Package: pitw (via r-universe)

November 26, 2026

Title PIT-Wasserstein Metrics for Diagnosing Dispersion Miscalibration

Version 0.0.1

Description Implements the PIT-Wasserstein family of metrics for diagnosing dispersion miscalibration in probabilistic estimators. The package provides three nested metrics—B_pit (baseline), U_pit (bias-adjusted undirected), and D_pit (bias-adjusted signed, detecting overconfidence vs. underconfidence)—each with a significance test based on a large-sample reference distribution for independent predictions. Also provides batch scoring, CRPS calculation, simulation helpers, and diagnostic visualisation tools.

License MIT + file LICENSE

Config/roxygen2/version 8.1.1

Encoding UTF-8

Roxygen list(markdown = TRUE, packages = ``pkgtools", roclets = c(``collate", ``namespace", ``pkgtools::pkgtools_rd", ``pkgtools::pkgtools_unit"))

Imports base64enc, dplyr, flextable, fs, ggplot2, grDevices, htmltools, huxtable, interfacer (>= 0.4.2), knitr, magrittr, officer, pdftools, purrr, ragg, readr, rlang, rmarkdown, rstudioapi, rsvg, scales, stats, stringr, svglite, systemfonts, scoringRules, testthat, tibble, tidyr, tidyselect, utils

Suggests deSolve, EpiEstim, devtools, forcats, here, patchwork, tidyverse

URL https://ai4ci.github.io/pit-wasserstein, https://github.com/ai4ci/pit-wasserstein, https://doi.org/10.5281/zenodo.24555797

Config/Needs/build terminological/pkgtools

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev texlive libicu-dev libjpeg-dev libpng-dev librsvg2-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev libpoppler-cpp-dev poppler-data libx11-dev

Repository https://ai4ci.r-universe.dev

Date/Publication 2026-11-26 14:50:28 UTC

RemoteUrl https://github.com/ai4ci/pit-wasserstein

RemoteRef 0.0.1

RemoteSha 2dabde77a8f9f4ce8a671e343b08d2f74728a015

Contents

b_pit_significance . . . 3
calibrated_locations . . . 5
calibrated_locations_general . . . 6
configuration_plot . . . 6
crps_fn_generator . . . 7
d_pit_significance . . . 8
dbeta2 . . . 10
dcgamma . . . 11
ddist . . . 12
dgamma2 . . . 12
dgamma4 . . . 13
dlnorm2 . . . 14
dlnorm4 . . . 15
dlogitnorm . . . 17
dlogitnorm2 . . . 17
dnbinom2 . . . 18
dnorm4 . . . 20
dnull . . . 22
dwedge . . . 22
generate_simulation . . . 23
p_l1_bridge . . . 24
pbeta2 . . . 25
pcgamma . . . 26
pdist . . . 27
pgamma2 . . . 27
pgamma4 . . . 28
pit_w . . . 29
plnorm2 . . . 30
plnorm4 . . . 31
plogitnorm . . . 33
plogitnorm2 . . . 34
pnbinom2 . . . 34
pnorm4 . . . 36
pnull . . . 38
prediction_scores . . . 39

pwedge . . . 40
qbeta2 . . . 41
qcgamma . . . 42
qdist . . . 43
qgamma2 . . . 43
qgamma4 . . . 44
qlnorm2 . . . 45
qlnorm4 . . . 46
qlogitnorm . . . 48
qlogitnorm2 . . . 48
qnbinom2 . . . 49
qnorm4 . . . 51
qnull . . . 53
qwedge . . . 53
rbern . . . 54
rbeta2 . . . 55
rcategorical . . . 56
rcgamma . . . 56
rdist . . . 57
reparam_fn . . . 58
rexpgrowth . . . 58
rexpgrowthI0 . . . 59
rgamma2 . . . 60
rgamma4 . . . 61
rlnorm2 . . . 62
rlnorm4 . . . 63
rlogitnorm . . . 65
rlogitnorm2 . . . 65
rnbinom2 . . . 66
rnorm4 . . . 67
rnull . . . 69
rwedge . . . 70
std_size . . . 71
u_pit_significance . . . 71
wedge . . . 72

Index **74**

`b_pit_significance` *Significance test for the baseline PIT-Wasserstein metric*

Description

Tests the strongest of the three null hypotheses in the PIT-Wasserstein family: that the predictions are **fully probabilistically calibrated**, so that the raw PIT values are i.i.d. $U(0,1)$ — correct location *and* correct dispersion *and* correct shape. The test is one-sided: `B_pit` is a distance, so only large values are evidence against calibration.

Usage

```
b_pit_significance(B_pit, n_predictions)
```

Arguments

`B_pit`	the baseline (unadjusted) PIT-Wasserstein metric, reported as `B_pit` by `pit_w()` and `prediction_scores()`
`n_predictions`	the number of predictions each `B_pit` was computed from, as returned by `prediction_scores()`

Details

Under that null $\sqrt{n}\,B_{pit}$ converges to $\int_0^1 |B^\circ|$, the integrated absolute Brownian bridge, whose c.d.f. is `p_l1_bridge()`. Nothing is estimated from the data, so the null is exact and fully analytical.

`B_pit` rejects when *anything* is wrong and says nothing about what. Use it as the omnibus first stage, then `d_pit_significance()` to test dispersion free of any location error, and `u_pit_significance()` for the residual shape.

Value

a dataframe with one row per input value and columns:

- `B_pit`, `n_predictions` — as supplied
- `statistic` — $\sqrt{n}\,B_{pit}$, the quantity with a fixed null
- `expected` — the null mean of `B_pit`, $\sqrt{2\pi}/(8\sqrt{n})$
- `critical_0.05` — the one-sided 5% critical value of `B_pit`
- `p_value` — the probability of a `B_pit` at least this large under full probabilistic calibration

Unit tests

```
# a calibrated forecaster sits near the null mean and is not significant
res = b_pit_significance(sqrt(2 * pi) / 8 / sqrt(100), 100)
testthat::expect_true(res$p_value > 0.3)

# the 5% critical value has, by construction, a p value of 0.05
crit = b_pit_significance(0, 100)$critical_0.05
testthat::expect_equal(b_pit_significance(crit, 100)$p_value, 0.05, tolerance = 1e-5)

# the critical value scales as 1/sqrt(n)
testthat::expect_equal(
  b_pit_significance(0, 400)$critical_0.05 * 2,
  b_pit_significance(0, 100)$critical_0.05,
  tolerance = 1e-6
)

# and is strictly larger than the adjusted test's, which has estimated a
# location parameter from the same data
```

```
testthat::expect_true(
 b_pit_significance(0, 100)$critical_0.05 > u_pit_significance(0, 100)$critical_0.05
)

# vectorised over both arguments
testthat::expect_equal(nrow(b_pit_significance(c(0.02, 0.09), c(76, 76))), 2L)
```

`calibrated_locations` *Uniform predictions for testing*

Description

Predictions whose PIT values are exactly uniform, for a scale family (gamma4, lnorm4, ...) where `location` is the median.

Usage

```
calibrated_locations(qfn, median, kappa, n = 100)
```

Arguments

`qfn`	a quantile function parametrised by median and something else
`median`	the median
`kappa`	a spread parameter, typically dispersion (sigma/mu)
`n`	the number of samples (default `` `100` `` [numeric])

Value

a set of locations for a distribution family that map directly to uniform PIT values.

Examples

```
loc = calibrated_locations(qfn = qgamma4, median = 1, kappa = 0.5)
pit_w(1, loc, 0.5, dist = "gamma4", link = "log")$B_pit # ~1e-17
```

calibrated_locations_general *Uniform predictions for testing (general)*

Description

For a general (non-scale) family, solve it numerically instead:

Usage

```
calibrated_locations_general(
  pfn,
  median,
  kappa,
  n = 100,
  interval = c(1e-08, 1e+08)
)
```

Arguments

pfn	a CDF parametrised by mean and the spread parameter
median	the median
kappa	a spread parameter, typically dispersion (sigma/mu)
n	the number of samples (default `100` [numeric])
interval	uniroot limits (default `c(1e-08, 1e+08)` [numeric])

Value

a set of locations for a distribution family that map directly to uniform PIT values.

configuration_plot *Produce a summary plot of a experimental configuration.*

Description

Produce a summary plot of a experimental configuration.

