Package ‘distributional’

September 2, 2022

Title Vectorised Probability Distributions

Version 0.3.1

Description Vectorised distribution objects with tools for manipulating, visualising, and using probability distributions. Designed to allow model prediction outputs to return distributions rather than their parameters, allowing users to directly interact with predictive distributions in a data-oriented workflow. In addition to providing generic replacements for p/d/q/r functions, other useful statistics can be computed including means, variances, intervals, and highest density regions.

License GPL-3

Imports vctrs (>= 0.3.0), rlang (>= 0.4.5), generics, stats, numDeriv, ggplot2, scales, farver, digest, utils, lifecycle

Suggests testthat (>= 2.1.0), covr, mvtnorm, actuar (>= 2.0.0), ggdist

RdMacros lifecycle

URL https://pkg.mitchelloharawild.com/distributional/, https://github.com/mitchelloharawild/distributional

BugReports https://github.com/mitchelloharawild/distributional/issues

Encoding UTF-8

Language en-GB

RoxygenNote 7.2.1

NeedsCompilation no

Author Mitchell O'Hara-Wild [aut, cre] (<https://orcid.org/0000-0001-6729-7695>), Matthew Kay [aut] (<https://orcid.org/0000-0001-9446-0419>), Alex Hayes [aut] (<https://orcid.org/0000-0002-4985-5160>), Earo Wang [ctb] (<https://orcid.org/0000-0001-6448-5260>)

Maintainer Mitchell O'Hara-Wild <mail@mitchelloharawild.com>

Repository CRAN

Date/Publication 2022-09-02 07:10:02 UTC

R topics documented:

`cdf` *The cumulative distribution function*

Description

[Stable]

Usage

```
cdf(x, q, ..., log = FALSE)

## S3 method for class 'distribution'
cdf(x, q, ...)
```

Arguments

`x`	The distribution(s).
`q`	The quantile at which the cdf is calculated.
`...`	Additional arguments passed to methods.
`log`	If `TRUE`, probabilities will be given as log probabilities.

`covariance` *Covariance*

Description

[Stable]

A generic function for computing the covariance of an object.

Usage

```
covariance(x, ...)
```

Arguments

x	An object.
...	Additional arguments used by methods.

See Also

covariance.distribution(), variance()

`covariance.distribution` *Covariance of a probability distribution*

Description

[Stable]

Returns the empirical covariance of the probability distribution. If the method does not exist, the covariance of a random sample will be returned.

Usage

```
## S3 method for class 'distribution'
covariance(x, ...)
```

Arguments

x	The distribution(s).
...	Additional arguments used by methods.

density.distribution — *The probability density/mass function*

Description

[Stable]

Computes the probability density function for a continuous distribution, or the probability mass function for a discrete distribution.

Usage

```
## S3 method for class 'distribution'
density(x, at, ..., log = FALSE)
```

Arguments

`x`	The distribution(s).
`at`	The point at which to compute the density/mass.
`...`	Additional arguments passed to methods.
`log`	If `TRUE`, probabilities will be given as log probabilities.

dist_bernoulli — *The Bernoulli distribution*

Description

[Stable]

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

```
dist_bernoulli(prob)
```

Arguments

`prob`	The probability of success on each trial, `prob` can be any value in `[0, 1]`.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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):

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

Examples

```
dist <- dist_bernoulli(prob = c(0.05, 0.5, 0.3, 0.9, 0.1))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

dist_beta *The Beta distribution*

Description

[Stable]

Usage

```
dist_beta(shape1, shape2)
```

Arguments

shape1, shape2	The non-negative shape parameters of the Beta distribution.

See Also

stats::Beta

Examples

```
dist <- dist_beta(shape1 = c(0.5, 5, 1, 2, 2), shape2 = c(0.5, 1, 3, 2, 5))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

dist_binomial *The Binomial distribution*

Description

[Stable]

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

```
dist_binomial(size, prob)
```

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.
`prob`	The probability of success on each trial, `prob` can be any value in `[0, 1]`.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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.

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):

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

Examples

```
dist <- dist_binomial(size = 1:5, prob = c(0.05, 0.5, 0.3, 0.9, 0.1))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_burr *The Burr distribution*

Description

[Stable]

Usage

```
dist_burr(shape1, shape2, rate = 1, scale = 1/rate)
```

Arguments

shape1, shape2, scale	parameters. Must be strictly positive.
rate	an alternative way to specify the scale.

See Also

actuar::Burr

Examples

```
dist <- dist_burr(shape1 = c(1,1,1,2,3,0.5), shape2 = c(1,2,3,1,1,2))
dist


mean(dist)
variance(dist)
support(dist)
```

```
generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

`dist_categorical` *The Categorical distribution*

Description

[Stable]

Categorical distributions are used to represent events with multiple outcomes, such as what number appears on the roll of a dice. This is also referred to as the 'generalised Bernoulli' or 'multinoulli' distribution. The Cateogorical distribution is a special case of the `Multinomial()` distribution with `n = 1`.

Usage

```
dist_categorical(prob, outcomes = NULL)
```

Arguments

`prob`	A list of probabilities of observing each outcome category.
`outcomes`	The values used to represent each outcome.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

In the following, let X be a Categorical random variable with probability parameters `p` = $\{p_1, p_2, \ldots, p_k\}$.

The Categorical probability distribution is widely used to model the occurance of multiple events. A simple example is the roll of a dice, where $p = \{1/6, 1/6, 1/6, 1/6, 1/6, 1/6\}$ giving equal chance of observing each number on a 6 sided dice.

