left(\frac{t-x_0}{\gamma}\right) + \frac{1}{2}$$

Moment generating function (m.g.f):

Does not exist.

Value

A Cauchy object.

See Also

Other continuous distributions: Beta, ChiSquare, Exponential, FisherF, Gamma, LogNormal, Logistic, Normal, StudentsT, Tukey, Uniform, Weibull

Examples

```
set.seed(27)

X <- Cauchy(10, 0.2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 2)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

cdf — *Evaluate the probability density of a probability distribution*

Description

For discrete distributions, the probabilty mass function.

Usage

```
cdf(d, x, ...)
```

Arguments

d	A probability distribution object such as those created by a call to Bernoulli(), Beta(), or Binomial().
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
X <- Normal()

cdf(X, c(1, 2, 3, 4, 5))
```

---

cdf.Bernoulli *Evaluate the cumulative distribution function of a Bernoulli distribution*

Description

Evaluate the cumulative distribution function of a Bernoulli distribution

Usage

```
## S3 method for class 'Bernoulli'
cdf(d, x, ...)
```

Arguments

d	A Bernoulli object created by a call to Bernoulli().
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- Bernoulli(0.7)
X

random(X, 10)
pdf(X, 1)
log_pdf(X, 1)
cdf(X, 0)
quantile(X, 0.7)
```

```
cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

---

cdf.Beta *Evaluate the cumulative distribution function of a Beta distribution*

Description

Evaluate the cumulative distribution function of a Beta distribution

Usage

```
## S3 method for class 'Beta'
cdf(d, x, ...)
```

Arguments

d	A `Beta` object created by a call to `Beta()`.
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution `d`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

Examples

```
set.seed(27)

X <- Beta(1, 2)
X

random(X, 10)

pdf(X, 0.7)
log_pdf(X, 0.7)

cdf(X, 0.7)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

`cdf.Binomial` *Evaluate the cumulative distribution function of a Binomial distribution*

Description

Evaluate the cumulative distribution function of a Binomial distribution

Usage

```
## S3 method for class 'Binomial'
cdf(d, x, ...)
```

Arguments

d	A Binomial object created by a call to Binomial().
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- Binomial(10, 0.2)
X

random(X, 10)

pdf(X, 2L)
log_pdf(X, 2L)

cdf(X, 4L)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

cdf.Categorical — *Evaluate the cumulative distribution function of a Categorical distribution*

Description

Evaluate the cumulative distribution function of a Categorical distribution

Usage

```
## S3 method for class 'Categorical'
cdf(d, x, ...)
```

Arguments

d	A `Categorical` object created by a call to `Categorical()`.
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- Categorical(1:3, p = c(0.4, 0.1, 0.5))
X

Y <- Categorical(LETTERS[1:4])
Y

random(X, 10)
random(Y, 10)

pdf(X, 1)
log_pdf(X, 1)

cdf(X, 1)
quantile(X, 0.5)
## Not run:
# cdfs are only defined for numeric sample spaces. this errors!
cdf(Y, "a")

# same for quantiles. this also errors!
```

```
quantile(Y, 0.7)

## End(Not run)
```

---

cdf.Cauchy *Evaluate the cumulative distribution function of a Cauchy distribution*

Description

Evaluate the cumulative distribution function of a Cauchy distribution

Usage

```
## S3 method for class 'Cauchy'
cdf(d, x, ...)
```

Arguments

d	A Cauchy object created by a call to Cauchy().
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- Cauchy(10, 0.2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 2)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`cdf.ChiSquare` *Evaluate the cumulative distribution function of a chi square distribution*

Description

Evaluate the cumulative distribution function of a chi square distribution

Usage

```
## S3 method for class 'ChiSquare'
cdf(d, x, ...)
```

Arguments

d	A ChiSquare object created by a call to ChiSquare().
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- ChiSquare(5)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`cdf.Exponential` *Evaluate the cumulative distribution function of a Exponential distribution*

Description

Evaluate the cumulative distribution function of a Exponential distribution

Usage

```
## S3 method for class 'Exponential'
cdf(d, x, ...)
```

Arguments

d	A `Exponential` object created by a call to `Exponential()`.
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution `d`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

Examples

```
set.seed(27)

X <- Exponential(5)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`cdf.FisherF` *Evaluate the cumulative distribution function of an F distribution*

Description

Evaluate the cumulative distribution function of an F distribution

Usage

```
## S3 method for class 'FisherF'
cdf(d, x, ...)
```

Arguments

d	A FisherF object created by a call to FisherF().
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- FisherF(5, 10, 0.2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

cdf.Gamma *Evaluate the cumulative distribution function of a Gamma distribution*

Description

Evaluate the cumulative distribution function of a Gamma distribution

Usage

```
## S3 method for class 'Gamma'
cdf(d, x, ...)
```

Arguments

d	A `Gamma` object created by a call to `Gamma()`.
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution `d`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

Examples

```
set.seed(27)

X <- Gamma(5, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`cdf.Geometric` *Evaluate the cumulative distribution function of a Geometric distribution*

Description

Evaluate the cumulative distribution function of a Geometric distribution

Usage

```
## S3 method for class 'Geometric'
cdf(d, x, ...)
```

Arguments

d	A `Geometric` object created by a call to `Geometric()`.
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution `d`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

See Also

Other Geometric distribution: `pdf.Geometric`, `quantile.Geometric`, `random.Geometric`

Examples

```
set.seed(27)

X <- Geometric(0.3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

`cdf.HyperGeometric` *Evaluate the cumulative distribution function of a HyperGeometric distribution*

Description

Evaluate the cumulative distribution function of a HyperGeometric distribution

Usage

```
## S3 method for class 'HyperGeometric'
cdf(d, x, ...)
```

Arguments

d	A `HyperGeometric` object created by a call to `HyperGeometric()`.
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution `d`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

See Also

Other HyperGeometric distribution: `pdf.HyperGeometric`, `quantile.HyperGeometric`, `random.HyperGeometric`

Examples

```
set.seed(27)

X <- HyperGeometric(4, 5, 8)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

`cdf.Logistic` *Evaluate the cumulative distribution function of a Logistic distribution*

Description

Evaluate the cumulative distribution function of a Logistic distribution

Usage

```
## S3 method for class 'Logistic'
cdf(d, x, ...)
```

Arguments

d	A `Logistic` object created by a call to `Logistic()`.
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution `d`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

See Also

Other Logistic distribution: `pdf.Logistic`, `quantile.Logistic`, `random.Logistic`

Examples

```
set.seed(27)

X <- Logistic(2, 4)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

`cdf.LogNormal` *Evaluate the cumulative distribution function of a LogNormal distribution*

Description

Evaluate the cumulative distribution function of a LogNormal distribution

Usage

```
## S3 method for class 'LogNormal'
cdf(d, x, ...)
```

Arguments

d	A LogNormal object created by a call to `LogNormal()`.
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

See Also

Other LogNormal distribution: `fit_mle.LogNormal`, `pdf.LogNormal`, `quantile.LogNormal`, `random.LogNormal`

Examples

```
set.seed(27)

X <- LogNormal(0.3, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

cdf.NegativeBinomial *Evaluate the cumulative distribution function of a negative binomial distribution*

Description

Evaluate the cumulative distribution function of a negative binomial distribution

Usage

```
## S3 method for class 'NegativeBinomial'
cdf(d, x, ...)
```

Arguments

d	A NegativeBinomial object created by a call to NegativeBinomial().
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

See Also

Other NegativeBinomial distribution: pdf.NegativeBinomial, quantile.NegativeBinomial, random.NegativeBinomia

Examples

```
set.seed(27)

X <- NegativeBinomial(10, 0.3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

cdf.Normal *Evaluate the cumulative distribution function of a Normal distribution*

Description

Evaluate the cumulative distribution function of a Normal distribution

Usage

```
## S3 method for class 'Normal'
cdf(d, x, ...)
```

Arguments

d	A Normal object created by a call to Normal().
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

See Also

Other Normal distribution: fit_mle.Normal, pdf.Normal, quantile.Normal

Examples

```
set.seed(27)

X <- Normal(5, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

### example: calculating p-values for two-sided Z-test

# here the null hypothesis is H_0: mu = 3
# and we assume sigma = 2
```

```
# exactly the same as: Z <- Normal(0, 1)
Z <- Normal()

# data to test
x <- c(3, 7, 11, 0, 7, 0, 4, 5, 6, 2)
nx <- length(x)

# calculate the z-statistic
z_stat <- (mean(x) - 3) / (2 / sqrt(nx))
z_stat

# calculate the two-sided p-value
1 - cdf(Z, abs(z_stat)) + cdf(Z, -abs(z_stat))

# exactly equivalent to the above
2 * cdf(Z, -abs(z_stat))

# p-value for one-sided test
# H_0: mu <= 3   vs   H_A: mu > 3
1 - cdf(Z, z_stat)

# p-value for one-sided test
# H_0: mu >= 3   vs   H_A: mu < 3
cdf(Z, z_stat)

### example: calculating a 88 percent Z CI for a mean

# same `x` as before, still assume `sigma = 2`

# lower-bound
mean(x) - quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# upper-bound
mean(x) + quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# equivalent to
mean(x) + c(-1, 1) * quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# also equivalent to
mean(x) + quantile(Z, 0.12 / 2) * 2 / sqrt(nx)
mean(x) + quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

### generating random samples and plugging in ks.test()

set.seed(27)

# generate a random sample
ns <- random(Normal(3, 7), 26)

# test if sample is Normal(3, 7)
ks.test(ns, pnorm, mean = 3, sd = 7)
```

```
# test if sample is gamma(8, 3) using base R pgamma()
ks.test(ns, pgamma, shape = 8, rate = 3)

### MISC

# note that the cdf() and quantile() functions are inverses
cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

cdf.Poisson *Evaluate the cumulative distribution function of a Poisson distribution*

Description

Evaluate the cumulative distribution function of a Poisson distribution

Usage

```
## S3 method for class 'Poisson'
cdf(d, x, ...)
```

Arguments

d	A Poisson object created by a call to Poisson().
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- Poisson(2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

```
cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`cdf.StudentsT` *Evaluate the cumulative distribution function of a StudentsT distribution*

Description

Evaluate the cumulative distribution function of a StudentsT distribution

Usage

```
## S3 method for class 'StudentsT'
cdf(d, x, ...)
```

Arguments

d	A `StudentsT` object created by a call to `StudentsT()`.
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

See Also

Other StudentsT distribution: `pdf.StudentsT`, `quantile.StudentsT`, `random.StudentsT`

Examples

```
set.seed(27)

X <- StudentsT(3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

```
### example: calculating p-values for two-sided T-test

# here the null hypothesis is H_0: mu = 3

# data to test
x <- c(3, 7, 11, 0, 7, 0, 4, 5, 6, 2)
nx <- length(x)

# calculate the T-statistic
t_stat <- (mean(x) - 3) / (sd(x) / sqrt(nx))
t_stat

# null distribution of statistic depends on sample size!
T <- StudentsT(df = nx - 1)

# calculate the two-sided p-value
1 - cdf(T, abs(t_stat)) + cdf(T, -abs(t_stat))

# exactly equivalent to the above
2 * cdf(T, -abs(t_stat))

# p-value for one-sided test
# H_0: mu <= 3   vs   H_A: mu > 3
1 - cdf(T, t_stat)

# p-value for one-sided test
# H_0: mu >= 3   vs   H_A: mu < 3
cdf(T, t_stat)

### example: calculating a 88 percent T CI for a mean

# lower-bound
mean(x) - quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# upper-bound
mean(x) + quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# equivalent to
mean(x) + c(-1, 1) * quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# also equivalent to
mean(x) + quantile(T, 0.12 / 2) * sd(x) / sqrt(nx)
mean(x) + quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)
```