Usage

```
configuration_plot(config = i_sim_plot, xlim = c(NA, NA), labels = TRUE, ...)
```

Arguments

config	dataframe grouped by designed panels - a dataframe with columns: • bias_quantile (numeric + default(0.5)) - The quantile of bias of the plotted prediction • sharpness_quantile (numeric + default(0.5)) - The quantile of sharpness of the plotted prediction • true_value (numeric + default(0)) - the true value being estimated • estimator_dist (character + default("norm")) - the estimator distribution name (e.g. norm) parameterised by location (true_value+bias) and scale (sharpness) • bias_dist (character + default("norm")) - the distribution name of the estimator generation error function • bias_location (numeric + default(0)) - the location of the estimator generation error function, this is the true bias / mu_e • bias_scale (numeric + default(1)) - the scale of the estimator generation error function, this is the true uncertainty of the estimator / sigma_e • sharpness_dist (character + default("lnorm4")) - the distribution of the stated uncertainty of the estimator • sharpness_location (numeric + default(1)) - the location of the stated uncertainty of the estimator / sigma_s • sharpness_scale (numeric + default(0)) - the scale of the stated uncertainty of the estimator, determines how consistent the stated uncertainty is realtive to the true uncertainty / kappa, Any grouping allowed.
xlim	limits on the x-axis (default `c(NA, NA)` [logical])
labels	include labels on graphs - too cluttered on mulitple panels (default `TRUE` [logical])
...	addtionally passed on to

Value

a ggplot

---

crps_fn_generator *Generate a CRPS calculating function based on a parametrised distribution name*

Description

scoringRules has implementations of common functions but not so much for re-parametrisations. This detects if a re-parametrisation is available for a distribution name and remaps the input to a more common distribution that scoringRules has support for.

Usage

```
crps_fn_generator(dist_name)
```

Arguments

dist_name	the name of the distribution, e.g. norm, lnorm4

Value

a function which takes 3 parameters: true_value, location, and scale and returns a CRPS for them.

Unit tests

```
fn=crps_fn_generator("lnorm4")
mapped = fn(1:5,1,2)
tmp = .reparam.lnorm4(1,2)
direct = scoringRules::crps(1:5, tmp$dist, meanlog = tmp$meanlog, sdlog = tmp$sdlog)
testthat::expect_equal(mapped, direct)
```

---

d_pit_significance *Significance test for the directed PIT-Wasserstein metric*

Description

Tests the null hypothesis that the predictions are calibrated. D_pit is signed, so unlike u_pit_significance() this test reports a direction: positive values indicate **overconfidence** (predictive distributions too narrow) and negative values **underconfidence** (too wide).

Usage

```
d_pit_significance(
  D_pit,
  n_predictions,
  alternative = c("two.sided", "overconfident", "underconfident")
)
```

Arguments

D_pit	the directed PIT-Wasserstein metric, as returned by pit_w() or prediction_scores()
n_predictions	the number of predictions each D_pit was computed from, as returned by prediction_scores()
alternative	one of "two.sided" (any dispersion miscalibration), "overconfident" (positive D_pit only) or "underconfident" (negative D_pit only) (default `two.sided`; allowed: `two.sided`,`overconfident`,`underconfident`)

Details

D_pit is a linear functional of the empirical quantile process, so its null is exactly Gaussian with a known variance, $D^*_{pit} \sim \mathcal{N}(0, 1/(48n))$. No simulation, bootstrap or nuisance parameter is required.

The normal approximation to a sum of order statistics converges from above, so for n_predictions below about 50 the test is mildly anti-conservative (empirical size around 8% at n = 10, 6% at n = 25, 5.5% at n = 50); treat small-sample p values as approximate.

Value

a dataframe with one row per input value and columns:

- D_pit, n_predictions — as supplied
- statistic — the z score, $D^*_{pit}\sqrt{48n}$
- se — the null standard error, $1/\sqrt{48n}$
- critical_0.05 — the magnitude of D_pit rejected at 5%
- direction — "overconfident", "underconfident" or "calibrated"
- p_value — computed against alternative

Unit tests

```
# a z score of 1.96 is a two sided p of 0.05
res = d_pit_significance(qnorm(0.975) / sqrt(48 * 100), 100)
testthat::expect_equal(res$statistic, qnorm(0.975))
testthat::expect_equal(res$p_value, 0.05)
testthat::expect_equal(res$direction, "overconfident")

# the one sided test is half the two sided test in the expected direction
testthat::expect_equal(
  d_pit_significance(0.01, 100, "overconfident")$p_value,
  d_pit_significance(0.01, 100)$p_value / 2
)

# and is not fooled by a deviation in the other direction
testthat::expect_true(d_pit_significance(-0.03, 100, "overconfident")$p_value > 0.95)

# exactly calibrated
testthat::expect_equal(d_pit_significance(0, 100)$p_value, 1)
testthat::expect_equal(d_pit_significance(0, 100)$direction, "calibrated")
```

dbeta2 *The Beta Distribution*

Description

Density, distribution function, quantile function and random generation for the Beta distribution with parameters shape1 and shape2 (and optional non-centrality parameter ncp).

Usage

```
dbeta2(x, prob, kappa, log = FALSE)
```

Arguments

x	vector of quantiles
prob	the mean probability (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
log	logical; if TRUE, probabilities p are given as log(p).

Value

dbeta gives the density, pbeta is the cumulative distribution function, and qbeta is the quantile function of the Beta distribution. rbeta generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rbeta, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::dbeta()

Examples

```
dbeta2(c(0.25, 0.5, 0.75), 0.5, 0.25)
```

dcgamma *Density: gamma distribution constrained to have mean > sd*

Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

Usage

```
dcgamma(x, median, kappa = 1/median, log = FALSE)
```

Arguments

x	vector of quantiles
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
log	logical; if TRUE, probabilities p are given as log(p).

Value

dgamma gives the density, pgamma is the cumulative distribution function, and qgamma is the quantile function of the Gamma distribution. rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::dgamma()

Examples

```
dcgamma(seq(0, 4, 0.25), 2, 0.5)
```

ddist *Density function a named distribution*

Description

Density function a named distribution

Usage

```
ddist(dist, x, ...)
```

Arguments

dist	the distribution name e.g.lnorm
x	vector of quantiles
...	passed onto named functions

dgamma2 *The Gamma Distribution*

Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

Usage

```
dgamma2(x, mean, sd = sqrt(mean), log = FALSE)
```

Arguments

x	vector of quantiles
mean	the mean value on the true scale (vectorised)
sd	the standard deviation on the true scale (vectorised)
log	logical; if TRUE, probabilities p are given as log(p).

Value

dgamma gives the density, pgamma is the cumulative distribution function, and qgamma is the quantile function of the Gamma distribution. rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::dgamma()

Examples

```
dgamma2(seq(0, 4, 0.25), 2, 1)
```

---

dgamma4 *The Gamma Distribution*

Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

Usage

```
dgamma4(x, median, kappa = 1/median, log = FALSE)
```

Arguments

x	vector of quantiles
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
log	logical; if TRUE, probabilities p are given as log(p).

Value

dgamma gives the density, pgamma is the cumulative distribution function, and qgamma is the quantile function of the Gamma distribution. rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::dgamma()

Examples

```
dgamma2(seq(0, 4, 0.25), 2, 1)
```

dlnorm2 *The Log-Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the log-normal distribution whose logarithm has mean equal to `meanlog` and standard deviation equal to `sdlog`.

Usage

```
dlnorm2(x, mean = 1, sd = sqrt(exp(1) - 1), log = FALSE)
```

Arguments

`x`	vector of quantiles
`mean`	the mean value on the true scale (vectorised)
`sd`	the standard deviation on the true scale (vectorised)
`log`	logical; if TRUE, probabilities p are given as log(p).

Details

The log-normal distribution has density

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

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

Value

`dlnorm` gives the density, `plnorm` is the cumulative distribution function, and `qlnorm` is the quantile function of the log-normal distribution. `rlnorm` generates random deviates.

The length of the result is determined by `n` for `rlnorm`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

Note

The cumulative hazard $H(t) = -\log(1 - F(t))$ is `-plnorm(t, r, lower = FALSE, log = TRUE)`.

Source

`dlnorm` is calculated from the definition (in 'Details'). `[pqr]lnorm` are based on the relationship to the normal.

Consequently, they model a single point mass at `exp(meanlog)` for the boundary case `sdlog = 0`.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions*, volume 1. Wiley, New York. ISBN 978-0-471-58495-7.
Chapter 14.