Support: $\{1, \ldots, k\}$

Mean: p

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

Probability mass function (p.m.f):

$$P(X = i) = p_i$$

Cumulative distribution function (c.d.f):

The cdf() of a categorical distribution is undefined as the outcome categories aren't ordered.

Examples

```
dist <- dist_categorical(prob = list(c(0.05, 0.5, 0.15, 0.2, 0.1), c(0.3, 0.1, 0.6)))

dist

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

# The outcomes aren't ordered, so many statistics are not applicable.
cdf(dist, 4)
quantile(dist, 0.7)
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

dist <- dist_categorical(
  prob = list(c(0.05, 0.5, 0.15, 0.2, 0.1), c(0.3, 0.1, 0.6)),
  outcomes = list(letters[1:5], letters[24:26])
)

generate(dist, 10)

density(dist, "a")
density(dist, "z", log = TRUE)
```

`dist_cauchy` *The Cauchy distribution*

Description

[Stable]

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

```
dist_cauchy(location, scale)
```

Arguments

`location, scale`
location and scale parameters.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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.

See Also

stats::Cauchy

Examples

```
dist <- dist_cauchy(location = c(0, 0, 0, -2), scale = c(0.5, 1, 2, 1))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

`dist_chisq` *The (non-central) Chi-Squared Distribution*

Description

[Stable]

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

Usage

```
dist_chisq(df, ncp = 0)
```

Arguments

`df`	degrees of freedom (non-negative, but can be non-integer).
`ncp`	non-centrality parameter (non-negative).

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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):

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

See Also

stats::Chisquare

Examples

```
dist <- dist_chisq(df = c(1,2,3,4,6,9))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_degenerate *The degenerate distribution*

Description

[Stable]

The degenerate distribution takes a single value which is certain to be observed. It takes a single parameter, which is the value that is observed by the distribution.

Usage

```
dist_degenerate(x)
```

Arguments

x	The value of the distribution.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

In the following, let X be a degenerate random variable with value x = k_0.

Support: R, the set of all real numbers

Mean: k_0

Variance: 0

Probability density function (p.d.f):

$$f(x) = 1 for x = k_0$$

$$f(x) = 0 for x \neq k_0$$

Cumulative distribution function (c.d.f):

The cumulative distribution function has the form

$$F(x) = 0 for x < k_0$$

$$F(x) = 1 for x \geq k_0$$

Moment generating function (m.g.f):

$$E(e^{tX}) = e^{k_0 t}$$

Examples

```
dist_degenerate(x = 1:5)
```

`dist_exponential` *The Exponential Distribution*

Description

[Stable]

Usage

```
dist_exponential(rate)
```

Arguments

`rate`	vector of rates.

See Also

stats::Exponential

Examples

```
dist <- dist_exponential(rate = c(2, 1, 2/3))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

`dist_f` *The F Distribution*

Description

[Stable]

Usage

```
dist_f(df1, df2, ncp = NULL)
```

Arguments

`df1, df2`	degrees of freedom. `Inf` is allowed.
`ncp`	non-centrality parameter. If omitted the central F is assumed.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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):

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

See Also

stats::FDist

Examples

```
dist <- dist_f(df1 = c(1,2,5,10,100), df2 = c(1,1,2,1,100))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_gamma *The Gamma distribution*

Description

[Stable]

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

```
dist_gamma(shape, rate, scale = 1/rate)
```

Arguments

shape, scale	shape and scale parameters. Must be positive, scale strictly.
rate	an alternative way to specify the scale.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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):

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

See Also

stats::GammaDist

Examples

```
dist <- dist_gamma(shape = c(1,2,3,5,9,7.5,0.5), rate = c(0.5,0.5,0.5,1,2,1,1))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
```

```
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

`dist_geometric` *The Geometric Distribution*

Description

[Stable]

The Geometric distribution can be thought of as a generalization of the `dist_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

```
dist_geometric(prob)
```

Arguments

`prob`	probability of success in each trial. `0` < `prob` <= `1`.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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):

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

See Also

stats::Geometric

Examples

```
dist <- dist_geometric(prob = c(0.2, 0.5, 0.8))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_gumbel *The Gumbel distribution*

Description

[Stable]

The Gumbel distribution is a special case of the Generalized Extreme Value distribution, obtained when the GEV shape parameter ξ is equal to 0. It may be referred to as a type I extreme value distribution.

Usage

```
dist_gumbel(alpha, scale)
```

Arguments

alpha	location parameter.
scale	parameter. Must be strictly positive.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

In the following, let X be a Gumbel random variable with location parameter `mu` = μ, scale parameter `sigma` = σ.

Support: R, the set of all real numbers.

Mean: $\mu + \sigma\gamma$, where γ is Euler's constant, approximately equal to 0.57722.

Median: $\mu - \sigma \ln(\ln 2)$.

Variance: $\sigma^2\pi^2/6$.

Probability density function (p.d.f):

$$f(x) = \sigma^{-1} \exp[-(x - \mu)/\sigma] \exp\{-\exp[-(x - \mu)/\sigma]\}$$

for x in R, the set of all real numbers.

Cumulative distribution function (c.d.f):

In the $\xi = 0$ (Gumbel) special case

$$F(x) = \exp\{-\exp[-(x - \mu)/\sigma]\}$$

for x in R, the set of all real numbers.