`cdf.Tukey` *Evaluate the cumulative distribution function of a Tukey distribution*

Description

Evaluate the cumulative distribution function of a Tukey distribution

Usage

```
## S3 method for class 'Tukey'
cdf(d, x, ...)
```

Arguments

d	A Tukey distribution created by a call to Tukey().
x	A vector of elements whose cumulative probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

See Also

Other Tukey distribution: quantile.Tukey

Examples

```
set.seed(27)

X <- Tukey(4L, 16L, 2L)
X

cdf(X, 4)
quantile(X, 0.7)
```

---

cdf.Uniform *Evaluate the cumulative distribution function of a continuous Uniform distribution*

Description

Evaluate the cumulative distribution function of a continuous Uniform distribution

Usage

```
## S3 method for class 'Uniform'
cdf(d, x, ...)
```

Arguments

- `d` — A `Uniform` object created by a call to `Uniform()`.
- `x` — A vector of elements whose cumulative probabilities you would like to determine given the distribution `d`.
- `...` — Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

Examples

```
set.seed(27)

X <- Uniform(1, 2)
X

random(X, 10)

pdf(X, 0.7)
log_pdf(X, 0.7)

cdf(X, 0.7)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

---

cdf.Weibull — *Evaluate the cumulative distribution function of a Weibull distribution*

Description

Evaluate the cumulative distribution function of a Weibull distribution

Usage

```
## S3 method for class 'Weibull'
cdf(d, x, ...)
```

Arguments

`d`	A `Weibull` object created by a call to `Weibull()`.
`x`	A vector of elements whose cumulative probabilities you would like to determine given the distribution `d`.
`...`	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

See Also

Other Weibull distribution: `pdf.Weibull`, `quantile.Weibull`, `random.Weibull`

Examples

```
set.seed(27)

X <- Weibull(0.3, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

ChiSquare — *Create a Chi-Square distribution*

Description

Chi-square distributions show up often in frequentist settings as the sampling distribution of test statistics, especially in maximum likelihood estimation settings.

Usage

```
ChiSquare(df)
```

Arguments

`df`	Degrees of freedom. Must be positive.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail and much greater clarity.

In the following, let X be a χ^2 random variable with `df` = k.

Support: R^+, the set of positive real numbers

Mean: k

Variance: $2k$

Probability density function (p.d.f):

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

Cumulative distribution function (c.d.f):

The cumulative distribution function has the form

$$F(t) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} dx$$

but this integral does not have a closed form solution and must be approximated numerically. The c.d.f. of a standard normal is sometimes called the "error function". The notation $\Phi(t)$ also stands for the c.d.f. of a standard normal evaluated at t. Z-tables list the value of $\Phi(t)$ for various t.

Moment generating function (m.g.f):

$$E(e^{tX}) = e^{\mu t + \sigma^2 t^2/2}$$

Value

A `ChiSquare` object.

Transformations

A squared standard `Normal()` distribution is equivalent to a χ_1^2 distribution with one degree of freedom. The χ^2 distribution is a special case of the `Gamma()` distribution with shape (TODO: check this) parameter equal to a half. Sums of χ^2 distributions are also distributed as χ^2 distributions, where the degrees of freedom of the contributing distributions get summed. The ratio of two χ^2 distributions is a `FisherF()` distribution. The ratio of a `Normal()` and the square root of a scaled `ChiSquare()` is a `StudentsT()` distribution.

See Also

Other continuous distributions: `Beta`, `Cauchy`, `Exponential`, `FisherF`, `Gamma`, `LogNormal`, `Logistic`, `Normal`, `StudentsT`, `Tukey`, `Uniform`, `Weibull`

Examples

```
set.seed(27)

X <- ChiSquare(5)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`Exponential` *Create a Exponential distribution*

Description

Create a Exponential distribution

Usage

```
Exponential(rate = 1)
```

Arguments

`rate`	The rate parameter. Can be any positive number. Defaults to 1.

Value

A `Exponential` object.

See Also

Other continuous distributions: `Beta`, `Cauchy`, `ChiSquare`, `FisherF`, `Gamma`, `LogNormal`, `Logistic`, `Normal`, `StudentsT`, `Tukey`, `Uniform`, `Weibull`

Examples

```
set.seed(27)

X <- Exponential(5)
X
```

```
random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

FisherF *Create an F distribution*

Description

Create an F distribution

Usage

```
FisherF(df1, df2, lambda = 0)
```

Arguments

df1	Numerator degrees of freedom. Can be any positive number.
df2	Denominator degrees of freedom. Can be any positive number.
lambda	Non-centrality parameter. Can be any positive number. Defaults to `0`.

Value

A `FisherF` object.

See Also

Other continuous distributions: `Beta`, `Cauchy`, `ChiSquare`, `Exponential`, `Gamma`, `LogNormal`, `Logistic`, `Normal`, `StudentsT`, `Tukey`, `Uniform`, `Weibull`

Examples

```
set.seed(27)

X <- FisherF(5, 10, 0.2)
X

random(X, 10)

pdf(X, 2)
```

```
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`fit_mle` *Fit a distribution to data*

Description

Approximates an empirical distribution with a theoretical one

Usage

```
fit_mle(d, x, ...)
```

Arguments

d	A probability distribution object such as those created by a call to `Bernoulli()`, `Beta()`, or `Binomial()`.
x	A vector of data to compute the likelihood.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A distribution (the same kind as `d`) where the parameters are the MLE estimates based on `x`.

Examples

```
X <- Normal()

fit_mle(X, c(-1, 0, 0, 0, 3))
```

fit_mle.Bernoulli — *Fit a Bernoulli distribution to data*

Description

Fit a Bernoulli distribution to data

Usage

```
## S3 method for class 'Bernoulli'
fit_mle(d, x, ...)
```

Arguments

d	A Bernoulli object.
x	A vector of zeroes and ones.
...	Unused.

Value

a Bernoulli object

fit_mle.Binomial — *Fit a Binomial distribution to data*

Description

The fit distribution will inherit the same size parameter as the Binomial object passed.

Usage

```
## S3 method for class 'Binomial'
fit_mle(d, x, ...)
```

Arguments

d	A Binomial object.
x	A vector of zeroes and ones.
...	Unused.

Value

a Binomial object

fit_mle.Exponential — *Fit an Exponential distribution to data*

Description

Fit an Exponential distribution to data

Usage

```
## S3 method for class 'Exponential'
fit_mle(d, x, ...)
```

Arguments

d	An `Exponential` object created by a call to `Exponential()`.
x	A vector of data.
...	Unused.

Value

An `Exponential` object.

fit_mle.Gamma — *Fit a Gamma distribution to data*

Description

Fit a Gamma distribution to data

Usage

```
## S3 method for class 'Gamma'
fit_mle(d, x, ...)
```

Arguments

d	A `Gamma` object created by a call to `Gamma()`.
x	A vector to fit the Gamma distribution to.
...	Unused.

Value

a `Gamma` object

fit_mle.Geometric *Fit a Geometric distribution to data*

Description

Fit a Geometric distribution to data

Usage

```
## S3 method for class 'Geometric'
fit_mle(d, x, ...)
```

Arguments

d	A Geometric object.
x	A vector of zeroes and ones.
...	Unused.

Value

a Geometric object

fit_mle.LogNormal *Fit a Log Normal distribution to data*

Description

Fit a Log Normal distribution to data

Usage

```
## S3 method for class 'LogNormal'
fit_mle(d, x, ...)
```

Arguments

d	A LogNormal object created by a call to LogNormal().
x	A vector of data.
...	Unused.

Value

A LogNormal object.

See Also

Other LogNormal distribution: cdf.LogNormal, pdf.LogNormal, quantile.LogNormal, random.LogNormal `fit_mle.Normal` *Fit a Normal distribution to data*

Description

Fit a Normal distribution to data

Usage

```
## S3 method for class 'Normal'
fit_mle(d, x, ...)
```

Arguments

d	A Normal object created by a call to Normal().
x	A vector of data.
...	Unused.

Value

A Normal object.

See Also

Other Normal distribution: cdf.Normal, pdf.Normal, quantile.Normal

`fit_mle.Poisson` *Fit an Poisson distribution to data*

Description

Fit an Poisson distribution to data

Usage

```
## S3 method for class 'Poisson'
fit_mle(d, x, ...)
```

Arguments

d	An Poisson object created by a call to Poisson().
x	A vector of data.
...	Unused.

Value

An Poisson object.

Gamma *Create a Gamma distribution*

Description

Several important distributions are special cases of the Gamma distribution. When the shape parameter is 1, the Gamma is an exponential distribution with parameter $1/\beta$. When the $shape = n/2$ and $rate = 1/2$, the Gamma is a equivalent to a chi squared distribution with n degrees of freedom. Moreover, if we have X_1 is $Gamma(\alpha_1, \beta)$ and X_2 is $Gamma(\alpha_2, \beta)$, a function of these two variables of the form $\frac{X_1}{X_1+X_2}$ $Beta(\alpha_1, \alpha_2)$. This last property frequently appears in another distributions, and it has extensively been used in multivariate methods. More about the Gamma distribution will be added soon.

Usage

```
Gamma(shape, rate = 1)
```

Arguments

shape	The shape parameter. Can be any positive number.
rate	The rate parameter. Can be any positive number. Defaults to 1.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail.

In the following, let X be a Gamma random variable with parameters `shape` = α and `rate` = β.

Support: $x \in (0, \infty)$

Mean: $\frac{\alpha}{\beta}$

Variance: $\frac{\alpha}{\beta^2}$

Probability density function (p.m.f):

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

Cumulative distribution function (c.d.f):

$$f(x) = \frac{\Gamma(\alpha, \beta x)}{\Gamma \alpha}$$

Moment generating function (m.g.f):

$$E(e^{tX}) = \Big(\frac{\beta}{\beta - t}\Big)^\alpha, \, t < \beta$$

Value

A Gamma object.

See Also

Other continuous distributions: Beta, Cauchy, ChiSquare, Exponential, FisherF, LogNormal, Logistic, Normal, StudentsT, Tukey, Uniform, Weibull

Examples

```
set.seed(27)

X <- Gamma(5, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

Geometric *Create a Geometric distribution*

Description

The Geometric distribution can be thought of as a generalization of the Bernoulli() distribution where we ask: "if I keep flipping a coin with probability p of heads, what is the probability I need k flips before I get my first heads?" The Geometric distribution is a special case of Negative Binomial distribution.

Usage

```
Geometric(p = 0.5)
```

Arguments

p	The success probability for the distribution. p can be any value in [0,1], and defaults to 0.5.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail and much greater clarity.

In the following, let X be a Geometric random variable with success probability p = p. Note that there are multiple parameterizations of the Geometric distribution.

Support: 0 < p < 1, $x = 0, 1, \ldots$

Mean: $\frac{1-p}{p}$

Variance: $\frac{1-p}{p^2}$

Probability mass function (p.m.f):

$$P(X = x) = p(1-p)^x,$$

Cumulative distribution function (c.d.f):

$$P(X \le x) = 1 - (1-p)^{x+1}$$

Moment generating function (m.g.f):

$$E(e^{tX}) = \frac{pe^t}{1 - (1-p)e^t}$$

Value

A `Geometric` object.