See Also

Distributions for other standard distributions, including `dnorm` for the normal distribution.

Examples

```
dlnorm2(seq(0, 4, 0.25), 2, 1)
```

`dlnorm4` *The Log-Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the log-normal distribution whose logarithm has mean equal to `meanlog` and standard deviation equal to `sdlog`.

Usage

```
dlnorm4(x, median = 1, kappa = 1, log = FALSE)
```

Arguments

`x`	vector of quantiles
`median`	the median value on the true scale (vectorised)
`kappa`	a coefficient of variation (sd/mean). where 0 is no variability.
`log`	logical; if TRUE, probabilities p are given as log(p).

Details

The log-normal distribution has density

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

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

Value

`dlnorm` gives the density, `plnorm` is the cumulative distribution function, and `qlnorm` is the quantile function of the log-normal distribution. `rlnorm` generates random deviates.

The length of the result is determined by `n` for `rlnorm`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

Note

The cumulative hazard $H(t) = -\log(1 - F(t))$ is `-plnorm(t, r, lower = FALSE, log = TRUE)`.

Source

`dlnorm` is calculated from the definition (in 'Details'). `[pqr]lnorm` are based on the relationship to the normal.

Consequently, they model a single point mass at `exp(meanlog)` for the boundary case `sdlog = 0`.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions*, volume 1. Wiley, New York. ISBN 978-0-471-58495-7.
Chapter 14.

See Also

Distributions for other standard distributions, including `dnorm` for the normal distribution.

Examples

```
dlnorm4(seq(0, 4, 0.25), 2, 1)
```

dlogitnorm *Logit-normal distribution*

Description

The logit-normal distribution has a support of 0 to 1.

Usage

```
dlogitnorm(x, meanlogit = 0, sdlogit = 1, log = FALSE)
```

Arguments

x	vector of quantiles (0<x<1)
meanlogit	the mean on the logit scale
sdlogit	the sd on the logit scale
log	logical; if TRUE, probabilities p are given as log(p).

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
dlogitnorm(seq(0.1, 0.9, 0.1), 0, 1)
```

dlogitnorm2 *Logit-normal distribution*

Description

The logit-normal distribution has a support of 0 to 1.

Usage

```
dlogitnorm2(x, prob.0.5 = 0.5, kappa = 1 - exp(-1), log = FALSE)
```

Arguments

x	vector of quantiles (0<x<1)
prob.0.5	the median on the true scale
kappa	a dispersion parameter from 0 (none) to 1 maximum dispersion
log	logical; if TRUE, probabilities p are given as log(p).

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
q = seq(0.1, 0.9, 0.1)
eps = sqrt(.Machine$double.eps)
dlogitnorm2(q, 0.75, 0.2)

(plogitnorm2(q + eps, 0.75, 0.2) -
  plogitnorm2(q - eps, 0.75, 0.2)) / (2 * eps)
```

---

dnbinom2 *The Negative Binomial Distribution*

Description

Density, distribution function, quantile function and random generation for the negative binomial distribution with parameters `size` and `prob`.

Usage

```
dnbinom2(x, mean, sd = sqrt(mean), log = FALSE)
```

Arguments

`x`	vector of (non-negative integer) quantiles.
`mean`	the mean value on the true scale (vectorised)
`sd`	the standard deviation on the true scale (vectorised)
`log`	logical; if TRUE, probabilities p are given as log(p).

Details

The negative binomial distribution with `size` $= n$ and `prob` $= p$ has density

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

for $x = 0, 1, 2, \ldots$, $n > 0$ and $0 < p \leq 1$.

This represents the number of failures which occur in a sequence of Bernoulli trials before a target number of successes is reached. The mean is $\mu = n(1-p)/p$ and variance $n(1-p)/p^2$.

A negative binomial distribution can also arise as a mixture of Poisson distributions with mean distributed as a gamma distribution (see `pgamma`) with scale parameter `(1 - prob)/prob` and shape parameter `size`. (This definition allows non-integer values of `size`.)

An alternative parametrization (often used in ecology) is by the *mean* `mu` (see above), and `size`, the *dispersion parameter*, where `prob = size/(size+mu)`. The variance is `mu + mu^2/size` in this parametrization.

If an element of x is not integer, the result of dnbinom is zero, with a warning.

The case size == 0 is the distribution concentrated at zero. This is the limiting distribution for size approaching zero, even if mu rather than prob is held constant. Notice though, that the mean of the limit distribution is 0, whatever the value of mu.

The quantile is defined as the smallest value x such that $F(x) \geq p$, where F is the distribution function.

Value

dnbinom gives the density, pnbinom is the cumulative distribution function, and qnbinom is the quantile function of the negative binomial distribution. rnbinom generates random deviates.

Invalid size or prob will result in return value NaN, with a warning.

The length of the result is determined by n for rnbinom, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

rnbinom returns a vector of type integer unless generated values exceed the maximum representable integer when double values are returned.

Source

dnbinom computes via binomial probabilities, using code contributed by Catherine Loader (see dbinom).

pnbinom uses pbeta.

qnbinom uses the Cornish–Fisher Expansion to include a skewness correction to a normal approximation, followed by a search.

rnbinom uses the derivation as a gamma mixture of Poisson distributions, see Devroye (1986, page 480).

References

Devroye L (1986). *Non-Uniform Random Variate Generation*. Springer, New York. doi:10.1007/9781461386438. https://luc.devroye.org/handbooksimulation1.pdf.

See Also

Distributions for standard distributions, including dbinom for the binomial, dpois for the Poisson and dgeom for the geometric distribution, which is a special case of the negative binomial.

Examples

```
dnbinom2(0:5, 2, sqrt(2))
```

dnorm4 *The Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to mean and standard deviation equal to sd.

Usage

```
dnorm4(x, median = 0, kappa = 1, log = FALSE)
```

Arguments

x	vector of quantiles
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
log	logical; if TRUE, probabilities p are given as log(p).

Details

If mean or sd are not specified they assume the default values of 0 and 1, respectively.

The normal distribution has density

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

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

Value

dnorm gives the density, pnorm is the cumulative distribution function, and qnorm is the quantile function of the normal distribution. rnorm generates random deviates.

The length of the result is determined by n for rnorm, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

For sd = 0 this gives the limit as sd decreases to 0, a point mass at mu. sd < 0 is an error and returns NaN.

Source

For pnorm, based on Cody (1993).

For qnorm, the code is based on a C translation of Wichura (1988) which provides precise results up to about 16 digits for log.p=FALSE. For log scale probabilities in the extreme tails, since R version 4.1.0, extensively since 4.3.0, asymptotic expansions are used which have been derived and explored in Mächler (2022).

For rnorm, see RNG for how to select the algorithm and for references to the supplied methods.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Cody WJ (1993). "Algorithm 715: SPECFUN—A Portable FORTRAN Package of Special Function Routines and Test Drivers." *ACM Transactions on Mathematical Software*, **19**(1), 22–30. doi:10.1145/151271.151273.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions*, volume 1. Wiley, New York. ISBN 978-0-471-58495-7.
Chapter 13.

Mächler M (2022). *Asymptotic Tail Formulas for Gaussian Quantiles*. Vignette, CRAN package **DPQ**, https://CRAN.R-project.org/package=DPQ/vignettes/qnorm-asymp.pdf.

Wichura MJ (1988). "Algorithm AS 241: The Percentage Points of the Normal Distribution." *Applied Statistics*, **37**(3), 477. doi:10.2307/2347330.

See Also

Distributions for other standard distributions, including `dlnorm` for the *log*-normal distribution.

Examples

```
require(graphics)

dnorm(0) == 1/sqrt(2*pi)
dnorm(1) == exp(-1/2)/sqrt(2*pi)
dnorm(1) == 1/sqrt(2*pi*exp(1))

## Using "log = TRUE" for an extended range :
par(mfrow = c(2,1))
plot(function(x) dnorm(x, log = TRUE), -60, 50,
     main = "log { Normal density }")
curve(log(dnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("dnorm(x, log=TRUE)", adj = 0)
mtext("log(dnorm(x))", col = "red", adj = 1)

plot(function(x) pnorm(x, log.p = TRUE), -50, 10,
     main = "log { Normal Cumulative }")
curve(log(pnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("pnorm(x, log=TRUE)", adj = 0)
mtext("log(pnorm(x))", col = "red", adj = 1)

## if you want the so-called 'error function'
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
## (see Abramowitz and Stegun 29.2.29)
## and the so-called 'complementary error function'
erfc <- function(x) 2 * pnorm(x * sqrt(2), lower = FALSE)
## and the inverses
erfinv <- function (x) qnorm((1 + x)/2)/sqrt(2)
erfcinv <- function (x) qnorm(x/2, lower = FALSE)/sqrt(2)
```

dnull *Null distributions always returns NA*

Description

Null distributions always returns NA

Usage

```
dnull(x, ...)
```

Arguments

x	vector of quantiles
...	passed onto named functions

Examples

```
dnull(c(0.25, 0.5, 0.75), 0.5, 0.25)
```

dwedge *Wedge distribution*

Description

The wedge distribution has a support of 0 to 1 and has a linear probability density function over that support.