See Also

actuar::Gumbel

Examples

```
dist <- dist_gumbel(alpha = c(0.5, 1, 1.5, 3), scale = c(2, 2, 3, 4))
dist


mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)
support(dist)
generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

dist_hypergeometric *The Hypergeometric distribution*

Description

[Stable]

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.

Usage

```
dist_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://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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

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

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):

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

Cumulative distribution function (c.d.f):

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

See Also

stats::Hypergeometric

Examples

```
dist <- dist_hypergeometric(m = rep(500, 3), n = c(50, 60, 70), k = c(100, 200, 300))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_inflated — *Inflate a value of a probability distribution*

Description

[Stable]

Usage

```
dist_inflated(dist, prob, x = 0)
```

Arguments

dist	The distribution(s) to inflate.
prob	The added probability of observing x.
x	The value to inflate. The default of x = 0 is for zero-inflation.

---

dist_inverse_exponential — *The Inverse Exponential distribution*

Description

[Stable]

Usage

```
dist_inverse_exponential(rate)
```

Arguments

`rate`	an alternative way to specify the scale.

See Also

actuar::InverseExponential

Examples

```
dist <- dist_inverse_exponential(rate = 1:5)
dist


mean(dist)
variance(dist)
support(dist)
generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

`dist_inverse_gamma` *The Inverse Gamma distribution*

Description

[Stable]

Usage

```
dist_inverse_gamma(shape, rate = 1/scale, scale)
```

Arguments

`shape, scale`	parameters. Must be strictly positive.
`rate`	an alternative way to specify the scale.

See Also

actuar::InverseGamma

Examples

```
dist <- dist_inverse_gamma(shape = c(1,2,3,3), rate = c(1,1,1,2))
dist


mean(dist)
variance(dist)
support(dist)
generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_inverse_gaussian *The Inverse Gaussian distribution*

Description

[Stable]

Usage

```
dist_inverse_gaussian(mean, shape)
```

Arguments

mean, shape	parameters. Must be strictly positive. Infinite values are supported.

See Also

actuar::InverseGaussian

Examples

```
dist <- dist_inverse_gaussian(mean = c(1,1,1,3,3), shape = c(0.2, 1, 3, 0.2, 1))
dist


mean(dist)
variance(dist)
support(dist)
generate(dist, 10)

density(dist, 2)
```

```
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_logarithmic *The Logarithmic distribution*

Description

[Stable]

Usage

```
dist_logarithmic(prob)
```

Arguments

prob	parameter. 0 <= prob < 1.

See Also

actuar::Logarithmic

Examples

```
dist <- dist_logarithmic(prob = c(0.33, 0.66, 0.99))
dist


mean(dist)
variance(dist)
support(dist)
generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

`dist_logistic` *The Logistic distribution*

Description

[Stable]

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

```
dist_logistic(location, scale)
```

Arguments

`location, scale`
location and scale parameters.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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.

See Also

stats::Logistic

Examples

```
dist <- dist_logistic(location = c(5,9,9,6,2), scale = c(2,3,4,2,1))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_lognormal *The log-normal distribution*

Description

[Stable]

The log-normal distribution is a commonly used transformation of the Normal distribution. If X follows a log-normal distribution, then $\ln X$ would be characterised by a Normal distribution.

Usage

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

Arguments

mu	The mean (location parameter) of the distribution, which is the mean of the associated Normal distribution. Can be any real number.
sigma	The standard deviation (scale parameter) of the distribution. Can be any positive number.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

In the following, let Y be a Normal random variable with mean `mu` = μ and standard deviation `sigma` = σ. The log-normal distribution $X = exp(Y)$ is characterised by:

Support: $R+$, the set of all real numbers greater than or equal to 0.

Mean: $e^(\mu + \sigma^2/2$

Variance: $(e^(\sigma^2) - 1)e^(2\mu + \sigma^2$

Probability density function (p.d.f):

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

Cumulative distribution function (c.d.f):

The cumulative distribution function has the form

$$F(x) = \Phi((\ln x - \mu)/\sigma)$$

Where Phi is the CDF of a standard Normal distribution, N(0,1).

See Also

stats::Lognormal

Examples

```
dist <- dist_lognormal(mu = 1:5, sigma = 0.1)

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)

# A log-normal distribution X is exp(Y), where Y is a Normal distribution of
# the same parameters. So log(X) will produce the Normal distribution Y.
log(dist)
```

`dist_missing` *Missing distribution*

Description

[Maturing]

A placeholder distribution for handling missing values in a vector of distributions.

Usage

```
dist_missing(length = 1)
```

Arguments

`length`	The number of missing distributions

Examples

```
dist <- dist_missing(3L)

dist
mean(dist)
variance(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_mixture *Create a mixture of distributions*

Description

[Maturing]

Usage

```
dist_mixture(..., weights = numeric())
```

Arguments

`...`	Distributions to be used in the mixture.
`weights`	The weight of each distribution passed to `...`.

Examples

```
dist_mixture(dist_normal(0, 1), dist_normal(5, 2), weights = c(0.3, 0.7))
```

dist_multinomial *The Multinomial distribution*

Description

[Stable]

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

Usage

```
dist_multinomial(size, prob)
```

Arguments

size	The number of draws from the Categorical distribution.
prob	The probability of an event occurring from each draw.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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):

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

See Also

stats::Multinomial

Examples

```
dist <- dist_multinomial(size = c(4, 3), prob = list(c(0.3, 0.5, 0.2), c(0.1, 0.5, 0.4)))

dist
mean(dist)
variance(dist)

generate(dist, 10)