See Also

Other discrete distributions: `Bernoulli`, `Binomial`, `Categorical`, `HyperGeometric`, `Multinomial`, `NegativeBinomial`, `Poisson`

Examples

```
set.seed(27)

X <- Geometric(0.3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

HyperGeometric *Create a HyperGeometric distribution*

Description

To understand the HyperGeometric distribution, consider a set of r objects, of which m are of the type I and n are of the type II. A sample with size k ($k < r$) with no replacement is randomly chosen. The number of observed type I elements observed in this sample is set to be our random variable X. For example, consider that in a set of 20 car parts, there are 4 that are defective (type I). If we take a sample of size 5 from those car parts, the probability of finding 2 that are defective will be given by the HyperGeometric distribution (needs double checking).

Usage

```
HyperGeometric(m, n, k)
```

Arguments

m	The number of type I elements available.
n	The number of type II elements available.
k	The size of the sample taken.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail and much greater clarity.

In the following, let X be a HyperGeometric random variable with success probability p = $p = m/(m+n)$.

Support: $x \in \{\max(0, k-(n-m), \dots, \min(k, m)\}$

Mean: $\frac{km}{n+m} = kp$

Variance: $\frac{km(n)(n+m-k)}{(n+m)^2(n+m-1)} = kp(1-p)(1-\frac{k-1}{m+n-1})$

Probability mass function (p.m.f):

$$P(X = x) = \frac{\binom{m}{x}\binom{n}{k-x}}{\binom{m+n}{k}}$$

Cumulative distribution function (c.d.f):

$$P(X \le k) \approx \Phi\Big(\frac{x - kp}{\sqrt{kp(1-p)}}\Big)$$

Moment generating function (m.g.f):

Not useful.

Value

A HyperGeometric object.

See Also

Other discrete distributions: Bernoulli, Binomial, Categorical, Geometric, Multinomial, NegativeBinomial, Poisson

Examples

```
set.seed(27)

X <- HyperGeometric(4, 5, 8)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

---

is_distribution *Is an object a distribution?*

Description

is_distribution tests if x inherits from "distribution".

Usage

```
is_distribution(x)
```

Arguments

x	An object to test.

Examples

```
Z <- Normal()

is_distribution(Z)
is_distribution(1L)
```

`likelihood` *Compute the likelihood of a probability distribution given data*

Description

Compute the likelihood of a probability distribution given data

Usage

```
likelihood(d, x, ...)
```

Arguments

`d`	A probability distribution object such as those created by a call to `Bernoulli()`, `Beta()`, or `Binomial()`.
`x`	A vector of data to compute the likelihood.
`...`	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

the likelihood

Examples

```
X <- Normal()

likelihood(X, c(-1, 0, 0, 0, 3))
```

`Logistic` *Create a Logistic distribution*

Description

A continuous distribution on the real line. For binary outcomes the model given by $P(Y = 1|X) = F(X\beta)$ where F is the Logistic `cdf()` is called *logistic regression*.

Usage

```
Logistic(location = 0, scale = 1)
```

Arguments

`location`	The location parameter for the distribution. For Logistic distributions, the location parameter is the mean, median and also mode. Defaults to zero.
`scale`	The scale parameter for the distribution. Defaults to one.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail and much greater clarity.

In the following, let X be a Logistic random variable with `location` = μ and `scale` = s.

Support: R, the set of all real numbers

Mean: μ

Variance: $s^2\pi^2/3$

Probability density function (p.d.f):

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

Cumulative distribution function (c.d.f):

$$F(t) = \frac{1}{1 + e^{-(\frac{t-\mu}{s})}}$$

Moment generating function (m.g.f):

$$E(e^{tX}) = e^{\mu t}\beta(1 - st, 1 + st)$$

where $\beta(x, y)$ is the Beta function.

Value

A `Logistic` object.

See Also

Other continuous distributions: `Beta`, `Cauchy`, `ChiSquare`, `Exponential`, `FisherF`, `Gamma`, `LogNormal`, `Normal`, `StudentsT`, `Tukey`, `Uniform`, `Weibull`

Examples

```
set.seed(27)

X <- Logistic(2, 4)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

LogNormal *Create a LogNormal distribution*

Description

A random variable created by exponentiating a Normal() distribution. Taking the log of LogNormal data returns in Normal() data.

Usage

```
LogNormal(log_mu = 0, log_sigma = 1)
```

Arguments

log_mu	The location parameter, written μ in textbooks. Can be any real number. Defaults to 0.
log_sigma	The scale parameter, written σ in textbooks. Can be any positive real number. Defaults to 1.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail and much greater clarity.

In the following, let X be a LogNormal random variable with success probability p = p.

Support: R^+

Mean: $\exp(\mu + \sigma^2/2)$

Variance: $[\exp(\sigma^2) - 1]\exp(2\mu + \sigma^2)$

Probability density function (p.d.f):

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

Cumulative distribution function (c.d.f):

$$F(x) = \frac{1}{2} + \frac{1}{2\sqrt{pi}}\int_{-x}^{x} e^{-t^2}dt$$

Moment generating function (m.g.f): Undefined.

Value

A LogNormal object.

See Also

Other continuous distributions: Beta, Cauchy, ChiSquare, Exponential, FisherF, Gamma, Logistic, Normal, StudentsT, Tukey, Uniform, Weibull

Examples

```
set.seed(27)

X <- LogNormal(0.3, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

---

log_likelihood *Compute the log-likelihood of a probability distribution given data*

Description

Compute the log-likelihood of a probability distribution given data

Usage

```
log_likelihood(d, x, ...)
```

Arguments

d	A probability distribution object such as those created by a call to `Bernoulli()`, `Beta()`, or `Binomial()`.
x	A vector of data to compute the likelihood.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

the log-likelihood

Examples

```
X <- Normal()

log_likelihood(X, c(-1, 0, 0, 0, 3))
```

`Multinomial` *Create a Multinomial distribution*

Description

The multinomial distribution is a generalization of the binomial distribution to multiple categories. It is perhaps easiest to think that we first extend a `Bernoulli()` distribution to include more than two categories, resulting in a `Categorical()` distribution. We then extend repeat the Categorical experiment several (n) times.

Usage

```
Multinomial(size, p)
```

Arguments

`size`	The number of trials. Must be an integer greater than or equal to one. When `size = 1L`, the Multinomial distribution reduces to the categorical distribution (also called the discrete uniform). Often called `n` in textbooks.
`p`	A vector of success probabilities for each trial. `p` can take on any positive value, and the vector is normalized internally.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail and much greater clarity.

In the following, let $X = (X_1, ..., X_k)$ be a Multinomial random variable with success probability `p` = p. Note that p is vector with k elements that sum to one. Assume that we repeat the Categorical experiment `size` = n times.

Support: Each X_i is in $0, 1, 2, ..., n$.

Mean: The mean of X_i is np_i.

Variance: The variance of X_i is $np_i(1 - p_i)$. For $i \neq j$, the covariance of X_i and X_j is $-np_ip_j$.

Probability mass function (p.m.f):

$$P(X_1 = x_1, ..., X_k = x_k) = \frac{n!}{x_1!x_2!...x_k!} p_1^{x_1} \cdot p_2^{x_2} \cdot ... \cdot p_k^{x_k}$$

Cumulative distribution function (c.d.f):

Omitted for multivariate random variables for the time being.

Moment generating function (m.g.f):

$$E(e^{tX}) = (\sum_{i=1}^{k} p_i e^{t_i})^n$$

Value

A Multinomial object.

See Also

Other discrete distributions: Bernoulli, Binomial, Categorical, Geometric, HyperGeometric, NegativeBinomial, Poisson

Examples

```
set.seed(27)

X <- Multinomial(size = 5, p = c(0.3, 0.4, 0.2, 0.1))
X

random(X, 10)

# pdf(X, 2)
# log_pdf(X, 2)
```

NegativeBinomial *Create a Negative Binomial distribution*

Description

A generalization of the geometric distribution. It is the number of successes in a sequence of i.i.d. Bernoulli trials before a specified number (r) of failures occurs.

Usage

```
NegativeBinomial(size, p = 0.5)
```

Arguments

size	The number of failures (an integer greater than 0) until the experiment is stopped. Denoted r below.
p	The success probability for a given trial. p can be any value in [0,1], and defaults to 0.5.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail and much greater clarity.

In the following, let X be a Negative Binomial random variable with success probability p = p.

Support: $\{0, 1, 2, 3, ...\}$

Mean: $\frac{pr}{1-p}$

Variance: $\frac{pr}{(1-p)^2}$

Probability mass function (p.m.f):

$$f(k) = \binom{k+r-1}{k} \cdot (1-p)^r p^k$$

Cumulative distribution function (c.d.f):

Too nasty, ommited.

Moment generating function (m.g.f):

$$\frac{(1-p)^r}{(1-pe^t)^r}, t < -\log p$$

Value

A `NegativeBinomial` object.

See Also

Other discrete distributions: `Bernoulli`, `Binomial`, `Categorical`, `Geometric`, `HyperGeometric`, `Multinomial`, `Poisson`

Examples

```
set.seed(27)

X <- NegativeBinomial(10, 0.3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

`Normal` *Create a Normal distribution*

Description

The Normal distribution is ubiquituous in statistics, partially because of the central limit theorem, which states that sums of i.i.d. random variables eventually become Normal. Linear transformations of Normal random variables result in new random variables that are also Normal. If you are taking an intro stats course, you'll likely use the Normal distribution for Z-tests and in simple linear regression. Under regularity conditions, maximum likelihood estimators are asymptotically Normal. The Normal distribution is also called the gaussian distribution.

Usage

```
Normal(mu = 0, sigma = 1)
```

Arguments

mu	The location parameter, written μ in textbooks, which is also the mean of the distribution. Can be any real number. Defaults to `0`.
sigma	The scale parameter, written σ in textbooks, which is also the **standard deviation** of the distribution. Can be any positive number. Defaults to 1. If you would like a Normal distribution with **variance** σ^2, be sure to take the square root, as this is a common source of errors.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail and much greater clarity.

In the following, let X be a Normal random variable with mean `mu` = μ and standard deviation `sigma` = σ.

Support: R, the set of all real numbers

Mean: μ

Variance: σ^2

Probability density function (p.d.f):

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

Cumulative distribution function (c.d.f):

The cumulative distribution function has the form

$$F(t) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} dx$$

but this integral does not have a closed form solution and must be approximated numerically. The c.d.f. of a standard Normal is sometimes called the "error function". The notation $\Phi(t)$ also stands for the c.d.f. of a standard Normal evaluated at t. Z-tables list the value of $\Phi(t)$ for various t.

Moment generating function (m.g.f):

$$E(e^{tX}) = e^{\mu t + \sigma^2 t^2/2}$$

Value

A `Normal` object.

See Also

Other continuous distributions: `Beta`, `Cauchy`, `ChiSquare`, `Exponential`, `FisherF`, `Gamma`, `LogNormal`, `Logistic`, `StudentsT`, `Tukey`, `Uniform`, `Weibull`

Examples

```
set.seed(27)

X <- Normal(5, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

### example: calculating p-values for two-sided Z-test

# here the null hypothesis is H_0: mu = 3
# and we assume sigma = 2

# exactly the same as: Z <- Normal(0, 1)
Z <- Normal()

# data to test
x <- c(3, 7, 11, 0, 7, 0, 4, 5, 6, 2)
nx <- length(x)

# calculate the z-statistic
z_stat <- (mean(x) - 3) / (2 / sqrt(nx))
z_stat

# calculate the two-sided p-value
1 - cdf(Z, abs(z_stat)) + cdf(Z, -abs(z_stat))

# exactly equivalent to the above
2 * cdf(Z, -abs(z_stat))

# p-value for one-sided test
# H_0: mu <= 3   vs   H_A: mu > 3
1 - cdf(Z, z_stat)

# p-value for one-sided test
# H_0: mu >= 3   vs   H_A: mu < 3
cdf(Z, z_stat)

### example: calculating a 88 percent Z CI for a mean

# same `x` as before, still assume `sigma = 2`

# lower-bound
mean(x) - quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)
```

```
# upper-bound
mean(x) + quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# equivalent to
mean(x) + c(-1, 1) * quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# also equivalent to
mean(x) + quantile(Z, 0.12 / 2) * 2 / sqrt(nx)
mean(x) + quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

### generating random samples and plugging in ks.test()

set.seed(27)

# generate a random sample
ns <- random(Normal(3, 7), 26)

# test if sample is Normal(3, 7)
ks.test(ns, pnorm, mean = 3, sd = 7)

# test if sample is gamma(8, 3) using base R pgamma()
ks.test(ns, pgamma, shape = 8, rate = 3)

### MISC

# note that the cdf() and quantile() functions are inverses
cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

pdf *Evaluate the probability density of a probability distribution*

Description

For discrete distributions, the probabilty mass function. `pmf()` is an alias.