Usage

```
dwedge(x, a, lower.tail = TRUE, log = FALSE)
```

Arguments

x	vector of quantiles
a	a gradient from -2 (left skewed) to 2 (right skewed)
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log	logical; if TRUE, probabilities p are given as log(p).

Details

The rwedge can be combined with quantile functions to skew standard distributions, or introduce correlation or down weight certain parts of the distribution.

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
pwedge(seq(0, 1, 0.1), a = 1)
dwedge(seq(0, 1, 0.1), a = 1)
qwedge(c(0.25, 0.5, 0.75), a = -1)

stats::cor(
  stats::qnorm(rwedge(1000, a = 2)),
  stats::qnorm(rwedge(1000, a = -2))
)
```

---

generate_simulation *Generate an set of scenarios for testing scoring*

Description

Scenarios are based on an estimator producing predictions as parametrised distributions, and the estimator can be characterised in terms of the bias and sharpness distributions that govern predictions. Predictions are then expressed as samples from these bias and sharpness distributions and the associated distribution function (which will be parameterised with `mean = bias` and `sd = sharpness`)

Usage

```
generate_simulation(config = i_sim_config, true_value_fn = NULL)
```

Arguments

`config` the simulation configuration - a dataframe with columns:

- samples (integer) - The number of estimators simulated
- link (character + default("identity")) - a transformation that puts the quantity being estimated on a linear scale
- true_value (numeric + default(0)) - the true value being estimated
- estimator_dist (character + default("norm")) - the estimator distribution name (e.g. norm) parameterised by location (true_value+bias) and scale (sharpness)
- bias_dist (character + default("norm")) - the distribution name of the estimator generation error function
- bias_location (numeric + default(0)) - the location of the estimator generation error function, this is the true bias / mu_e
- bias_scale (numeric + default(1)) - the scale of the estimator generation error function, this is the true uncertainty of the estimator / sigma_e
- sharpness_dist (character + default("lnorm4")) - the distribution of the stated uncertainty of the estimator

- sharpness_location (numeric + default(1)) - the location of the stated uncertainty of the estimator / sigma_s
- sharpness_scale (numeric + default(0)) - the scale of the stated uncertainty of the estimator, determines how consistent the stated uncertainty is realtive to the true uncertainty / kappa,

Any grouping allowed.

`true_value_fn` a function that generates the true values e.g. a random sampler (default `` `NULL` `` [NULL])

Value

a

Examples

```
config = dplyr::tibble(samples = 100)
config %>% generate_simulation()
```

`p_l1_bridge` *Distribution of the L1 norm of the Brownian bridge*

Description

The limiting null distribution of $\sqrt{n}\,U^*_{pit}$. For a standard Brownian bridge B, Johnson & Killeen (1983) give the c.d.f. of $L = \int_0^1 |B(t)|\,\mathrm{d}t$ as the convergent series

Usage

```
p_l1_bridge(q, terms = 30L)

q_l1_bridge(p, terms = 30L)
```

Arguments

`q`	a vector of quantiles (non-negative)
`terms`	number of terms of the series to sum (default `` `30` `` [integer])
`p`	a vector of probabilities

Details

$$P(L \le \ell) = \sqrt{\pi/2}\sum_{j=1}^{\infty} \delta_j^{-3/2}\psi(\ell/\delta_j^{3/2})$$

with $\psi(t) = 3^{2/3}t^{-1/3}e^{-2/(27t^2)}Ai((3t)^{-4/3})$ and $\delta_j = |a'_j|/2^{1/3}$, where a'_j is the jth zero of Ai'. Terms are suppressed by $e^{-2/(27t^2)}$ with $t = q/\delta_j^{3/2}$, so the number of terms needed grows with `q`. The default is ample over the whole range where the upper tail is numerically distinguishable from zero; beyond `q = 10` the c.d.f. is returned as exactly 1.

Value

p_l1_bridge() returns the c.d.f. evaluated at q; q_l1_bridge() returns the quantile function evaluated at p

Unit tests

```
# the mean of the L1 norm of the bridge is known exactly to be sqrt(2 pi)/8
m = integrate(function(x) 1 - p_l1_bridge(x), 0, 5, subdivisions = 500L)$value
testthat::expect_equal(m, sqrt(2 * pi) / 8, tolerance = 1e-5)

# a proper distribution function
testthat::expect_equal(p_l1_bridge(0), 0)
testthat::expect_equal(p_l1_bridge(10), 1)
testthat::expect_true(all(diff(p_l1_bridge(seq(0.05, 1, 0.05))) > 0))

# and its inverse
testthat::expect_equal(p_l1_bridge(q_l1_bridge(0.95)), 0.95, tolerance = 1e-6)
```

---

pbeta2 *The Beta Distribution*

Description

Density, distribution function, quantile function and random generation for the Beta distribution with parameters shape1 and shape2 (and optional non-centrality parameter ncp).

Usage

```
pbeta2(q, prob, kappa, lower.tail = TRUE, log.p = FALSE)
```

Arguments

q	vector of quantiles
prob	the mean probability (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Value

dbeta gives the density, pbeta is the cumulative distribution function, and qbeta is the quantile function of the Beta distribution. rbeta generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rbeta, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::pbeta()

Examples

```
pbeta2(c(0.25, 0.5, 0.75), 0.5, 0.25)
```

---

pcgamma *Cumulative probability: gamma distribution constrained to have mean > sd*

Description

The following conversion describes the parameters mean and kappa

Usage

```
pcgamma(q, median, kappa = 1/median, lower.tail = TRUE, log.p = FALSE)
```

Arguments

q	vector of quantiles
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Details

$$\text{shape:}\alpha = \frac{1}{\kappa} \text{rate:}\beta = \frac{1}{\mu \times \kappa} \text{scale:}\sigma = \mu \times \kappa$$

Value

dgamma gives the density, pgamma is the cumulative distribution function, and qgamma is the quantile function of the Gamma distribution. rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::pgamma()

Examples

```
pcgamma(seq(0, 4, 0.25), 2, 0.5)
```

---

pdist *Cumulative probability from a named distribution*

Description

Cumulative probability from a named distribution

Usage

```
pdist(dist, q, ...)
```

Arguments

dist	the distribution name e.g.lnorm
q	vector of quantiles
...	passed onto named functions

---

pgamma2 *The Gamma Distribution*

Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

Usage

```
pgamma2(q, mean, sd = sqrt(mean), lower.tail = TRUE, log.p = FALSE)
```

Arguments

q	vector of quantiles
mean	the mean value on the true scale (vectorised)
sd	the standard deviation on the true scale (vectorised)
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Value

dgamma gives the density, pgamma is the cumulative distribution function, and qgamma is the quantile function of the Gamma distribution. rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::pgamma()

Examples

```
pgamma2(seq(0, 4, 0.25), 2, 1)
```

---

pgamma4 *The Gamma Distribution*

Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

Usage

```
pgamma4(q, median, kappa = 1/median, lower.tail = TRUE, log.p = FALSE)
```

Arguments

q	vector of quantiles
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Value

dgamma gives the density, pgamma is the cumulative distribution function, and qgamma is the quantile function of the Gamma distribution. rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::pgamma()

Examples

```
pgamma2(seq(0, 4, 0.25), 2, 1)
```

---

pit_w *Calculate a pit-wasserstein value for a single estimator*

Description

An estimator produces multiple parametrised probability distributions, from which we can calculate scores such as CPRS and PIT-W

Usage