# TODO: Needs fixing to support multiple inputs
# density(dist, 2)
# density(dist, 2, log = TRUE)
```

`dist_multivariate_normal`

The multivariate normal distribution

Description

[Stable]

Usage

```
dist_multivariate_normal(mu = 0, sigma = diag(1))
```

Arguments

mu	A list of numeric vectors for the distribution's mean.
sigma	A list of matrices for the distribution's variance-covariance matrix.

See Also

mvtnorm::dmvnorm, mvtnorm::qmvnorm

Examples

```
dist <- dist_multivariate_normal(mu = list(c(1,2)), sigma = list(matrix(c(4,2,2,3), ncol=2)))
dist


mean(dist)
variance(dist)
support(dist)
generate(dist, 10)

density(dist, c(2, 1))
density(dist, c(2, 1), log = TRUE)
```

```
cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_negative_binomial
The Negative Binomial distribution

Description

[Stable]

A generalization of the geometric distribution. It is the number of failures in a sequence of i.i.d. Bernoulli trials before a specified number of successes (`size`) occur. The probability of success in each trial is given by `prob`.

Usage

```
dist_negative_binomial(size, prob)
```

Arguments

`size`	target for number of successful trials, or dispersion parameter (the shape parameter of the gamma mixing distribution). Must be strictly positive, need not be integer.
`prob`	probability of success in each trial. `0` < `prob` <= `1`.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

In the following, let X be a Negative Binomial random variable with success probability `prob` = p and the number of successes `size` = r.

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, omitted.

Moment generating function (m.g.f):

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

See Also

stats::NegBinomial

Examples

```
dist <- dist_negative_binomial(size = 10, prob = 0.5)

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)
support(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_normal *The Normal distribution*

Description

[Stable]

The Normal distribution is ubiquitous 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

```
dist_normal(mu = 0, sigma = 1, mean = mu, sd = sigma)
```

Arguments

mu, mean	The mean (location parameter) of the distribution, which is also the mean of the distribution. Can be any real number.
sigma, sd	The standard deviation (scale parameter) of the distribution. Can be any positive number. 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://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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):

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

See Also

stats::Normal

Examples

```
dist <- dist_normal(mu = 1:5, sigma = 3)

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

dist_pareto *The Pareto distribution*

Description

[Stable]

Usage

```
dist_pareto(shape, scale)
```

Arguments

shape, scale	parameters. Must be strictly positive.

See Also

actuar::Pareto

Examples

```
dist <- dist_pareto(shape = c(10, 3, 2, 1), scale = rep(1, 4))
dist


mean(dist)
variance(dist)
support(dist)
generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

dist_percentile *Percentile distribution*

Description

[Stable]

Usage

```
dist_percentile(x, percentile)
```

Arguments

`x`	A list of values
`percentile`	A list of percentiles

Examples

```
dist <- dist_normal()
percentiles <- seq(0.01, 0.99, by = 0.01)
x <- vapply(percentiles, quantile, double(1L), x = dist)
dist_percentile(list(x), list(percentiles*100))
```

`dist_poisson` *The Poisson Distribution*

Description

[Stable]

Poisson distributions are frequently used to model counts.

Usage

```
dist_poisson(lambda)
```

Arguments

`lambda`	vector of (non-negative) means.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

In the following, let X be a Poisson random variable with parameter `lambda` = 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):

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

See Also

stats::Poisson

Examples

```
dist <- dist_poisson(lambda = c(1, 4, 10))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_poisson_inverse_gaussian

The Poisson-Inverse Gaussian distribution

Description

[Stable]

Usage

```
dist_poisson_inverse_gaussian(mean, shape)
```

Arguments

mean, shape	parameters. Must be strictly positive. Infinite values are supported.

See Also

actuar::PoissonInverseGaussian

Examples

```
dist <- dist_poisson_inverse_gaussian(mean = rep(0.1, 3), shape = c(0.4, 0.8, 1))
dist


mean(dist)
variance(dist)
support(dist)
generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

`dist_sample` *Sampling distribution*

Description

[Stable]

Usage

```
dist_sample(x)
```

Arguments

x	A list of sampled values.

Examples

```
# Univariate numeric samples
dist <- dist_sample(x = list(rnorm(100), rnorm(100, 10)))

dist
mean(dist)
variance(dist)
skewness(dist)
generate(dist, 10)

density(dist, 1)

# Multivariate numeric samples
dist <- dist_sample(x = list(cbind(rnorm(100), rnorm(100, 10))))
```

```
dist
mean(dist)
variance(dist)
skewness(dist)
generate(dist, 10)

density(dist, 1)
```

`dist_studentized_range` *The Studentized Range distribution*

Description

[Stable]

Tukey's studentized range distribution, used for Tukey's honestly significant differences test in ANOVA.

Usage

```
dist_studentized_range(nmeans, df, nranges)
```

Arguments

`nmeans`	sample size for range (same for each group).
`df`	degrees of freedom for s (see below).
`nranges`	number of *groups* whose **maximum** range is considered.

Details

We recommend reading this documentation on `https://pkg.mitchelloharawild.com/distributional/`, where the math will render nicely.

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.

See Also

stats::Tukey

Examples

```
dist <- dist_studentized_range(nmeans = c(6, 2), df = c(5, 4), nranges = c(1, 1))

dist

cdf(dist, 4)

quantile(dist, 0.7)
```

`dist_student_t` *The (non-central) location-scale Student t Distribution*

Description

[Stable]

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