Usage

```
pdf(d, x, ...)

log_pdf(d, x, ...)

pmf(d, x, ...)
```

Arguments

d	A probability distribution object such as those created by a call to `Bernoulli()`, `Beta()`, or `Binomial()`.

x	A vector of elements whose probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
X <- Normal()

pdf(X, c(1, 2, 3, 4, 5))
pmf(X, c(1, 2, 3, 4, 5))

log_pdf(X, c(1, 2, 3, 4, 5))
```

---

pdf.Bernoulli — *Evaluate the probability mass function of a Bernoulli distribution*

Description

Evaluate the probability mass function of a Bernoulli distribution

Usage

```
## S3 method for class 'Bernoulli'
pdf(d, x, ...)

## S3 method for class 'Bernoulli'
log_pdf(d, x, ...)
```

Arguments

d	A Bernoulli object created by a call to Bernoulli().
x	A vector of elements whose probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- Bernoulli(0.7)
X

random(X, 10)
pdf(X, 1)
log_pdf(X, 1)
cdf(X, 0)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

---

pdf.Beta *Evaluate the probability mass function of a Beta distribution*

Description

Evaluate the probability mass function of a Beta distribution

Usage

```
## S3 method for class 'Beta'
pdf(d, x, ...)

## S3 method for class 'Beta'
log_pdf(d, x, ...)
```

Arguments

d	A `Beta` object created by a call to `Beta()`.
x	A vector of elements whose probabilities you would like to determine given the distribution `d`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

Examples

```
set.seed(27)

X <- Beta(1, 2)
X

random(X, 10)

pdf(X, 0.7)
log_pdf(X, 0.7)

cdf(X, 0.7)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

---

pdf.Binomial — *Evaluate the probability mass function of a Binomial distribution*

Description

Evaluate the probability mass function of a Binomial distribution

Usage

```
## S3 method for class 'Binomial'
pdf(d, x, ...)

## S3 method for class 'Binomial'
log_pdf(d, x, ...)
```

Arguments

`d`	A `Binomial` object created by a call to `Binomial()`.
`x`	A vector of elements whose probabilities you would like to determine given the distribution `d`.
`...`	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

Examples

```
set.seed(27)

X <- Binomial(10, 0.2)
X

random(X, 10)

pdf(X, 2L)
log_pdf(X, 2L)

cdf(X, 4L)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

pdf.Categorical — *Evaluate the probability mass function of a Categorical discrete distribution*

Description

Evaluate the probability mass function of a Categorical discrete distribution

Usage

```
## S3 method for class 'Categorical'
pdf(d, x, ...)

## S3 method for class 'Categorical'
log_pdf(d, x, ...)
```

Arguments

d	A Categorical object created by a call to Categorical().
x	A vector of elements whose probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- Categorical(1:3, p = c(0.4, 0.1, 0.5))
X

Y <- Categorical(LETTERS[1:4])
Y

random(X, 10)
random(Y, 10)

pdf(X, 1)
log_pdf(X, 1)

cdf(X, 1)
quantile(X, 0.5)
## Not run:
# cdfs are only defined for numeric sample spaces. this errors!
cdf(Y, "a")

# same for quantiles. this also errors!
quantile(Y, 0.7)

## End(Not run)
```

---

pdf.Cauchy   *Evaluate the probability mass function of a Cauchy distribution*

Description

Evaluate the probability mass function of a Cauchy distribution

Usage

```
## S3 method for class 'Cauchy'
pdf(d, x, ...)

## S3 method for class 'Cauchy'
log_pdf(d, x, ...)
```

Arguments

d	A Cauchy object created by a call to `Cauchy()`.
x	A vector of elements whose probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- Cauchy(10, 0.2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 2)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

pdf.ChiSquare *Evaluate the probability mass function of a chi square distribution*

Description

Evaluate the probability mass function of a chi square distribution

Usage

```
## S3 method for class 'ChiSquare'
pdf(d, x, ...)

## S3 method for class 'ChiSquare'
log_pdf(d, x, ...)
```

Arguments

d	A ChiSquare object created by a call to ChiSquare().
x	A vector of elements whose probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- ChiSquare(5)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`pdf.Exponential` *Evaluate the probability mass function of a Exponential distribution*

Description

Evaluate the probability mass function of a Exponential distribution

Usage

```
## S3 method for class 'Exponential'
pdf(d, x, ...)

## S3 method for class 'Exponential'
log_pdf(d, x, ...)
```

Arguments

`d`	A `Exponential` object created by a call to `Exponential()`.
`x`	A vector of elements whose probabilities you would like to determine given the distribution `d`.
`...`	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

Examples

```
set.seed(27)

X <- Exponential(5)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

pdf.FisherF *Evaluate the probability mass function of an F distribution*

Description

Evaluate the probability mass function of an F distribution

Usage

```
## S3 method for class 'FisherF'
pdf(d, x, ...)

## S3 method for class 'FisherF'
log_pdf(d, x, ...)
```

Arguments

d	A FisherF object created by a call to FisherF().
x	A vector of elements whose probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- FisherF(5, 10, 0.2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

pdf.Gamma *Evaluate the probability mass function of a Gamma distribution*

Description

Evaluate the probability mass function of a Gamma distribution

Usage

```
## S3 method for class 'Gamma'
pdf(d, x, ...)

## S3 method for class 'Gamma'
log_pdf(d, x, ...)
```

Arguments

d	A Gamma object created by a call to Gamma().
x	A vector of elements whose probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- Gamma(5, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

pdf.Geometric *Evaluate the probability mass function of a Geometric distribution*

Description

Please see the documentation of Geometric() for some properties of the Geometric distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Usage

```
## S3 method for class 'Geometric'
pdf(d, x, ...)

## S3 method for class 'Geometric'
log_pdf(d, x, ...)
```

Arguments

d	A Geometric object created by a call to Geometric().
x	A vector of elements whose probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

See Also

Other Geometric distribution: cdf.Geometric, quantile.Geometric, random.Geometric

Examples

```
set.seed(27)

X <- Geometric(0.3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

---

pdf.HyperGeometric — *Evaluate the probability mass function of a HyperGeometric distribution*

Description

Please see the documentation of `HyperGeometric()` for some properties of the HyperGeometric distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Usage

```
## S3 method for class 'HyperGeometric'
pdf(d, x, ...)

## S3 method for class 'HyperGeometric'
log_pdf(d, x, ...)
```

Arguments

d	A `HyperGeometric` object created by a call to `HyperGeometric()`.
x	A vector of elements whose probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

See Also

Other HyperGeometric distribution: `cdf.HyperGeometric`, `quantile.HyperGeometric`, `random.HyperGeometric`

Examples

```
set.seed(27)

X <- HyperGeometric(4, 5, 8)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

`pdf.Logistic` *Evaluate the probability mass function of a Logistic distribution*

Description

Please see the documentation of `Logistic()` for some properties of the Logistic distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Usage

```
## S3 method for class 'Logistic'
pdf(d, x, ...)

## S3 method for class 'Logistic'
log_pdf(d, x, ...)
```

Arguments

`d`	A `Logistic` object created by a call to `Logistic()`.
`x`	A vector of elements whose probabilities you would like to determine given the distribution `d`.
`...`	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

See Also

Other Logistic distribution: `cdf.Logistic`, `quantile.Logistic`, `random.Logistic`

Examples

```
set.seed(27)

X <- Logistic(2, 4)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

---

pdf.LogNormal — *Evaluate the probability mass function of a LogNormal distribution*

Description

Please see the documentation of `LogNormal()` for some properties of the LogNormal distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Usage

```
## S3 method for class 'LogNormal'
pdf(d, x, ...)

## S3 method for class 'LogNormal'
log_pdf(d, x, ...)
```

Arguments

d	A `LogNormal` object created by a call to `LogNormal()`.
x	A vector of elements whose probabilities you would like to determine given the distribution `d`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

See Also

Other LogNormal distribution: `cdf.LogNormal`, `fit_mle.LogNormal`, `quantile.LogNormal`, `random.LogNormal`

Examples

```
set.seed(27)

X <- LogNormal(0.3, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

pdf.Multinomial *Evaluate the probability mass function of a Multinomial distribution*

Description

Please see the documentation of `Multinomial()` for some properties of the Multinomial distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Usage

```
## S3 method for class 'Multinomial'
pdf(d, x, ...)

## S3 method for class 'Multinomial'
log_pdf(d, x, ...)
```

Arguments

d	A `Multinomial` object created by a call to `Multinomial()`.
x	A vector of elements whose probabilities you would like to determine given the distribution `d`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

See Also

Other Multinomial distribution: `random.Multinomial`

Examples

```
set.seed(27)

X <- Multinomial(size = 5, p = c(0.3, 0.4, 0.2, 0.1))
X

random(X, 10)

# pdf(X, 2)
# log_pdf(X, 2)
```

`pdf.NegativeBinomial` *Evaluate the probability mass function of a NegativeBinomial distribution*

Description

Evaluate the probability mass function of a NegativeBinomial distribution

Usage

```
## S3 method for class 'NegativeBinomial'
pdf(d, x, ...)