```
pit_w(
  true_value,
  location,
  scale,
  dist = "norm",
  link = "identity",
  adjust = c("link", "probit")
)
```

Arguments

true_value	The true value
location	The central estimate of the probabilistic prediction
scale	The spread estimate of the probabilisitic prediction
dist	The distribution name of the prediction (default `norm` [character])
link	A link function mapping the value space to a linear one, e.g. "identity", "log" (default `identity` [character])
adjust	adjustment method for constructing the pit value (default `link`; allowed: `link`,`probit`)

Value

a one row data frame with:

- U_pit an undirected pit-wasserstein
- D_pit a directed pit-wasserstein
- B_pit raw unadjusted undrected pit-wasserstein

Unit tests

```
scores = pit_w(0, qnorm(seq(1,199,2)/200, 0, 1), 1, dist = "norm", link="identity")
testthat::expect_equal(scores$U_pit, 0)

scores = pit_w(1, calibrated_locations(qlnorm4, 1, 1), 1, dist = "lnorm4", link="log")
testthat::expect_equal(scores$U_pit, 0)

scores = pit_w(1, calibrated_locations(qlnorm4, 1, 1), 1, dist = "lnorm4", link="log")
testthat::expect_equal(scores$U_pit, 0)

scores = pit_w(1, calibrated_locations(qgamma4, 1, 0.5), 0.5, dist = "gamma4", link="log")
testthat::expect_equal(scores$U_pit, 0, tolerance = 10^-5)
```

`plnorm2` *The Log-Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the log-normal distribution whose logarithm has mean equal to `meanlog` and standard deviation equal to `sdlog`.

Usage

```
plnorm2(q, mean = 1, sd = sqrt(exp(1) - 1), lower.tail = TRUE, log.p = FALSE)
```

Arguments

`q`	vector of quantiles
`mean`	the mean value on the true scale (vectorised)
`sd`	the standard deviation on the true scale (vectorised)
`lower.tail`	logical; if `TRUE` (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
`log.p`	logical; if TRUE, probabilities p are given as log(p).

Details

The log-normal distribution has density

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

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

Value

dlnorm gives the density, plnorm is the cumulative distribution function, and qlnorm is the quantile function of the log-normal distribution. rlnorm generates random deviates.

The length of the result is determined by n for rlnorm, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

Note

The cumulative hazard $H(t) = -\log(1 - F(t))$ is -plnorm(t, r, lower = FALSE, log = TRUE).

Source

dlnorm is calculated from the definition (in 'Details'). [pqr]lnorm are based on the relationship to the normal.

Consequently, they model a single point mass at exp(meanlog) for the boundary case sdlog = 0.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions*, volume 1. Wiley, New York. ISBN 978-0-471-58495-7.
Chapter 14.

See Also

Distributions for other standard distributions, including dnorm for the normal distribution.

Examples

```
plnorm2(seq(0, 4, 0.25), 2, 1)
```

---

plnorm4 *The Log-Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the log-normal distribution whose logarithm has mean equal to meanlog and standard deviation equal to sdlog.

Usage

```
plnorm4(q, median = 1, kappa = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

q	vector of quantiles
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).

Details

The log-normal distribution has density

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

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

Value

dlnorm gives the density, plnorm is the cumulative distribution function, and qlnorm is the quantile function of the log-normal distribution. rlnorm generates random deviates.

The length of the result is determined by n for rlnorm, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

Note

The cumulative hazard $H(t) = -\log(1 - F(t))$ is -plnorm(t, r, lower = FALSE, log = TRUE).

Source

dlnorm is calculated from the definition (in 'Details'). [pqr]lnorm are based on the relationship to the normal.

Consequently, they model a single point mass at exp(meanlog) for the boundary case sdlog = 0.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions*, volume 1. Wiley, New York. ISBN 978-0-471-58495-7.
Chapter 14.

See Also

Distributions for other standard distributions, including dnorm for the normal distribution.

Examples

```
plnorm4(seq(0, 4, 0.25), 2, 1)
```

---

plogitnorm *Logit-normal distribution*

Description

The logit-normal distribution has a support of 0 to 1.

Usage

```
plogitnorm(q, meanlogit = 0, sdlogit = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

q	vector of quantiles (0<q<1)
meanlogit	the mean on the logit scale
sdlogit	the sd on the logit scale
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
plogitnorm(seq(0.1, 0.9, 0.1), 0, 1)
```

plogitnorm2 *Logit-normal distribution*

Description

The logit-normal distribution has a support of 0 to 1.

Usage

```
plogitnorm2(
  q,
  prob.0.5 = 0.5,
  kappa = 1 - exp(-1),
  lower.tail = TRUE,
  log.p = FALSE
)
```

Arguments

q	vector of quantiles (0<q<1)
prob.0.5	the median on the true scale
kappa	a dispersion parameter from 0 (none) to 1 maximum dispersion
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
plogitnorm2(seq(0.1, 0.9, 0.1), 0.75, 0.2)
plogitnorm2(qlogitnorm2(seq(0.1, 0.9, 0.1), 0.75, 0.2), 0.75, 0.2)
```

pnbinom2 *The Negative Binomial Distribution*

Description

Density, distribution function, quantile function and random generation for the negative binomial distribution with parameters size and prob.

Usage

```
pnbinom2(q, mean, sd = sqrt(mean), lower.tail = TRUE, log.p = FALSE)
```

Arguments

q	vector of quantiles.
mean	the mean value on the true scale (vectorised)
sd	the standard deviation on the true scale (vectorised)
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).

Details

The negative binomial distribution with size $= n$ and prob $= p$ has density

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

for $x = 0, 1, 2, \ldots$, $n > 0$ and $0 < p \leq 1$.

This represents the number of failures which occur in a sequence of Bernoulli trials before a target number of successes is reached. The mean is $\mu = n(1-p)/p$ and variance $n(1-p)/p^2$.

A negative binomial distribution can also arise as a mixture of Poisson distributions with mean distributed as a gamma distribution (see pgamma) with scale parameter (1 - prob)/prob and shape parameter size. (This definition allows non-integer values of size.)

An alternative parametrization (often used in ecology) is by the *mean* mu (see above), and size, the *dispersion parameter*, where prob = size/(size+mu). The variance is mu + mu^2/size in this parametrization.

If an element of x is not integer, the result of dnbinom is zero, with a warning.

The case size == 0 is the distribution concentrated at zero. This is the limiting distribution for size approaching zero, even if mu rather than prob is held constant. Notice though, that the mean of the limit distribution is 0, whatever the value of mu.

The quantile is defined as the smallest value x such that $F(x) \geq p$, where F is the distribution function.

Value

dnbinom gives the density, pnbinom is the cumulative distribution function, and qnbinom is the quantile function of the negative binomial distribution. rnbinom generates random deviates.

Invalid size or prob will result in return value NaN, with a warning.

The length of the result is determined by n for rnbinom, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

rnbinom returns a vector of type integer unless generated values exceed the maximum representable integer when double values are returned.

Source

dnbinom computes via binomial probabilities, using code contributed by Catherine Loader (see dbinom).

pnbinom uses pbeta.

qnbinom uses the Cornish–Fisher Expansion to include a skewness correction to a normal approximation, followed by a search.

rnbinom uses the derivation as a gamma mixture of Poisson distributions, see Devroye (1986, page 480).

References

Devroye L (1986). *Non-Uniform Random Variate Generation*. Springer, New York. doi:10.1007/9781461386438. https://luc.devroye.org/handbooksimulation1.pdf.

See Also

Distributions for standard distributions, including dbinom for the binomial, dpois for the Poisson and dgeom for the geometric distribution, which is a special case of the negative binomial.

Examples

```
pnbinom2(0:5, 2, sqrt(2))
```

---

pnorm4 *The Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to mean and standard deviation equal to sd.

Usage

```
pnorm4(q, median = 0, kappa = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

q	vector of quantiles
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).

Details

If mean or sd are not specified they assume the default values of 0 and 1, respectively.

The normal distribution has density

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

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

Value

dnorm gives the density, pnorm is the cumulative distribution function, and qnorm is the quantile function of the normal distribution. rnorm generates random deviates.

The length of the result is determined by n for rnorm, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

For sd = 0 this gives the limit as sd decreases to 0, a point mass at mu. sd < 0 is an error and returns NaN.

Source

For pnorm, based on Cody (1993).

For qnorm, the code is based on a C translation of Wichura (1988) which provides precise results up to about 16 digits for log.p=FALSE. For log scale probabilities in the extreme tails, since R version 4.1.0, extensively since 4.3.0, asymptotic expansions are used which have been derived and explored in Mächler (2022).