```
dist_student_t(df, mu = 0, sigma = 1, ncp = NULL)
```

Arguments

`df`	degrees of freedom (> 0, maybe non-integer). `df = Inf` is allowed.
`mu`	The location parameter of the distribution. If `ncp == 0` (or `NULL`), this is the median.
`sigma`	The scale parameter of the distribution.
`ncp`	non-centrality parameter δ; currently except for `rt()`, only for `abs(ncp) <= 37.62`. If omitted, use the central t distribution.

Details

We recommend reading this documentation on `https://pkg.mitchelloharawild.com/distributional/`, where the math will render nicely.

In the following, let X be a **central** 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):

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

See Also

stats::TDist

Examples

```
dist <- dist_student_t(df = c(1,2,5), mu = c(0,1,2), sigma = c(1,2,3))

dist
mean(dist)
variance(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_transformed *Modify a distribution with a transformation*

Description

[Maturing]

The `density()`, `mean()`, and `variance()` methods are approximate as they are based on numerical derivatives.

Usage

```
dist_transformed(dist, transform, inverse)
```

Arguments

`dist`	A univariate distribution vector.
`transform`	A function used to transform the distribution. This transformation should be monotonic over appropriate domain.
`inverse`	The inverse of the `transform` function.

Examples

```
# Create a log normal distribution
dist <- dist_transformed(dist_normal(0, 0.5), exp, log)
density(dist, 1) # dlnorm(1, 0, 0.5)
cdf(dist, 4) # plnorm(4, 0, 0.5)
quantile(dist, 0.1) # qlnorm(0.1, 0, 0.5)
generate(dist, 10) # rlnorm(10, 0, 0.5)
```

---

dist_truncated *Truncate a distribution*

Description

[Stable]

Note that the samples are generated using inverse transform sampling, and the means and variances are estimated from samples.

Usage

```
dist_truncated(dist, lower = -Inf, upper = Inf)
```

Arguments

dist	The distribution(s) to truncate.
lower, upper	The range of values to keep from a distribution.

Examples

```
dist <- dist_truncated(dist_normal(2,1), lower = 0)

dist
mean(dist)
variance(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)

if(requireNamespace("ggdist")) {
library(ggplot2)
ggplot() +
  ggdist::stat_dist_halfeye(
```

```
    aes(y = c("Normal", "Truncated"),
        dist = c(dist_normal(2,1), dist_truncated(dist_normal(2,1), lower = 0)))
  )
}
```

---

dist_uniform *The Uniform distribution*

Description

[Stable]

A distribution with constant density on an interval.

Usage

```
dist_uniform(min, max)
```

Arguments

`min, max`	lower and upper limits of the distribution. Must be finite.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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

Support: $[a, b]$

Mean: $\frac{1}{2}(a + b)$

Variance: $\frac{1}{12}(b - a)^2$

Probability mass function (p.m.f):

$$f(x) = \frac{1}{b - a} for x \in [a, b]$$

$$f(x) = 0 otherwise$$

Cumulative distribution function (c.d.f):

$$F(x) = 0 for x < a$$

$$F(x) = \frac{x - a}{b - a} for x \in [a, b]$$

$$F(x) = 1 for x > b$$

Moment generating function (m.g.f):

$$E(e^{tX}) = \frac{e^{tb} - e^{ta}}{t(b - a)} for t \neq 0$$

$$E(e^{tX}) = 1 for t = 0$$

See Also

stats::Uniform

Examples

```
dist <- dist_uniform(min = c(3, -2), max = c(5, 4))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

---

dist_weibull — *The Weibull distribution*

Description

[Stable]

Generalization of the gamma distribution. Often used in survival and time-to-event analyses.

Usage

```
dist_weibull(shape, scale)
```

Arguments

shape, scale	shape and scale parameters, the latter defaulting to 1.

Details

We recommend reading this documentation on https://pkg.mitchelloharawild.com/distributional/, where the math will render nicely.

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):

$$f(x) = \frac{k}{\lambda}(\frac{x}{\lambda})^{k-1}e^{-(x/\lambda)^k}, x \ge 0$$

Cumulative distribution function (c.d.f):

$$F(x) = 1 - e^{-(x/\lambda)^k}, x \ge 0$$

Moment generating function (m.g.f):

$$\sum_{n=0}^{\infty} \frac{t^n \lambda^n}{n!} \Gamma(1 + n/k), k \ge 1$$

See Also

stats::Weibull

Examples

```
dist <- dist_weibull(shape = c(0.5, 1, 1.5, 5), scale = rep(1, 4))

dist
mean(dist)
variance(dist)
skewness(dist)
kurtosis(dist)

generate(dist, 10)

density(dist, 2)
density(dist, 2, log = TRUE)

cdf(dist, 4)

quantile(dist, 0.7)
```

`dist_wrap` *Create a distribution from p/d/q/r style functions*

Description

[Maturing]

If a distribution is not yet supported, you can vectorise p/d/q/r functions using this function. `dist_wrap()` stores the distributions parameters, and provides wrappers which call the appropriate p/d/q/r functions.

Using this function to wrap a distribution should only be done if the distribution is not yet available in this package. If you need a distribution which isn't in the package yet, consider making a request at https://github.com/mitchelloharawild/distributional/issues.

Usage

```
dist_wrap(dist, ..., package = NULL)
```

Arguments

dist	The name of the distribution used in the functions (name that is prefixed by p/d/q/r)
...	Named arguments used to parameterise the distribution.
package	The package from which the distribution is provided. If NULL, the calling environment's search path is used to find the distribution functions. Alternatively, an arbitrary environment can also be provided here.

Examples