## S3 method for class 'NegativeBinomial'
log_pdf(d, x, ...)
```

Arguments

d	A `NegativeBinomial` object created by a call to `NegativeBinomial()`.
x	A vector of elements whose probabilities you would like to determine given the distribution `d`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

See Also

Other NegativeBinomial distribution: `cdf.NegativeBinomial`, `quantile.NegativeBinomial`, `random.NegativeBinomia`

Examples

```
set.seed(27)

X <- NegativeBinomial(10, 0.3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

`pdf.Normal` *Evaluate the probability mass function of a Normal distribution*

Description

Please see the documentation of `Normal()` for some properties of the Normal distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Usage

```
## S3 method for class 'Normal'
pdf(d, x, ...)

## S3 method for class 'Normal'
log_pdf(d, x, ...)
```

Arguments

d	A `Normal` object created by a call to `Normal()`.
x	A vector of elements whose probabilities you would like to determine given the distribution `d`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

See Also

Other Normal distribution: `cdf.Normal`, `fit_mle.Normal`, `quantile.Normal`

Examples

```
set.seed(27)

X <- Normal(5, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

### example: calculating p-values for two-sided Z-test

# here the null hypothesis is H_0: mu = 3
# and we assume sigma = 2

# exactly the same as: Z <- Normal(0, 1)
Z <- Normal()

# data to test
x <- c(3, 7, 11, 0, 7, 0, 4, 5, 6, 2)
nx <- length(x)

# calculate the z-statistic
z_stat <- (mean(x) - 3) / (2 / sqrt(nx))
z_stat

# calculate the two-sided p-value
1 - cdf(Z, abs(z_stat)) + cdf(Z, -abs(z_stat))

# exactly equivalent to the above
2 * cdf(Z, -abs(z_stat))

# p-value for one-sided test
# H_0: mu <= 3   vs   H_A: mu > 3
1 - cdf(Z, z_stat)

# p-value for one-sided test
# H_0: mu >= 3   vs   H_A: mu < 3
cdf(Z, z_stat)

### example: calculating a 88 percent Z CI for a mean

# same `x` as before, still assume `sigma = 2`

# lower-bound
mean(x) - quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)
```

```
# upper-bound
mean(x) + quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# equivalent to
mean(x) + c(-1, 1) * quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# also equivalent to
mean(x) + quantile(Z, 0.12 / 2) * 2 / sqrt(nx)
mean(x) + quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

### generating random samples and plugging in ks.test()

set.seed(27)

# generate a random sample
ns <- random(Normal(3, 7), 26)

# test if sample is Normal(3, 7)
ks.test(ns, pnorm, mean = 3, sd = 7)

# test if sample is gamma(8, 3) using base R pgamma()
ks.test(ns, pgamma, shape = 8, rate = 3)

### MISC

# note that the cdf() and quantile() functions are inverses
cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

pdf.Poisson *Evaluate the probability mass function of a Poisson distribution*

Description

Evaluate the probability mass function of a Poisson distribution

Usage

```
## S3 method for class 'Poisson'
pdf(d, x, ...)

## S3 method for class 'Poisson'
log_pdf(d, x, ...)
```

Arguments

d	A Poisson object created by a call to `Poisson()`.
x	A vector of elements whose probabilities you would like to determine given the distribution d.

`...`	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

Examples

```
set.seed(27)

X <- Poisson(2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

pdf.StudentsT *Evaluate the probability mass function of a StudentsT distribution*

Description

Please see the documentation of `StudentsT()` for some properties of the StudentsT distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Usage

```
## S3 method for class 'StudentsT'
pdf(d, x, ...)

## S3 method for class 'StudentsT'
log_pdf(d, x, ...)
```

Arguments

`d`	A `StudentsT` object created by a call to `StudentsT()`.
`x`	A vector of elements whose probabilities you would like to determine given the distribution `d`.
`...`	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

See Also

Other StudentsT distribution: cdf.StudentsT, quantile.StudentsT, random.StudentsT

Examples

```
set.seed(27)

X <- StudentsT(3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

### example: calculating p-values for two-sided T-test

# here the null hypothesis is H_0: mu = 3

# data to test
x <- c(3, 7, 11, 0, 7, 0, 4, 5, 6, 2)
nx <- length(x)

# calculate the T-statistic
t_stat <- (mean(x) - 3) / (sd(x) / sqrt(nx))
t_stat

# null distribution of statistic depends on sample size!
T <- StudentsT(df = nx - 1)

# calculate the two-sided p-value
1 - cdf(T, abs(t_stat)) + cdf(T, -abs(t_stat))

# exactly equivalent to the above
2 * cdf(T, -abs(t_stat))

# p-value for one-sided test
# H_0: mu <= 3   vs   H_A: mu > 3
1 - cdf(T, t_stat)

# p-value for one-sided test
# H_0: mu >= 3   vs   H_A: mu < 3
cdf(T, t_stat)
```

```
### example: calculating a 88 percent T CI for a mean

# lower-bound
mean(x) - quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# upper-bound
mean(x) + quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# equivalent to
mean(x) + c(-1, 1) * quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# also equivalent to
mean(x) + quantile(T, 0.12 / 2) * sd(x) / sqrt(nx)
mean(x) + quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)
```

`pdf.Uniform` *Evaluate the probability mass function of a continuous Uniform distribution*

Description

Evaluate the probability mass function of a continuous Uniform distribution

Usage

```
## S3 method for class 'Uniform'
pdf(d, x, ...)

## S3 method for class 'Uniform'
log_pdf(d, x, ...)
```

Arguments

d	A `Uniform` object created by a call to `Uniform()`.
x	A vector of elements whose probabilities you would like to determine given the distribution d.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of x.

Examples

```
set.seed(27)

X <- Uniform(1, 2)
X

random(X, 10)

pdf(X, 0.7)
log_pdf(X, 0.7)

cdf(X, 0.7)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

`pdf.Weibull` *Evaluate the probability mass function of a Weibull distribution*

Description

Please see the documentation of `Weibull()` for some properties of the Weibull distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Usage

```
## S3 method for class 'Weibull'
pdf(d, x, ...)

## S3 method for class 'Weibull'
log_pdf(d, x, ...)
```

Arguments

`d`	A `Weibull` object created by a call to `Weibull()`.
`x`	A vector of elements whose probabilities you would like to determine given the distribution `d`.
`...`	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of probabilities, one for each element of `x`.

See Also

Other Weibull distribution: `cdf.Weibull`, `quantile.Weibull`, `random.Weibull`

Examples

```
set.seed(27)

X <- Weibull(0.3, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

`Poisson` *Create a Poisson distribution*

Description

Poisson distributions are frequently used to model counts.

Usage

```
Poisson(lambda)
```

Arguments

`lambda`	The shape parameter, which is also the mean and the variance of the distribution. Can be any positive number.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail.

In the following, let X be a Poisson random variable with parameter `lamdba` = λ.

Support: $\{0, 1, 2, 3, ...\}$

Mean: λ

Variance: λ

Probability mass function (p.m.f):

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Cumulative distribution function (c.d.f):

$$P(X \le k) = e^{-\lambda} \sum_{i=0}^{\lfloor k \rfloor} \frac{\lambda^i}{i!}$$

Moment generating function (m.g.f):

$$E(e^{tX}) = e^{\lambda(e^t - 1)}$$

Value

A `Poisson` object.

See Also

Other discrete distributions: `Bernoulli`, `Binomial`, `Categorical`, `Geometric`, `HyperGeometric`, `Multinomial`, `NegativeBinomial`

Examples

```
set.seed(27)

X <- Poisson(2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

quantile — *Find the quantile of a probability distribution*

Description

TODO: Note that this current masks the `stats::quantile()` generic to allow for consistent argument names and warnings when arguments disappear into `...`.

Usage

```
quantile(d, p, ...)
```

Arguments

d	A probability distribution object such as those created by a call to `Bernoulli()`, `Beta()`, or `Binomial()`.
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of `p`.

Examples

```
X <- Normal()

cdf(X, c(0.2, 0.4, 0.6, 0.8))
```

quantile.Bernoulli — *Determine quantiles of a Bernoulli distribution*

Description

`quantile()` is the inverse of `cdf()`.

Usage

```
## S3 method for class 'Bernoulli'
quantile(d, p, ...)
```

Arguments

d	A Bernoulli object created by a call to Bernoulli().
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of p.

Examples

```
set.seed(27)

X <- Bernoulli(0.7)
X

random(X, 10)
pdf(X, 1)
log_pdf(X, 1)
cdf(X, 0)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

---

quantile.Beta *Determine quantiles of a Beta distribution*

Description

quantile() is the inverse of cdf().

Usage

```
## S3 method for class 'Beta'
quantile(d, p, ...)
```

Arguments

d	A Beta object created by a call to Beta().
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of p.

Examples

```
set.seed(27)

X <- Beta(1, 2)
X

random(X, 10)

pdf(X, 0.7)
log_pdf(X, 0.7)

cdf(X, 0.7)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

---

quantile.Binomial   *Determine quantiles of a Binomial distribution*

Description

quantile() is the inverse of cdf().

Usage

```
## S3 method for class 'Binomial'
quantile(d, p, ...)
```

Arguments

d	A Binomial object created by a call to Binomial().
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of p.

Examples

```
set.seed(27)

X <- Binomial(10, 0.2)
X

random(X, 10)

pdf(X, 2L)
log_pdf(X, 2L)

cdf(X, 4L)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

quantile.Categorical *Determine quantiles of a Categorical discrete distribution*

Description

`quantile()` is the inverse of `cdf()`.

Usage

```
## S3 method for class 'Categorical'
quantile(d, p, ...)
```

Arguments

d	A `Categorical` object created by a call to `Categorical()`.
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of p.

Examples

```
set.seed(27)

X <- Categorical(1:3, p = c(0.4, 0.1, 0.5))
X
```

```
Y <- Categorical(LETTERS[1:4])
Y

random(X, 10)
random(Y, 10)

pdf(X, 1)
log_pdf(X, 1)

cdf(X, 1)
quantile(X, 0.5)
## Not run:
# cdfs are only defined for numeric sample spaces. this errors!
cdf(Y, "a")

# same for quantiles. this also errors!
quantile(Y, 0.7)

## End(Not run)
```

---

quantile.Cauchy *Determine quantiles of a Cauchy distribution*

Description

quantile() is the inverse of cdf().

Usage

```
## S3 method for class 'Cauchy'
quantile(d, p, ...)
```

Arguments

d	A Cauchy object created by a call to Cauchy().
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of p.

Examples

```
set.seed(27)

X <- Cauchy(10, 0.2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 2)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

quantile.ChiSquare   *Determine quantiles of a chi square distribution*

Description

`quantile()` is the inverse of `cdf()`.

Usage

```
## S3 method for class 'ChiSquare'
quantile(d, p, ...)
```

Arguments

d	A `ChiSquare` object created by a call to `ChiSquare()`.
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of `p`.

Examples

```
set.seed(27)

X <- ChiSquare(5)
X
```

```
random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`quantile.Exponential` *Determine quantiles of a Exponential distribution*

Description

`quantile()` is the inverse of `cdf()`.

Usage

```
## S3 method for class 'Exponential'
quantile(d, p, ...)
```

Arguments

`d`	A `Exponential` object created by a call to `Exponential()`.
`p`	A vector of probabilites.
`...`	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of `p`.

Examples

```
set.seed(27)

X <- Exponential(5)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
```

```
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

quantile.FisherF *Determine quantiles of an F distribution*

Description

quantile() is the inverse of cdf().

Usage

```
## S3 method for class 'FisherF'
quantile(d, p, ...)
```

Arguments

d	A FisherF object created by a call to FisherF().
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of p.

Examples

```
set.seed(27)

X <- FisherF(5, 10, 0.2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

quantile.Gamma *Determine quantiles of a Gamma distribution*

Description

`quantile()` is the inverse of `cdf()`.

Usage

```
## S3 method for class 'Gamma'
quantile(d, p, ...)
```

Arguments

d	A Gamma object created by a call to `Gamma()`.
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of p.