For rnorm, see RNG for how to select the algorithm and for references to the supplied methods.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Cody WJ (1993). "Algorithm 715: SPECFUN—A Portable FORTRAN Package of Special Function Routines and Test Drivers." *ACM Transactions on Mathematical Software*, **19**(1), 22–30. doi:10.1145/151271.151273.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions*, volume 1. Wiley, New York. ISBN 978-0-471-58495-7.
Chapter 13.

Mächler M (2022). *Asymptotic Tail Formulas for Gaussian Quantiles*. Vignette, CRAN package **DPQ**, https://CRAN.R-project.org/package=DPQ/vignettes/qnorm-asymp.pdf.

Wichura MJ (1988). "Algorithm AS 241: The Percentage Points of the Normal Distribution." *Applied Statistics*, **37**(3), 477. doi:10.2307/2347330.

See Also

Distributions for other standard distributions, including dlnorm for the *log*-normal distribution.

Examples

```
require(graphics)

dnorm(0) == 1/sqrt(2*pi)
dnorm(1) == exp(-1/2)/sqrt(2*pi)
dnorm(1) == 1/sqrt(2*pi*exp(1))

## Using "log = TRUE" for an extended range :
par(mfrow = c(2,1))
plot(function(x) dnorm(x, log = TRUE), -60, 50,
     main = "log { Normal density }")
curve(log(dnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("dnorm(x, log=TRUE)", adj = 0)
mtext("log(dnorm(x))", col = "red", adj = 1)

plot(function(x) pnorm(x, log.p = TRUE), -50, 10,
     main = "log { Normal Cumulative }")
curve(log(pnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("pnorm(x, log=TRUE)", adj = 0)
mtext("log(pnorm(x))", col = "red", adj = 1)

## if you want the so-called 'error function'
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
## (see Abramowitz and Stegun 29.2.29)
## and the so-called 'complementary error function'
erfc <- function(x) 2 * pnorm(x * sqrt(2), lower = FALSE)
## and the inverses
erfinv <- function (x) qnorm((1 + x)/2)/sqrt(2)
erfcinv <- function (x) qnorm(x/2, lower = FALSE)/sqrt(2)
```

`pnull` *Null distributions always returns NA*

Description

Null distributions always returns NA

Usage

```
pnull(q, ...)
```

Arguments

`q`	vector of quantiles
`...`	passed onto named functions

Examples

```
pnull(c(0.25, 0.5, 0.75), 0.5, 0.25)
```

`prediction_scores` *Score a parametrised probability distribution*

Description

Create a basket of scores for a set of paramtrised probabilistic predictions compared to a known ground truth, predictions are assumed to be distributional and associated with a link function that transforms them to linear space.

Usage

```
prediction_scores(df = i_comparison, summarise = TRUE, threshold = NA_real_)
```

Arguments

`df`	input dataframe - a dataframe with columns: • true_value (numeric) - the true value • link (character + default("identity")) - a transformation that puts the quantity being estimated on a linear scale • location (numeric) - the location parameter is the estimator mean or median (i.e. the central estimate) • scale (numeric) - the scale parameter is the stated estimator uncertainty (i.e. the sharpness) • dist (character + default("norm")) - the estimator distribution name (e.g. norm) parameterised by location and scale • calibration (numeric) - the calibration as determined by the SD of the probabilistic prediction (sharpness) / the SD of the true values distribution, overconfident is <1, and conservative is >1 • adjust (character + default("link")) - method for adjusting for bias. Either link function adjustment of true values, or probit adjustment of PIT values Minimally grouped by: dist + link + calibration + adjust (and other groupings allowed).
`summarise`	produce the results as a groupwise summary or a prediction by prediction metric (default `` `TRUE` `` [logical])
`threshold`	If given calculates the threshold misclassification probability (default `` `NA` `` [numeric])

Value

a dataframe. columns depend on `summarise`

Examples

```
qs = seq(1, 199, 2) / 200
df = tidyr::crossing(
  true_value = 0,
  location = stats::qnorm(qs, 1, 1),
  scale = c(1 / 2, 1 / 1.5, 1, 1.5, 2),
  dist = "norm",
  link = "identity"
) %>%
  dplyr::mutate(calibration = scale) %>%
  dplyr::group_by(dist, link, calibration)
df %>%
  prediction_scores() %>%
  dplyr::glimpse()
```

---

pwedge *Wedge distribution*

Description

The wedge distribution has a support of 0 to 1 and has a linear probability density function over that support.

Usage

```
pwedge(q, a, log.p = FALSE)
```

Arguments

q	vector of quantiles
a	a gradient from -2 (left skewed) to 2 (right skewed)
log.p	logical; if TRUE, probabilities p are given as log(p).

Details

The rwedge can be combined with quantile functions to skew standard distributions, or introduce correlation or down weight certain parts of the distribution.

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
pwedge(seq(0, 1, 0.1), a = 1)
dwedge(seq(0, 1, 0.1), a = 1)
qwedge(c(0.25, 0.5, 0.75), a = -1)

stats::cor(
  stats::qnorm(rwedge(1000, a = 2)),
  stats::qnorm(rwedge(1000, a = -2))
)
```

---

qbeta2 *The Beta Distribution*

Description

Density, distribution function, quantile function and random generation for the Beta distribution with parameters shape1 and shape2 (and optional non-centrality parameter ncp).

Usage

```
qbeta2(p, prob, kappa, lower.tail = TRUE, log.p = FALSE)
```

Arguments

p	vector of probabilities
prob	the mean probability (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Value

dbeta gives the density, pbeta is the cumulative distribution function, and qbeta is the quantile function of the Beta distribution. rbeta generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rbeta, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::qbeta()

Examples

```
qbeta2(c(0.25, 0.5, 0.75), 0.5, 0.25)
```

qcgamma *Quantile: gamma distribution constrained to have mean > sd*

Description

The following conversion describes the parameters mean and kappa

Usage

```
qcgamma(p, median, kappa = 1/median, lower.tail = TRUE, log.p = FALSE)
```

Arguments

p	vector of probabilities
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Details

$$\text{shape:}\alpha = \frac{1}{\kappa} \text{rate:}\beta = \frac{1}{\mu \times \kappa} \text{scale:}\sigma = \mu \times \kappa$$

Value

dgamma gives the density, pgamma is the cumulative distribution function, and qgamma is the quantile function of the Gamma distribution. rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::qgamma()

Examples

```
qcgamma(c(0.25, 0.5, 0.75), 2, 0.5)
```

qdist *Quantile function of a named distribution*

Description

Quantile function of a named distribution

Usage

```
qdist(dist, p, ...)
```

Arguments

dist	the distribution name e.g.lnorm
p	vector of probabilities
...	passed onto named functions

qgamma2 *The Gamma Distribution*

Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

Usage

```
qgamma2(p, mean, sd = sqrt(mean), lower.tail = TRUE, log.p = FALSE)
```

Arguments

p	vector of probabilities
mean	the mean value on the true scale (vectorised)
sd	the standard deviation on the true scale (vectorised)
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Value

dgamma gives the density, pgamma is the cumulative distribution function, and qgamma is the quantile function of the Gamma distribution. rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::qgamma()

Examples

```
qgamma2(c(0.25, 0.5, 0.75), 2, 1)
```

---

qgamma4 *The Gamma Distribution*

Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

Usage

```
qgamma4(p, median, kappa = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

p	vector of probabilities
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Value

dgamma gives the density, pgamma is the cumulative distribution function, and qgamma is the quantile function of the Gamma distribution. rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::qgamma()

Examples

```
qgamma2(c(0.25, 0.5, 0.75), 2, 1)
```

qlnorm2 *The Log-Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the log-normal distribution whose logarithm has mean equal to `meanlog` and standard deviation equal to `sdlog`.

Usage

```
qlnorm2(p, mean = 1, sd = sqrt(exp(1) - 1), lower.tail = TRUE, log.p = FALSE)
```

Arguments

`p`	vector of probabilities.
`mean`	the mean value on the true scale (vectorised)
`sd`	the standard deviation on the true scale (vectorised)
`lower.tail`	logical; if `TRUE` (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
`log.p`	logical; if TRUE, probabilities p are given as log(p).

Details

The log-normal distribution has density

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

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

Value

`dlnorm` gives the density, `plnorm` is the cumulative distribution function, and `qlnorm` is the quantile function of the log-normal distribution. `rlnorm` generates random deviates.