```
dist <- dist_wrap("norm", mean = 1:3, sd = c(3, 9, 2))

density(dist, 1) # dnorm()
cdf(dist, 4) # pnorm()
quantile(dist, 0.975) # qnorm()
generate(dist, 10) # rnorm()

library(actuar)
dist <- dist_wrap("invparalogis", package = "actuar", shape = 2, rate = 2)
density(dist, 1) # actuar::dinvparalogis()
cdf(dist, 4) # actuar::pinvparalogis()
quantile(dist, 0.975) # actuar::qinvparalogis()
generate(dist, 10) # actuar::rinvparalogis()
```

family.distribution *Extract the name of the distribution family*

Description

[Experimental]

Usage

```
## S3 method for class 'distribution'
family(object, ...)
```

Arguments

object	The distribution(s).
...	Additional arguments used by methods.

Examples

```
dist <- c(
  dist_normal(1:2),
  dist_poisson(3),
  dist_multinomial(size = c(4, 3),
  prob = list(c(0.3, 0.5, 0.2), c(0.1, 0.5, 0.4)))
  )
family(dist)
```

---

generate.distribution *Randomly sample values from a distribution*

Description

[Stable]

Generate random samples from probability distributions.

Usage

```
## S3 method for class 'distribution'
generate(x, times, ...)
```

Arguments

x	The distribution(s).
times	The number of samples.
...	Additional arguments used by methods.

---

geom_hilo_linerange *Line ranges for hilo intervals*

Description

[Deprecated]

This function is deprecated in favour of the ggdist package and will removed in a future release of this package. Consider using `ggdist::stat_slabinterval()` or `ggdist::geom_slabinterval()` as an appropriate alternative.

`geom_hilo_linerange()` displays the interval defined by a hilo object. The luminance of the shaded area indicates its confidence level. The shade colour can be controlled by the `fill` aesthetic, however the luminance will be overwritten to represent the confidence level.

Usage

```
geom_hilo_linerange(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.
...	Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

See Also

`geom_hilo_ribbon()` for continuous hilo intervals (ribbons)

Examples

```
dist <- dist_normal(1:3, 1:3)
library(ggplot2)
ggplot(
  data.frame(x = rep(1:3, 2), interval = c(hilo(dist, 80), hilo(dist, 95)))
) +
  geom_hilo_linerange(aes(x = x, hilo = interval))
```

---

geom_hilo_ribbon *Ribbon plots for hilo intervals*

Description

[Deprecated]

This function is deprecated in favour of the ggdist package and will removed in a future release of this package. Consider using `ggdist::stat_lineribbon()` or `ggdist::geom_lineribbon()` as an appropriate alternative.

`geom_hilo_ribbon()` displays the interval defined by a hilo object. The luminance of the shaded area indicates its confidence level. The shade colour can be controlled by the `fill` aesthetic, however the luminance will be overwritten to represent the confidence level.

Usage

```
geom_hilo_ribbon(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

`mapping` Set of aesthetic mappings created by `aes()` or `aes_()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

`data` The data to be displayed in this layer. There are three options:

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

	A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`stat`	The statistical transformation to use on the data for this layer, as a string.
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display.
`inherit.aes`	If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. `borders()`.
`...`	Other arguments passed on to `layer()`. These are often aesthetics, used to set an aesthetic to a fixed value, like `colour = "red"` or `size = 3`. They may also be parameters to the paired geom/stat.

See Also

`geom_hilo_linerange()` for discrete hilo intervals (vertical lines)

Examples

```
dist <- dist_normal(1:3, 1:3)
library(ggplot2)
ggplot(
  data.frame(x = rep(1:3, 2), interval = c(hilo(dist, 80), hilo(dist, 95)))
) +
  geom_hilo_ribbon(aes(x = x, hilo = interval))
```

`guide_level` *Level shade bar guide*

Description

The level guide shows the colour from the forecast intervals which is blended with the series colour.

Usage

```
guide_level(title = waiver(), max_discrete = 5, ...)
```

Arguments

title	A character string or expression indicating a title of guide. If NULL, the title is not shown. By default (waiver()), the name of the scale object or the name specified in labs() is used for the title.
max_discrete	The maximum number of levels to be shown using ggplot2::guide_legend(). If the number of levels exceeds this value, level shades are shown with ggplot2::guide_colourbar().
...	Further arguments passed onto either ggplot2::guide_colourbar() or ggplot2::guide_legend()

---

hdr *Compute highest density regions*

Description

Used to extract a specified prediction interval at a particular confidence level from a distribution.

Usage

```
hdr(x, ...)
```

Arguments

x	Object to create hilo from.
...	Additional arguments used by methods.

---

hdr.distribution *Highest density regions of probability distributions*

Description

[Maturing]

This function is highly experimental and will change in the future. In particular, improved functionality for object classes and visualisation tools will be added in a future release.

Computes minimally sized probability intervals highest density regions.

Usage

```
## S3 method for class 'distribution'
hdr(x, size = 95, n = 512, ...)
```

Arguments

x	The distribution(s).
size	The size of the interval (between 0 and 100).
n	The resolution used to estimate the distribution's density.
...	Additional arguments used by methods.

hilo *Compute intervals*

Description

[Stable]

Used to extract a specified prediction interval at a particular confidence level from a distribution.

The numeric lower and upper bounds can be extracted from the interval using `<hilo>$lower` and `<hilo>$upper` as shown in the examples below.

Usage