Examples

```
set.seed(27)

X <- Gamma(5, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

quantile.Geometric *Determine quantiles of a Geometric distribution*

Description

Determine quantiles of a Geometric distribution

Usage

```
## S3 method for class 'Geometric'
quantile(d, p, ...)
```

Arguments

d	A Geometric object created by a call to Geometric().
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of p.

See Also

Other Geometric distribution: cdf.Geometric, pdf.Geometric, random.Geometric

Examples

```
set.seed(27)

X <- Geometric(0.3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

`quantile.HyperGeometric` *Determine quantiles of a HyperGeometric distribution*

Description

Determine quantiles of a HyperGeometric distribution

Usage

```
## S3 method for class 'HyperGeometric'
quantile(d, p, ...)
```

Arguments

`d`	A `HyperGeometric` object created by a call to `HyperGeometric()`.
`p`	A vector of probabilites.
`...`	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of `p`.

See Also

Other HyperGeometric distribution: `cdf.HyperGeometric`, `pdf.HyperGeometric`, `random.HyperGeometric`

Examples

```
set.seed(27)

X <- HyperGeometric(4, 5, 8)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

quantile.Logistic *Determine quantiles of a Logistic distribution*

Description

Determine quantiles of a Logistic distribution

Usage

```
## S3 method for class 'Logistic'
quantile(d, p, ...)
```

Arguments

d	A `Logistic` object created by a call to `Logistic()`.
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of `p`.

See Also

Other Logistic distribution: `cdf.Logistic`, `pdf.Logistic`, `random.Logistic`

Examples

```
set.seed(27)

X <- Logistic(2, 4)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

quantile.LogNormal *Determine quantiles of a LogNormal distribution*

Description

Determine quantiles of a LogNormal distribution

Usage

```
## S3 method for class 'LogNormal'
quantile(d, p, ...)
```

Arguments

d	A LogNormal object created by a call to LogNormal().
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of p.

See Also

Other LogNormal distribution: cdf.LogNormal, fit_mle.LogNormal, pdf.LogNormal, random.LogNormal

Examples

```
set.seed(27)

X <- LogNormal(0.3, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

`quantile.NegativeBinomial` *Determine quantiles of a NegativeBinomial distribution*

Description

Determine quantiles of a NegativeBinomial distribution

Usage

```
## S3 method for class 'NegativeBinomial'
quantile(d, p, ...)
```

Arguments

d	A `NegativeBinomial` object created by a call to `NegativeBinomial()`.
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of `p`.

See Also

Other NegativeBinomial distribution: `cdf.NegativeBinomial`, `pdf.NegativeBinomial`, `random.NegativeBinomial`

Examples

```
set.seed(27)

X <- NegativeBinomial(10, 0.3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

quantile.Normal *Determine quantiles of a Normal distribution*

Description

Please see the documentation of `Normal()` for some properties of the Normal distribution, as well as extensive examples showing to how calculate p-values and confidence intervals. `quantile()`

Usage

```
## S3 method for class 'Normal'
quantile(d, p, ...)
```

Arguments

d	A `Normal` object created by a call to `Normal()`.
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Details

This function returns the same values that you get from a Z-table. Note `quantile()` is the inverse of `cdf()`. Please see the documentation of `Normal()` for some properties of the Normal distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Value

A vector of quantiles, one for each element of `p`.

See Also

Other Normal distribution: `cdf.Normal`, `fit_mle.Normal`, `pdf.Normal`

Examples

```
set.seed(27)

X <- Normal(5, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
```

```
quantile(X, 0.7)

### example: calculating p-values for two-sided Z-test

# here the null hypothesis is H_0: mu = 3
# and we assume sigma = 2

# exactly the same as: Z <- Normal(0, 1)
Z <- Normal()

# data to test
x <- c(3, 7, 11, 0, 7, 0, 4, 5, 6, 2)
nx <- length(x)

# calculate the z-statistic
z_stat <- (mean(x) - 3) / (2 / sqrt(nx))
z_stat

# calculate the two-sided p-value
1 - cdf(Z, abs(z_stat)) + cdf(Z, -abs(z_stat))

# exactly equivalent to the above
2 * cdf(Z, -abs(z_stat))

# p-value for one-sided test
# H_0: mu <= 3   vs   H_A: mu > 3
1 - cdf(Z, z_stat)

# p-value for one-sided test
# H_0: mu >= 3   vs   H_A: mu < 3
cdf(Z, z_stat)

### example: calculating a 88 percent Z CI for a mean

# same `x` as before, still assume `sigma = 2`

# lower-bound
mean(x) - quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# upper-bound
mean(x) + quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# equivalent to
mean(x) + c(-1, 1) * quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# also equivalent to
mean(x) + quantile(Z, 0.12 / 2) * 2 / sqrt(nx)
mean(x) + quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

### generating random samples and plugging in ks.test()

set.seed(27)
```

```
# generate a random sample
ns <- random(Normal(3, 7), 26)

# test if sample is Normal(3, 7)
ks.test(ns, pnorm, mean = 3, sd = 7)

# test if sample is gamma(8, 3) using base R pgamma()
ks.test(ns, pgamma, shape = 8, rate = 3)

### MISC

# note that the cdf() and quantile() functions are inverses
cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`quantile.Poisson` *Determine quantiles of a Poisson distribution*

Description

`quantile()` is the inverse of `cdf()`.

Usage

```
## S3 method for class 'Poisson'
quantile(d, p, ...)
```

Arguments

`d`	A `Poisson` object created by a call to `Poisson()`.
`p`	A vector of probabilites.
`...`	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of `p`.

Examples

```
set.seed(27)

X <- Poisson(2)
X

random(X, 10)
```

```
pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

quantile.StudentsT *Determine quantiles of a StudentsT distribution*

Description

Please see the documentation of `StudentsT()` for some properties of the StudentsT distribution, as well as extensive examples showing to how calculate p-values and confidence intervals. `quantile()`

Usage

```
## S3 method for class 'StudentsT'
quantile(d, p, ...)
```

Arguments

`d`	A `StudentsT` object created by a call to `StudentsT()`.
`p`	A vector of probabilites.
`...`	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Details

This function returns the same values that you get from a Z-table. Note `quantile()` is the inverse of `cdf()`. Please see the documentation of `StudentsT()` for some properties of the StudentsT distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Value

A vector of quantiles, one for each element of `p`.

See Also

Other StudentsT distribution: `cdf.StudentsT`, `pdf.StudentsT`, `random.StudentsT`

Examples

```
set.seed(27)

X <- StudentsT(3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

### example: calculating p-values for two-sided T-test

# here the null hypothesis is H_0: mu = 3

# data to test
x <- c(3, 7, 11, 0, 7, 0, 4, 5, 6, 2)
nx <- length(x)

# calculate the T-statistic
t_stat <- (mean(x) - 3) / (sd(x) / sqrt(nx))
t_stat

# null distribution of statistic depends on sample size!
T <- StudentsT(df = nx - 1)

# calculate the two-sided p-value
1 - cdf(T, abs(t_stat)) + cdf(T, -abs(t_stat))

# exactly equivalent to the above
2 * cdf(T, -abs(t_stat))

# p-value for one-sided test
# H_0: mu <= 3   vs   H_A: mu > 3
1 - cdf(T, t_stat)

# p-value for one-sided test
# H_0: mu >= 3   vs   H_A: mu < 3
cdf(T, t_stat)

### example: calculating a 88 percent T CI for a mean

# lower-bound
mean(x) - quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# upper-bound
mean(x) + quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)
```

```
# equivalent to
mean(x) + c(-1, 1) * quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# also equivalent to
mean(x) + quantile(T, 0.12 / 2) * sd(x) / sqrt(nx)
mean(x) + quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)
```

---

quantile.Tukey *Determine quantiles of a Tukey distribution*

Description

Determine quantiles of a Tukey distribution

Usage

```
## S3 method for class 'Tukey'
quantile(d, p, ...)
```

Arguments

d	A Tukey distribution created by a call to Tukey().
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of p.

See Also

Other Tukey distribution: cdf.Tukey

Examples

```
set.seed(27)

X <- Tukey(4L, 16L, 2L)
X

cdf(X, 4)
quantile(X, 0.7)
```

quantile.Uniform *Determine quantiles of a continuous Uniform distribution*

Description

`quantile()` is the inverse of `cdf()`.

Usage

```
## S3 method for class 'Uniform'
quantile(d, p, ...)
```

Arguments

d	A `Uniform` object created by a call to `Uniform()`.
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of p.

Examples

```
set.seed(27)

X <- Uniform(1, 2)
X

random(X, 10)

pdf(X, 0.7)
log_pdf(X, 0.7)

cdf(X, 0.7)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

quantile.Weibull *Determine quantiles of a Weibull distribution*

Description

Determine quantiles of a Weibull distribution

Usage

```
## S3 method for class 'Weibull'
quantile(d, p, ...)
```

Arguments

d	A Weibull object created by a call to Weibull().
p	A vector of probabilites.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector of quantiles, one for each element of p.

See Also

Other Weibull distribution: cdf.Weibull, pdf.Weibull, random.Weibull

Examples

```
set.seed(27)

X <- Weibull(0.3, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

`random` *Draw a random sample from a probability distribution*

Description

Draw a random sample from a probability distribution

Usage

```
random(d, n = 1L, ...)
```

Arguments

d	A probability distribution object such as those created by a call to `Bernoulli()`, `Beta()`, or `Binomial()`.
n	The number of samples to draw. Should be a positive integer. Defaults to `1L`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Examples

```
X <- Normal()

random(X, 10)
```

`random.Bernoulli` *Draw a random sample from a Bernoulli distribution*

Description

Draw a random sample from a Bernoulli distribution

Usage

```
## S3 method for class 'Bernoulli'
random(d, n = 1L, ...)
```

Arguments

d	A `Bernoulli` object created by a call to `Bernoulli()`.
n	The number of samples to draw. Defaults to `1L`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

An integer vector of zeros and ones of length n.

Examples

```
set.seed(27)

X <- Bernoulli(0.7)
X

random(X, 10)
pdf(X, 1)
log_pdf(X, 1)
cdf(X, 0)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

---

random.Beta *Draw a random sample from a Beta distribution*

Description

Draw a random sample from a Beta distribution

Usage

```
## S3 method for class 'Beta'
random(d, n = 1L, ...)
```

Arguments

d	A Beta object created by a call to Beta().
n	The number of samples to draw. Defaults to 1L.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A numeric vector containing values in [0,1] of length n.

Examples

```
set.seed(27)

X <- Beta(1, 2)
X

random(X, 10)

pdf(X, 0.7)
log_pdf(X, 0.7)

cdf(X, 0.7)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

---

random.Binomial   *Draw a random sample from a Binomial distribution*

Description

Draw a random sample from a Binomial distribution

Usage

```
## S3 method for class 'Binomial'
random(d, n = 1L, ...)
```

Arguments

d	A `Binomial` object created by a call to `Binomial()`.
n	The number of samples to draw. Defaults to `1L`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

An integer vector containing values between `0` and `d$size` of length `n`.