The length of the result is determined by `n` for `rlnorm`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

Note

The cumulative hazard $H(t) = -\log(1 - F(t))$ is `-plnorm(t, r, lower = FALSE, log = TRUE)`.

Source

dlnorm is calculated from the definition (in 'Details'). [pqr]lnorm are based on the relationship to the normal.

Consequently, they model a single point mass at exp(meanlog) for the boundary case sdlog = 0.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions*, volume 1. Wiley, New York. ISBN 978-0-471-58495-7.
Chapter 14.

See Also

Distributions for other standard distributions, including dnorm for the normal distribution.

Examples

```
qlnorm2(c(0.25, 0.5, 0.72), 2, 1)
```

---

qlnorm4 *The Log-Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the log-normal distribution whose logarithm has mean equal to meanlog and standard deviation equal to sdlog.

Usage

```
qlnorm4(p, median = 1, kappa = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

p	vector of probabilities.
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).

Details

The log-normal distribution has density

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

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

Value

`dlnorm` gives the density, `plnorm` is the cumulative distribution function, and `qlnorm` is the quantile function of the log-normal distribution. `rlnorm` generates random deviates.

The length of the result is determined by `n` for `rlnorm`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

Note

The cumulative hazard $H(t) = -\log(1 - F(t))$ is `-plnorm(t, r, lower = FALSE, log = TRUE)`.

Source

`dlnorm` is calculated from the definition (in 'Details'). `[pqr]lnorm` are based on the relationship to the normal.

Consequently, they model a single point mass at `exp(meanlog)` for the boundary case `sdlog = 0`.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions*, volume 1. Wiley, New York. ISBN 978-0-471-58495-7.
Chapter 14.

See Also

Distributions for other standard distributions, including `dnorm` for the normal distribution.

Examples

```
qlnorm4(c(0.25, 0.5, 0.72), 2, 1)
```

qlogitnorm *Logit-normal distribution*

Description

The logit-normal distribution has a support of 0 to 1.

Usage

```
qlogitnorm(p, meanlogit = 0, sdlogit = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

p	vector of probabilities
meanlogit	the mean on the logit scale
sdlogit	the sd on the logit scale
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
qlogitnorm(c(0.25, 0.5, 0.75), 0, 1)
```

qlogitnorm2 *Logit-normal distribution*

Description

The logit-normal distribution has a support of 0 to 1.

Usage

```
qlogitnorm2(
  p,
  prob.0.5 = 0.5,
  kappa = 1 - exp(-1),
  lower.tail = TRUE,
  log.p = FALSE
)
```

Arguments

p	vector of probabilities
prob.0.5	the median on the true scale
kappa	a dispersion parameter from 0 (none) to 1 maximum dispersion
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
qlnorm2(c(0.25, 0.5, 0.72), 2, 1)
```

---

qnbinom2 *The Negative Binomial Distribution*

Description

Density, distribution function, quantile function and random generation for the negative binomial distribution with parameters size and prob.

Usage

```
qnbinom2(p, mean, sd = sqrt(mean), lower.tail = TRUE, log.p = FALSE)
```

Arguments

p	vector of probabilities.
mean	the mean value on the true scale (vectorised)
sd	the standard deviation on the true scale (vectorised)
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).

Details

The negative binomial distribution with size $= n$ and prob $= p$ has density

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

for $x = 0, 1, 2, \ldots$, $n > 0$ and $0 < p \leq 1$.

This represents the number of failures which occur in a sequence of Bernoulli trials before a target number of successes is reached. The mean is $\mu = n(1-p)/p$ and variance $n(1-p)/p^2$.

A negative binomial distribution can also arise as a mixture of Poisson distributions with mean distributed as a gamma distribution (see pgamma) with scale parameter (1 - prob)/prob and shape parameter size. (This definition allows non-integer values of size.)

An alternative parametrization (often used in ecology) is by the *mean* mu (see above), and size, the *dispersion parameter*, where prob = size/(size+mu). The variance is mu + mu^2/size in this parametrization.

If an element of x is not integer, the result of dnbinom is zero, with a warning.

The case size == 0 is the distribution concentrated at zero. This is the limiting distribution for size approaching zero, even if mu rather than prob is held constant. Notice though, that the mean of the limit distribution is 0, whatever the value of mu.

The quantile is defined as the smallest value x such that $F(x) \geq p$, where F is the distribution function.

Value

dnbinom gives the density, pnbinom is the cumulative distribution function, and qnbinom is the quantile function of the negative binomial distribution. rnbinom generates random deviates.

Invalid size or prob will result in return value NaN, with a warning.

The length of the result is determined by n for rnbinom, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

rnbinom returns a vector of type integer unless generated values exceed the maximum representable integer when double values are returned.

Source

dnbinom computes via binomial probabilities, using code contributed by Catherine Loader (see dbinom).

pnbinom uses pbeta.

qnbinom uses the Cornish–Fisher Expansion to include a skewness correction to a normal approximation, followed by a search.

rnbinom uses the derivation as a gamma mixture of Poisson distributions, see Devroye (1986, page 480).

References

Devroye L (1986). *Non-Uniform Random Variate Generation*. Springer, New York. doi:10.1007/9781461386438. https://luc.devroye.org/handbooksimulation1.pdf.

See Also

Distributions for standard distributions, including dbinom for the binomial, dpois for the Poisson and dgeom for the geometric distribution, which is a special case of the negative binomial.

Examples

```
qnbinom2(c(0.25, 0.5, 0.75), 5, sqrt(5))
```

---

qnorm4 *The Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to mean and standard deviation equal to sd.

Usage

```
qnorm4(p, median = 0, kappa = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

p	vector of probabilities.
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.
log.p	logical; if TRUE, probabilities p are given as log(p).

Details

If mean or sd are not specified they assume the default values of 0 and 1, respectively.

The normal distribution has density

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

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

Value

dnorm gives the density, pnorm is the cumulative distribution function, and qnorm is the quantile function of the normal distribution. rnorm generates random deviates.

The length of the result is determined by n for rnorm, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

For sd = 0 this gives the limit as sd decreases to 0, a point mass at mu. sd < 0 is an error and returns NaN.

Source

For pnorm, based on Cody (1993).

For qnorm, the code is based on a C translation of Wichura (1988) which provides precise results up to about 16 digits for log.p=FALSE. For log scale probabilities in the extreme tails, since R version 4.1.0, extensively since 4.3.0, asymptotic expansions are used which have been derived and explored in Mächler (2022).

For rnorm, see RNG for how to select the algorithm and for references to the supplied methods.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Cody WJ (1993). "Algorithm 715: SPECFUN—A Portable FORTRAN Package of Special Function Routines and Test Drivers." *ACM Transactions on Mathematical Software*, **19**(1), 22–30. doi:10.1145/151271.151273.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions*, volume 1. Wiley, New York. ISBN 978-0-471-58495-7.
Chapter 13.

Mächler M (2022). *Asymptotic Tail Formulas for Gaussian Quantiles*. Vignette, CRAN package **DPQ**, https://CRAN.R-project.org/package=DPQ/vignettes/qnorm-asymp.pdf.

Wichura MJ (1988). "Algorithm AS 241: The Percentage Points of the Normal Distribution." *Applied Statistics*, **37**(3), 477. doi:10.2307/2347330.

See Also

Distributions for other standard distributions, including dlnorm for the *log*-normal distribution.

Examples

```
require(graphics)

dnorm(0) == 1/sqrt(2*pi)
dnorm(1) == exp(-1/2)/sqrt(2*pi)
dnorm(1) == 1/sqrt(2*pi*exp(1))

## Using "log = TRUE" for an extended range :
par(mfrow = c(2,1))
plot(function(x) dnorm(x, log = TRUE), -60, 50,
     main = "log { Normal density }")
curve(log(dnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("dnorm(x, log=TRUE)", adj = 0)
mtext("log(dnorm(x))", col = "red", adj = 1)

plot(function(x) pnorm(x, log.p = TRUE), -50, 10,
     main = "log { Normal Cumulative }")
curve(log(pnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("pnorm(x, log=TRUE)", adj = 0)
mtext("log(pnorm(x))", col = "red", adj = 1)
```

```
## if you want the so-called 'error function'
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
## (see Abramowitz and Stegun 29.2.29)
## and the so-called 'complementary error function'
erfc <- function(x) 2 * pnorm(x * sqrt(2), lower = FALSE)
## and the inverses
erfinv <- function (x) qnorm((1 + x)/2)/sqrt(2)
erfcinv <- function (x) qnorm(x/2, lower = FALSE)/sqrt(2)
```

---

qnull *Null distributions always returns NA*

Description

Null distributions always returns NA

Usage