```
hilo(x, ...)
```

Arguments

x	Object to create hilo from.
...	Additional arguments used by methods.

Examples

```
# 95% interval from a standard normal distribution
interval <- hilo(dist_normal(0, 1), 95)
interval

# Extract the individual quantities with `$lower`, `$upper`, and `$level`
interval$lower
interval$upper
interval$level
```

hilo.distribution *Probability intervals of a probability distribution*

Description

[Stable]

Returns a `hilo` central probability interval with probability coverage of `size`. By default, the distribution's `quantile()` will be used to compute the lower and upper bound for a centered interval

Usage

```
## S3 method for class 'distribution'
hilo(x, size = 95, ...)
```

Arguments

`x`	The distribution(s).
`size`	The size of the interval (between 0 and 100).
`...`	Additional arguments used by methods.

See Also

`hdr.distribution()`

`is_distribution` *Test if the object is a distribution*

Description

[Stable]

This function returns `TRUE` for distributions and `FALSE` for all other objects.

Usage

```
is_distribution(x)
```

Arguments

`x`	An object.

Value

TRUE if the object inherits from the distribution class.

Examples

```
dist <- dist_normal()
is_distribution(dist)
is_distribution("distributional")
```

is_hdr *Is the object a hdr*

Description

Is the object a hdr

Usage

```
is_hdr(x)
```

Arguments

x	An object.

is_hilo *Is the object a hilo*

Description

Is the object a hilo

Usage

```
is_hilo(x)
```

Arguments

x	An object.

kurtosis *Kurtosis of a probability distribution*

Description

[Stable]

Usage

```
kurtosis(x, ...)

## S3 method for class 'distribution'
kurtosis(x, ...)
```

Arguments

`x`	The distribution(s).
`...`	Additional arguments used by methods.

`likelihood` *The (log) likelihood of a sample matching a distribution*

Description

[Stable]

Usage

```
likelihood(x, ...)

## S3 method for class 'distribution'
likelihood(x, sample, ..., log = FALSE)

log_likelihood(x, ...)
```

Arguments

`x`	The distribution(s).
`...`	Additional arguments used by methods.
`sample`	A list of sampled values to compare to distribution(s).
`log`	If `TRUE`, the log-likelihood will be computed.

`mean.distribution` *Mean of a probability distribution*

Description

[Stable]

Returns the empirical mean of the probability distribution. If the method does not exist, the mean of a random sample will be returned.

Usage

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

Arguments

`x`	The distribution(s).
`...`	Additional arguments used by methods.

median.distribution *Median of a probability distribution*

Description

[Stable]

Returns the median (50th percentile) of a probability distribution. This is equivalent to `quantile(x, p=0.5)`.

Usage

```
## S3 method for class 'distribution'
median(x, na.rm = FALSE, ...)
```

Arguments

`x`	The distribution(s).
`na.rm`	Unused, included for consistency with the generic function.
`...`	Additional arguments used by methods.

new_dist *Create a new distribution*

Description

[Maturing]

Allows extension package developers to define a new distribution class compatible with the distributional package.

Usage

```
new_dist(..., class = NULL, dimnames = NULL)
```

Arguments

`...`	Parameters of the distribution (named).
`class`	The class of the distribution for S3 dispatch.
`dimnames`	The names of the variables in the distribution (optional).

new_hdr *Construct hdr intervals*

Description

Construct hdr intervals

Usage

```
new_hdr(
  lower = list_of(.ptype = double()),
  upper = list_of(.ptype = double()),
  size = double()
)
```

Arguments

lower, upper	A list of numeric vectors specifying the region's lower and upper bounds.
size	A numeric vector specifying the coverage size of the region.

Value

A "hdr" vector

Author(s)

Mitchell O'Hara-Wild

Examples

```
new_hdr(lower = list(1, c(3,6)), upper = list(10, c(5, 8)), size = c(80, 95))
```

new_hilo *Construct hilo intervals*

Description

[Stable]

Class constructor function to help with manually creating hilo interval objects.

Usage

```
new_hilo(lower = double(), upper = double(), size = double())
```

Arguments

lower, upper	A numeric vector of values for lower and upper limits.
size	Size of the interval between [0, 100].

Value

A "hilo" vector

Author(s)

Earo Wang & Mitchell O'Hara-Wild

Examples

```
new_hilo(lower = rnorm(10), upper = rnorm(10) + 5, size = 95)
```

---

new_support_region *Create a new support region vector*

Description

Create a new support region vector

Usage

```
new_support_region(x, limits = NULL)
```

Arguments

x	A list of prototype vectors defining the distribution type.
limits	A list of value limits for the distribution.

---

parameters *Extract the parameters of a distribution*

Description

[Experimental]

Usage

```
parameters(x, ...)