Examples

```
set.seed(27)

X <- Binomial(10, 0.2)
X
```

```
random(X, 10)

pdf(X, 2L)
log_pdf(X, 2L)

cdf(X, 4L)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`random.Categorical` *Draw a random sample from a Categorical distribution*

Description

Draw a random sample from a Categorical distribution

Usage

```
## S3 method for class 'Categorical'
random(d, n = 1L, ...)
```

Arguments

d	A `Categorical` object created by a call to `Categorical()`.
n	The number of samples to draw. Defaults to `1L`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A vector containing values from `outcomes` of length `n`.

Examples

```
set.seed(27)

X <- Categorical(1:3, p = c(0.4, 0.1, 0.5))
X

Y <- Categorical(LETTERS[1:4])
Y

random(X, 10)
random(Y, 10)
```

```
pdf(X, 1)
log_pdf(X, 1)

cdf(X, 1)
quantile(X, 0.5)
## Not run:
# cdfs are only defined for numeric sample spaces. this errors!
cdf(Y, "a")

# same for quantiles. this also errors!
quantile(Y, 0.7)

## End(Not run)
```

---

random.Cauchy *Draw a random sample from a Cauchy distribution*

Description

Draw a random sample from a Cauchy distribution

Usage

```
## S3 method for class 'Cauchy'
random(d, n = 1L, ...)
```

Arguments

d	A Cauchy object created by a call to Cauchy().
n	The number of samples to draw. Defaults to 1L.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A numeric vector of length n.

Examples

```
set.seed(27)

X <- Cauchy(10, 0.2)
X

random(X, 10)

pdf(X, 2)
```

```
log_pdf(X, 2)

cdf(X, 2)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

random.ChiSquare *Draw a random sample from a chi square distribution*

Description

Draw a random sample from a chi square distribution

Usage

```
## S3 method for class 'ChiSquare'
random(d, n = 1L, ...)
```

Arguments

d	A ChiSquare object created by a call to ChiSquare().
n	The number of samples to draw. Defaults to 1L.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A numeric vector of length n.

Examples

```
set.seed(27)

X <- ChiSquare(5)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

`random.Exponential` *Draw a random sample from a Exponential distribution*

Description

Draw a random sample from a Exponential distribution

Usage

```
## S3 method for class 'Exponential'
random(d, n = 1L, ...)
```

Arguments

d	A `Exponential` object created by a call to `Exponential()`.
n	The number of samples to draw. Defaults to `1L`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A numeric vector of length `n`.

Examples

```
set.seed(27)

X <- Exponential(5)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

random.FisherF *Draw a random sample from an F distribution*

Description

Draw a random sample from an F distribution

Usage

```
## S3 method for class 'FisherF'
random(d, n = 1L, ...)
```

Arguments

d	A FisherF object created by a call to FisherF().
n	The number of samples to draw. Defaults to 1L.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A numeric vector of length n.

Examples

```
set.seed(27)

X <- FisherF(5, 10, 0.2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

random.Gamma *Draw a random sample from a Gamma distribution*

Description

Draw a random sample from a Gamma distribution

Usage

```
## S3 method for class 'Gamma'
random(d, n = 1L, ...)
```

Arguments

d	A Gamma object created by a call to Gamma().
n	The number of samples to draw. Defaults to 1L.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A numeric vector of length n.

Examples

```
set.seed(27)

X <- Gamma(5, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

random.Geometric — *Draw a random sample from a Geometric distribution*

Description

Please see the documentation of Geometric() for some properties of the Geometric distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Usage

```
## S3 method for class 'Geometric'
random(d, n = 1L, ...)
```

Arguments

d	A Geometric object created by a call to Geometric().
n	The number of samples to draw. Defaults to 1L.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

An integer vector of length n.

See Also

Other Geometric distribution: cdf.Geometric, pdf.Geometric, quantile.Geometric

Examples

```
set.seed(27)

X <- Geometric(0.3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

random.HyperGeometric *Draw a random sample from a HyperGeometric distribution*

Description

Please see the documentation of HyperGeometric() for some properties of the HyperGeometric distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Usage

```
## S3 method for class 'HyperGeometric'
random(d, n = 1L, ...)
```

Arguments

d	A HyperGeometric object created by a call to HyperGeometric().
n	The number of samples to draw. Defaults to 1L.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

An integer vector of length n.

See Also

Other HyperGeometric distribution: cdf.HyperGeometric, pdf.HyperGeometric, quantile.HyperGeometric

Examples

```
set.seed(27)

X <- HyperGeometric(4, 5, 8)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

random.Logistic *Draw a random sample from a Logistic distribution*

Description

Draw a random sample from a Logistic distribution

Usage

```
## S3 method for class 'Logistic'
random(d, n = 1L, ...)
```

Arguments

d	A `Logistic` object created by a call to `Logistic()`.
n	The number of samples to draw. Defaults to `1L`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

An integer vector of length `n`.

See Also

Other Logistic distribution: `cdf.Logistic`, `pdf.Logistic`, `quantile.Logistic`

Examples

```
set.seed(27)

X <- Logistic(2, 4)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

random.LogNormal — *Draw a random sample from a LogNormal distribution*

Description

Draw a random sample from a LogNormal distribution

Usage

```
## S3 method for class 'LogNormal'
random(d, n = 1L, ...)
```

Arguments

d	A LogNormal object created by a call to LogNormal().
n	The number of samples to draw. Defaults to 1L.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

An integer vector of length n.

See Also

Other LogNormal distribution: cdf.LogNormal, fit_mle.LogNormal, pdf.LogNormal, quantile.LogNormal

Examples

```
set.seed(27)

X <- LogNormal(0.3, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

`random.Multinomial` *Draw a random sample from a Multinomial distribution*

Description

Draw a random sample from a Multinomial distribution

Usage

```
## S3 method for class 'Multinomial'
random(d, n = 1L, ...)
```

Arguments

d	A `Multinomial` object created by a call to `Multinomial()`.
n	The number of samples to draw. Defaults to `1L`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

An integer vector of length `n`.

See Also

Other Multinomial distribution: `pdf.Multinomial`

Examples

```
set.seed(27)

X <- Multinomial(size = 5, p = c(0.3, 0.4, 0.2, 0.1))
X

random(X, 10)

# pdf(X, 2)
# log_pdf(X, 2)
```

random.NegativeBinomial

Draw a random sample from a negative binomial distribution

Description

Draw a random sample from a negative binomial distribution

Usage

```
## S3 method for class 'NegativeBinomial'
random(d, n = 1L, ...)
```

Arguments

d	A NegativeBinomial object created by a call to NegativeBinomial().
n	The number of samples to draw. Defaults to 1L.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

An integer vector of length n.

See Also

Other NegativeBinomial distribution: cdf.NegativeBinomial, pdf.NegativeBinomial, quantile.NegativeBinomial

Examples

```
set.seed(27)

X <- NegativeBinomial(10, 0.3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

random.Normal *Draw a random sample from a Normal distribution*

Description

Please see the documentation of Normal() for some properties of the Normal distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Usage

```
## S3 method for class 'Normal'
random(d, n = 1L, ...)
```

Arguments

d	A Normal object created by a call to Normal().
n	The number of samples to draw. Defaults to 1L.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A numeric vector of length n.

Examples

```
set.seed(27)

X <- Normal(5, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

### example: calculating p-values for two-sided Z-test

# here the null hypothesis is H_0: mu = 3
# and we assume sigma = 2

# exactly the same as: Z <- Normal(0, 1)
Z <- Normal()

# data to test
```

```
x <- c(3, 7, 11, 0, 7, 0, 4, 5, 6, 2)
nx <- length(x)

# calculate the z-statistic
z_stat <- (mean(x) - 3) / (2 / sqrt(nx))
z_stat

# calculate the two-sided p-value
1 - cdf(Z, abs(z_stat)) + cdf(Z, -abs(z_stat))

# exactly equivalent to the above
2 * cdf(Z, -abs(z_stat))

# p-value for one-sided test
# H_0: mu <= 3   vs   H_A: mu > 3
1 - cdf(Z, z_stat)

# p-value for one-sided test
# H_0: mu >= 3   vs   H_A: mu < 3
cdf(Z, z_stat)

### example: calculating a 88 percent Z CI for a mean

# same `x` as before, still assume `sigma = 2`

# lower-bound
mean(x) - quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# upper-bound
mean(x) + quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# equivalent to
mean(x) + c(-1, 1) * quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

# also equivalent to
mean(x) + quantile(Z, 0.12 / 2) * 2 / sqrt(nx)
mean(x) + quantile(Z, 1 - 0.12 / 2) * 2 / sqrt(nx)

### generating random samples and plugging in ks.test()

set.seed(27)

# generate a random sample
ns <- random(Normal(3, 7), 26)

# test if sample is Normal(3, 7)
ks.test(ns, pnorm, mean = 3, sd = 7)

# test if sample is gamma(8, 3) using base R pgamma()
ks.test(ns, pgamma, shape = 8, rate = 3)

### MISC
```

```
# note that the cdf() and quantile() functions are inverses
cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

---

random.Poisson *Draw a random sample from a Poisson distribution*

Description

Draw a random sample from a Poisson distribution

Usage

```
## S3 method for class 'Poisson'
random(d, n = 1L, ...)
```

Arguments

d	A `Poisson` object created by a call to `Poisson()`.
n	The number of samples to draw. Defaults to `1L`.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A numeric vector of length `n`.

Examples

```
set.seed(27)

X <- Poisson(2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 7))
```

random.StudentsT *Draw a random sample from a StudentsT distribution*

Description

Please see the documentation of StudentsT() for some properties of the T distribution, as well as extensive examples showing to how calculate p-values and confidence intervals.

Usage

```
## S3 method for class 'StudentsT'
random(d, n = 1L, ...)
```

Arguments

d	A StudentsT object created by a call to StudentsT().
n	The number of samples to draw. Defaults to 1L.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A numeric vector of length n.

See Also

Other StudentsT distribution: cdf.StudentsT, pdf.StudentsT, quantile.StudentsT

Examples

```
set.seed(27)

X <- StudentsT(3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

### example: calculating p-values for two-sided T-test

# here the null hypothesis is H_0: mu = 3
```

```
# data to test
x <- c(3, 7, 11, 0, 7, 0, 4, 5, 6, 2)
nx <- length(x)

# calculate the T-statistic
t_stat <- (mean(x) - 3) / (sd(x) / sqrt(nx))
t_stat

# null distribution of statistic depends on sample size!
T <- StudentsT(df = nx - 1)

# calculate the two-sided p-value
1 - cdf(T, abs(t_stat)) + cdf(T, -abs(t_stat))

# exactly equivalent to the above
2 * cdf(T, -abs(t_stat))

# p-value for one-sided test
# H_0: mu <= 3   vs   H_A: mu > 3
1 - cdf(T, t_stat)

# p-value for one-sided test
# H_0: mu >= 3   vs   H_A: mu < 3
cdf(T, t_stat)

### example: calculating a 88 percent T CI for a mean

# lower-bound
mean(x) - quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# upper-bound
mean(x) + quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# equivalent to
mean(x) + c(-1, 1) * quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# also equivalent to
mean(x) + quantile(T, 0.12 / 2) * sd(x) / sqrt(nx)
mean(x) + quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)
```

`random.Uniform` *Draw a random sample from a continuous Uniform distribution*

Description

Draw a random sample from a continuous Uniform distribution

Usage

```
## S3 method for class 'Uniform'
random(d, n = 1L, ...)
```

Arguments

- d — A Uniform object created by a call to Uniform().
- n — The number of samples to draw. Defaults to 1L.
- ... — Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

A numeric vector containing values in [a,b] of length n.