## S3 method for class 'distribution'
parameters(x, ...)
```

Arguments

`x`	The distribution(s).
`...`	Additional arguments used by methods.

Examples

```
dist <- c(
  dist_normal(1:2),
  dist_poisson(3),
  dist_multinomial(size = c(4, 3),
  prob = list(c(0.3, 0.5, 0.2), c(0.1, 0.5, 0.4)))
  )
parameters(dist)
```

`quantile.distribution` *Distribution Quantiles*

Description

[Stable]

Computes the quantiles of a distribution.

Usage

```
## S3 method for class 'distribution'
quantile(x, p, ..., log = FALSE)
```

Arguments

`x`	The distribution(s).
`p`	The probability of the quantile.
`...`	Additional arguments passed to methods.
`log`	If `TRUE`, probabilities will be given as log probabilities.

`scale_hilo_continuous` *Hilo interval scales*

Description

Hilo interval scales

Usage

```
scale_hilo_continuous(
  name = waiver(),
  breaks = waiver(),
  minor_breaks = waiver(),
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  oob = identity,
  na.value = NA,
  trans = "identity",
  guide = waiver(),
  position = "left",
  sec.axis = waiver()
)
```

Arguments

`name` The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

`breaks` One of:

- `NULL` for no breaks
- `waiver()` for the default breaks computed by the transformation object
- A numeric vector of positions
- A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Also accepts rlang lambda function notation.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (one minor break between each major break)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation.

n.breaks
: An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.

labels
: One of:
  - `NULL` for no labels
  - `waiver()` for the default labels computed by the transformation object
  - A character vector giving labels (must be same length as `breaks`)
  - A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

limits
: One of:
  - `NULL` to use the default scale range
  - A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum
  - A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang lambda function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see `coord_cartesian()`).

expand
: For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `expansion()` to generate the values for the `expand` argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

oob
: One of:
  - Function that handles limits outside of the scale limits (out of bounds). Also accepts rlang lambda function notation.
  - The default (`scales::censor()`) replaces out of bounds values with `NA`.
  - `scales::squish()` for squishing out of bounds values into range.
  - `scales::squish_infinite()` for squishing infinite values into range.

na.value
: Missing values will be replaced with this value.

trans
: For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time".

  A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called `<name>_trans` (e.g., `scales::boxcox_trans()`). You can create your own transformation with `scales::trans_new()`.

guide
: A function used to create a guide or its name. See `guides()` for more information.

position
: For position scales, The position of the axis. `left` or `right` for y axes, `top` or `bottom` for x axes.

sec.axis
: `sec_axis()` is used to specify a secondary axis.

`scale_level` *level luminance scales*

Description

This set of scales defines new scales for prob geoms equivalent to the ones already defined by ggplot2. This allows the shade of confidence intervals to work with the legend output.

Usage

```
scale_level_continuous(..., guide = "level")
```

Arguments

`...` Arguments passed on to `continuous_scale`

`scale_name` The name of the scale that should be used for error messages associated with this scale.

`palette` A palette function that when called with a numeric vector with values between 0 and 1 returns the corresponding output values (e.g., `scales::area_pal()`).

`name` The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

`breaks` One of:

- `NULL` for no breaks
- `waiver()` for the default breaks computed by the transformation object
- A numeric vector of positions
- A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Also accepts rlang lambda function notation.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (one minor break between each major break)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation.

`n.breaks` An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.

`labels` One of:

- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as `breaks`)

- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

limits One of:

- NULL to use the default scale range
- A numeric vector of length two providing limits of the scale. Use NA to refer to the existing minimum or maximum
- A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang lambda function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see coord_cartesian()).

rescaler A function used to scale the input values to the range [0, 1]. This is always scales::rescale(), except for diverging and n colour gradients (i.e., scale_colour_gradient2(), scale_colour_gradientn()). The rescaler is ignored by position scales, which always use scales::rescale(). Also accepts rlang lambda function notation.

oob One of:

- Function that handles limits outside of the scale limits (out of bounds). Also accepts rlang lambda function notation.
- The default (scales::censor()) replaces out of bounds values with NA.
- scales::squish() for squishing out of bounds values into range.
- scales::squish_infinite() for squishing infinite values into range.

trans For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time".

A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called <name>_trans (e.g., scales::boxcox_trans()). You can create your own transformation with scales::trans_new().

expand For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function expansion() to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

position For position scales, The position of the axis. left or right for y axes, top or bottom for x axes.

super The super class to use for the constructed scale

guide Type of legend. Use "colourbar" for continuous colour bar, or "legend" for discrete colour legend.

Value

A ggproto object inheriting from Scale

skewness — *Skewness of a probability distribution*

Description

[Stable]

Usage

```
skewness(x, ...)

## S3 method for class 'distribution'
skewness(x, ...)
```

Arguments

x	The distribution(s).
...	Additional arguments used by methods.

support — *Region of support of a distribution*

Description

[Experimental]

Usage

```
support(x, ...)

## S3 method for class 'distribution'
support(x, ...)
```

Arguments

x	The distribution(s).
...	Additional arguments used by methods.

variance *Variance*

Description

[Stable]

A generic function for computing the variance of an object.

Usage

```
variance(x, ...)

## S3 method for class 'numeric'
variance(x, ...)

## S3 method for class 'matrix'
variance(x, ...)

## S3 method for class 'numeric'
covariance(x, ...)
```

Arguments

`x`	An object.
`...`	Additional arguments used by methods.

Details

The implementation of `variance()` for numeric variables coerces the input to a vector then uses `stats::var()` to compute the variance. This means that, unlike `stats::var()`, if `variance()` is passed a matrix or a 2-dimensional array, it will still return the variance (`stats::var()` returns the covariance matrix in that case).

See Also

`variance.distribution()`, `covariance()`

variance.distribution *Variance of a probability distribution*

Description

[Stable]

Returns the empirical variance of the probability distribution. If the method does not exist, the variance of a random sample will be returned.

Usage

```
## S3 method for class 'distribution'
variance(x, ...)
```

Arguments

x	The distribution(s).
...	Additional arguments used by methods.

Index