Examples

```
set.seed(27)

X <- Uniform(1, 2)
X

random(X, 10)

pdf(X, 0.7)
log_pdf(X, 0.7)

cdf(X, 0.7)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

---

random.Weibull — *Draw a random sample from a Weibull distribution*

Description

Draw a random sample from a Weibull distribution

Usage

```
## S3 method for class 'Weibull'
random(d, n = 1L, ...)
```

Arguments

- d — A Weibull object created by a call to Weibull().
- n — The number of samples to draw. Defaults to 1L.
- ... — Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

An integer vector of length n.

See Also

Other Weibull distribution: cdf.Weibull, pdf.Weibull, quantile.Weibull

Examples

```
set.seed(27)

X <- Weibull(0.3, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

StudentsT *Create a Student's T distribution*

Description

The Student's T distribution is closely related to the Normal() distribution, but has heavier tails. As ν increases to ∞, the Student's T converges to a Normal. The T distribution appears repeatedly throughout classic frequentist hypothesis testing when comparing group means.

Usage

```
StudentsT(df)
```

Arguments

df	Degrees of freedom. Can be any positive number. Often called ν in textbooks.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail and much greater clarity.

In the following, let X be a Students T random variable with df = ν.

Support: R, the set of all real numbers

Mean: Undefined unless $\nu \geq 2$, in which case the mean is zero.

Variance:

$$\frac{\nu}{\nu - 2}$$

Undefined if $\nu < 1$, infinite when $1 < \nu \leq 2$.

Probability density function (p.d.f):

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

Cumulative distribution function (c.d.f):

Nasty, omitted.

Moment generating function (m.g.f):

Undefined.

Value

A `StudentsT` object.

See Also

Other continuous distributions: Beta, Cauchy, ChiSquare, Exponential, FisherF, Gamma, LogNormal, Logistic, Normal, Tukey, Uniform, Weibull

Examples

```
set.seed(27)

X <- StudentsT(3)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)

### example: calculating p-values for two-sided T-test

# here the null hypothesis is H_0: mu = 3

# data to test
x <- c(3, 7, 11, 0, 7, 0, 4, 5, 6, 2)
nx <- length(x)

# calculate the T-statistic
```

```
t_stat <- (mean(x) - 3) / (sd(x) / sqrt(nx))
t_stat

# null distribution of statistic depends on sample size!
T <- StudentsT(df = nx - 1)

# calculate the two-sided p-value
1 - cdf(T, abs(t_stat)) + cdf(T, -abs(t_stat))

# exactly equivalent to the above
2 * cdf(T, -abs(t_stat))

# p-value for one-sided test
# H_0: mu <= 3   vs   H_A: mu > 3
1 - cdf(T, t_stat)

# p-value for one-sided test
# H_0: mu >= 3   vs   H_A: mu < 3
cdf(T, t_stat)

### example: calculating a 88 percent T CI for a mean

# lower-bound
mean(x) - quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# upper-bound
mean(x) + quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# equivalent to
mean(x) + c(-1, 1) * quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)

# also equivalent to
mean(x) + quantile(T, 0.12 / 2) * sd(x) / sqrt(nx)
mean(x) + quantile(T, 1 - 0.12 / 2) * sd(x) / sqrt(nx)
```

`suff_stat` *Compute the sufficient statistics of a distribution from data*

Description

Compute the sufficient statistics of a distribution from data

Usage

```
suff_stat(d, x, ...)
```

Arguments

d	A probability distribution object such as those created by a call to `Bernoulli()`, `Beta()`, or `Binomial()`.

x	A vector of data to compute the likelihood.
...	Unused. Unevaluated arguments will generate a warning to catch mispellings or other possible errors.

Value

a named list of sufficient statistics

---

suff_stat.Bernoulli — *Compute the sufficient statistics for a Bernoulli distribution from data*

Description

Compute the sufficient statistics for a Bernoulli distribution from data

Usage

```
## S3 method for class 'Bernoulli'
suff_stat(d, x, ...)
```

Arguments

d	A `Bernoulli` object.
x	A vector of zeroes and ones.
...	Unused.

Value

A named list of the sufficient statistics of the Bernoulli distribution:

- `successes`: The number of successful trials (`sum(x == 1)`)
- `failures`: The number of failed trials (`sum(x == 0)`).

---

suff_stat.Binomial — *Compute the sufficient statistics for the Binomial distribution from data*

Description

Compute the sufficient statistics for the Binomial distribution from data

Usage

```
## S3 method for class 'Binomial'
suff_stat(d, x, ...)
```

Arguments

d	A Binomial object.
x	A vector of zeroes and ones.
...	Unused.

Value

A named list of the sufficient statistics of the Binomial distribution:

- successes: The total number of successful trials.
- experiments: The number of experiments run.
- trials: The number of trials run per experiment.

---

suff_stat.Exponential *Compute the sufficient statistics of an Exponential distribution from data*

Description

Compute the sufficient statistics of an Exponential distribution from data

Usage

```
## S3 method for class 'Exponential'
suff_stat(d, x, ...)
```

Arguments

d	An Exponential object created by a call to Exponential().
x	A vector of data.
...	Unused.

Value

A named list of the sufficient statistics of the exponential distribution:

- sum: The sum of the observations.
- samples: The number of observations.

suff_stat.Gamma — *Compute the sufficient statistics for a bernoulli distribution from data*

Description

- `sum`: The sum of the data.
  - `log_sum`: The log of the sum of the data.
  - `samples`: The number of samples in the data.

Usage

```
## S3 method for class 'Gamma'
suff_stat(d, x, ...)
```

Arguments

d	A `Gamma` object created by a call to `Gamma()`.
x	A vector to fit the Gamma distribution to.
...	Unused.

Value

a `Gamma` object

suff_stat.Geometric — *Compute the sufficient statistics for the Geometric distribution from data*

Description

Compute the sufficient statistics for the Geometric distribution from data

Usage

```
## S3 method for class 'Geometric'
suff_stat(d, x, ...)
```

Arguments

d	A `Geometric` object.
x	A vector of zeroes and ones.
...	Unused.

Value

A named list of the sufficient statistics of the Geometric distribution:

- `trials`: The total number of trials ran until the first success.
- `experiments`: The number of experiments run.

---

suff_stat.LogNormal   *Compute the sufficient statistics for a Log-normal distribution from data*

Description

Compute the sufficient statistics for a Log-normal distribution from data

Usage

```
## S3 method for class 'LogNormal'
suff_stat(d, x, ...)
```

Arguments

d	A `LogNormal` object created by a call to `LogNormal()`.
x	A vector of data.
...	Unused.

Value

A named list of the sufficient statistics of the normal distribution:

- `mu`: The sample mean of the log of the data.
- `sigma`: The sample standard deviation of the log of the data.
- `samples`: The number of samples in the data.

`suff_stat.Normal` *Compute the sufficient statistics for a Normal distribution from data*

Description

Compute the sufficient statistics for a Normal distribution from data

Usage

```
## S3 method for class 'Normal'
suff_stat(d, x, ...)
```

Arguments

`d`	A `Normal` object created by a call to `Normal()`.
`x`	A vector of data.
`...`	Unused.

Value

A named list of the sufficient statistics of the normal distribution:

- `mu`: The sample mean of the data.
- `sigma`: The sample standard deviation of the data.
- `samples`: The number of samples in the data.

`suff_stat.Poisson` *Compute the sufficient statistics of an Poisson distribution from data*

Description

Compute the sufficient statistics of an Poisson distribution from data

Usage

```
## S3 method for class 'Poisson'
suff_stat(d, x, ...)
```

Arguments

`d`	An `Poisson` object created by a call to `Poisson()`.
`x`	A vector of data.
`...`	Unused.

Value

A named list of the sufficient statistics of the Poisson distribution:

- `sum`: The sum of the data.
- `samples`: The number of samples in the data.

Tukey   *Create a Tukey distribution*

Description

Tukey's studentized range distribution, used for Tukey's honestly significant differences test in ANOVA.

Usage

```
Tukey(nmeans, df, nranges)
```

Arguments

`nmeans`	Sample size for each range.
`df`	Degrees of freedom.
`nranges`	Number of groups being compared.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail and much greater clarity.

Support: R^+, the set of positive real numbers.

Other properties of Tukey's Studentized Range Distribution are omitted, largely because the distribution is not fun to work with.

Value

A `Tukey` object.

See Also

Other continuous distributions: `Beta`, `Cauchy`, `ChiSquare`, `Exponential`, `FisherF`, `Gamma`, `LogNormal`, `Logistic`, `Normal`, `StudentsT`, `Uniform`, `Weibull`

Examples

```
set.seed(27)

X <- Tukey(4L, 16L, 2L)
X

cdf(X, 4)
quantile(X, 0.7)
```

Uniform *Create a Continuous Uniform distribution*

Description

A distribution with constant density on an interval. The continuous analogue to the `Categorical()` distribution.

Usage

```
Uniform(a = 0, b = 1)
```

Arguments

a	The a parameter. `a` can be any value in the set of real numbers. Defaults to `0`.
b	The a parameter. `b` can be any value in the set of real numbers. It should be strictly bigger than `a`, but if is not, the order of the parameters is inverted. Defaults to `1`.

Value

A `Uniform` object.

See Also

Other continuous distributions: `Beta`, `Cauchy`, `ChiSquare`, `Exponential`, `FisherF`, `Gamma`, `LogNormal`, `Logistic`, `Normal`, `StudentsT`, `Tukey`, `Weibull`

Examples

```
set.seed(27)

X <- Uniform(1, 2)
X

random(X, 10)
```

```
pdf(X, 0.7)
log_pdf(X, 0.7)

cdf(X, 0.7)
quantile(X, 0.7)

cdf(X, quantile(X, 0.7))
quantile(X, cdf(X, 0.7))
```

Weibull *Create a Weibull distribution*

Description

Generalization of the gamma distribution. Often used in survival and time-to-event analyses.

Usage

```
Weibull(shape, scale)
```

Arguments

shape	The shape parameter k. Can be any positive real number.
scale	The scale parameter λ. Can be any positive real number.

Details

We recommend reading this documentation on https://alexpghayes.github.io/distributions3, where the math will render with additional detail and much greater clarity.

In the following, let X be a Weibull random variable with success probability `p` = p.

Support: R^+ and zero.

Mean: $\lambda\Gamma(1 + 1/k)$, where Γ is the gamma function.

Variance: $\lambda[\Gamma(1 + \frac{2}{k}) - (\Gamma(1 + \frac{1}{k}))^2]$

Probability density function (p.d.f):

$$f(x) = \frac{k}{\lambda}(\frac{x}{\lambda})^{k-1}e^{-(x/\lambda)^k}, x \geq 0$$

Cumulative distribution function (c.d.f):

$$F(x) = 1 - e^{-(x/\lambda)^k}, x \geq 0$$

Moment generating function (m.g.f):

$$\sum_{n=0}^{\infty} \frac{t^n \lambda^n}{n!}\Gamma(1 + n/k), k \geq 1$$

Value

A `Weibull` object.

See Also

Other continuous distributions: `Beta`, `Cauchy`, `ChiSquare`, `Exponential`, `FisherF`, `Gamma`, `LogNormal`, `Logistic`, `Normal`, `StudentsT`, `Tukey`, `Uniform`

Examples

```
set.seed(27)

X <- Weibull(0.3, 2)
X

random(X, 10)

pdf(X, 2)
log_pdf(X, 2)

cdf(X, 4)
quantile(X, 0.7)
```

Index