```
qnull(p, ...)
```

Arguments

p	vector of probabilities
...	passed onto named functions

Examples

```
qnull(c(0.25, 0.5, 0.75), 0.5, 0.25)
```

---

qwedge *Wedge distribution*

Description

The wedge distribution has a support of 0 to 1 and has a linear probability density function over that support.

Usage

```
qwedge(p, a, lower.tail = TRUE, log.p = FALSE)
```

Arguments

p	vector of probabilities
a	a gradient from -2 (left skewed) to 2 (right skewed)
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
log.p	logical; if TRUE, probabilities p are given as log(p).

Details

The rwedge can be combined with quantile functions to skew standard distributions, or introduce correlation or down weight certain parts of the distribution.

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
pwedge(seq(0, 1, 0.1), a = 1)
dwedge(seq(0, 1, 0.1), a = 1)
qwedge(c(0.25, 0.5, 0.75), a = -1)

stats::cor(
  stats::qnorm(rwedge(1000, a = 2)),
  stats::qnorm(rwedge(1000, a = -2))
)
```

---

rbern *A random Bernoulli sample as a logical value*

Description

A random Bernoulli sample as a logical value

Usage

```
rbern(n, prob)
```

Arguments

n	number of observations
prob	the mean probability (vectorised)

Value

a vector of logical values of size n

Examples

```
table(rbern(100, 0.25))
```

rbeta2 *The Beta Distribution*

Description

Density, distribution function, quantile function and random generation for the Beta distribution with parameters shape1 and shape2 (and optional non-centrality parameter ncp).

Usage

```
rbeta2(n, prob, kappa)
```

Arguments

n	number of observations
prob	the mean probability (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.

Value

dbeta gives the density, pbeta is the cumulative distribution function, and qbeta is the quantile function of the Beta distribution. rbeta generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rbeta, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::rbeta()

Examples

```
rbeta2(3, c(0.1, 0.5, 0.9), 0.1)
```

rcategorical *Sampling from the multinomial equivalent of the Bernoulli distribution*

Description

Sampling from the multinomial equivalent of the Bernoulli distribution

Usage

```
rcategorical(n, prob, factor = FALSE)
```

Arguments

n	a sample size
prob	a (optionally named) vector of probabilities that will be normalised to sum to 1
factor	if not FALSE then factor levels will either be taken from the names of prob first, or if this is a character vector from this. (default `FALSE` [logical])

Value

a vector of random class labels of length n. Labels come from names of prob or from a character vector in factor.

Examples

```
prob = c("one" = 0.1, "two" = 0.2, "seven" = 0.7)
table(rcategorical(1000, prob))
rcategorical(10, prob, factor = TRUE)
rcategorical(10, rep(1, 26), factor = letters)
```

rcgamma *Sampling: gamma distribution constrained to have median > sd*

Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters shape and scale.

Usage

```
rcgamma(n, median, kappa = 1/median)
```

Arguments

n	number of observations
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.

Value

dgamma gives the density, pgamma is the cumulative distribution function, and qgamma is the quantile function of the Gamma distribution. rgamma generates random deviates.

Invalid arguments will result in return value NaN, with a warning.

The length of the result is determined by n for rgamma, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

stats::rgamma()

Examples

```
rcgamma(10, 2, 0.5)
```

---

rdist *Random sample from a named distribution*

Description

Random sample from a named distribution

Usage

```
rdist(dist, n, ...)
```

Arguments

dist	the distribution name e.g.lnorm
n	number of observations
...	passed onto named functions

reparam_fn *Get a converter that maps an unusual parametrisation to a canonical one.*

Description

Get a converter that maps an unusual parametrisation to a canonical one.

Usage

```
reparam_fn(dist_name)
```

Arguments

dist_name	the distribution family as a string, e.g. `lnorm4`

Value

a function that takes 2 parameters of the original distrbition and returns a list with the parameters and name of the canonical distribution.

Examples

```
tmp_fn = reparam_fn("lnorm2")
tmp_fn(mean = 1, sd = 2)
```

rexpgrowth *Randomly sample incident times in an exponentially growing process*

Description

Randomly sample incident times in an exponentially growing process

Usage

```
rexpgrowth(n, r, t_end, t_start = 0)
```

Arguments

n	the number of items to sample
r	and exponential growth rate (per unit time)
t_end	the end of the observation period
t_start	the start of the observation period (default `0` [numeric])

Value

a vector of n samples from an exponential growth process

Examples

```
graphics::hist(rexpgrowth(1000, 0.1, 40), breaks = 40)
graphics::hist(rexpgrowth(1000, -0.1, 40), breaks = 40)
```

---

rexpgrowthI0 *Randomly sample incident times in an exponentially growing process with initial case load*

Description

Randomly sample incident times in an exponentially growing process with initial case load

Usage

```
rexpgrowthI0(I0, r, t_end, t_start = 0)
```

Arguments

I0	the expected number of cases observed in the first day
r	an exponential growth rate (per unit time)
t_end	the end of the observation period
t_start	the start of the observation period (default `0` [numeric])

Value

a vector of n samples from an exponential growth process

Examples

```
graphics::hist(rexpgrowthI0(10, 0.1, 20), breaks = 40)
graphics::hist(rexpgrowthI0(1000, -0.1, 40), breaks = 40)
```

`rgamma2` *The Gamma Distribution*

Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters `shape` and `scale`.

Usage

```
rgamma2(n, mean, sd = sqrt(mean))
```

Arguments

`n`	number of observations
`mean`	the mean value on the true scale (vectorised)
`sd`	the standard deviation on the true scale (vectorised)

Value

`dgamma` gives the density, `pgamma` is the cumulative distribution function, and `qgamma` is the quantile function of the Gamma distribution. `rgamma` generates random deviates.

Invalid arguments will result in return value `NaN`, with a warning.

The length of the result is determined by `n` for `rgamma`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

`stats::rgamma()`

Examples

```
rgamma2(10, 2, 1)
```

`rgamma4` *The Gamma Distribution*

Description

Density, distribution function, quantile function and random generation for the Gamma distribution with parameters `shape` and `scale`.

Usage

```
rgamma4(n, median, kappa = 1/median)
```

Arguments

`n`	number of observations
`median`	the median value on the true scale (vectorised)
`kappa`	a coefficient of variation (sd/mean). where 0 is no variability.

Value

`dgamma` gives the density, `pgamma` is the cumulative distribution function, and `qgamma` is the quantile function of the Gamma distribution. `rgamma` generates random deviates.

Invalid arguments will result in return value `NaN`, with a warning.

The length of the result is determined by `n` for `rgamma`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

See Also

`stats::rgamma()`

Examples

```
rgamma4(10, 2, 1)
```

`rlnorm2` *The Log-Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the log-normal distribution whose logarithm has mean equal to `meanlog` and standard deviation equal to `sdlog`.

Usage

```
rlnorm2(n, mean = 1, sd = sqrt(exp(1) - 1))
```

Arguments

`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`mean`	the mean value on the true scale (vectorised)
`sd`	the standard deviation on the true scale (vectorised)

Details

The log-normal distribution has density

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

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

Value

`dlnorm` gives the density, `plnorm` is the cumulative distribution function, and `qlnorm` is the quantile function of the log-normal distribution. `rlnorm` generates random deviates.

The length of the result is determined by `n` for `rlnorm`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

Note

The cumulative hazard $H(t) = -\log(1 - F(t))$ is `-plnorm(t, r, lower = FALSE, log = TRUE)`.

Source

`dlnorm` is calculated from the definition (in 'Details'). `[pqr]lnorm` are based on the relationship to the normal.

Consequently, they model a single point mass at `exp(meanlog)` for the boundary case `sdlog = 0`.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions*, volume 1. Wiley, New York. ISBN 978-0-471-58495-7.
Chapter 14.

See Also

Distributions for other standard distributions, including `dnorm` for the normal distribution.

Examples

```
rlnorm2(10, 2, 1)
```

`rlnorm4` *The Log-Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the log-normal distribution whose logarithm has mean equal to `meanlog` and standard deviation equal to `sdlog`.

Usage

```
rlnorm4(n, median = 1, kappa = 1)
```

Arguments

`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`median`	the median value on the true scale (vectorised)
`kappa`	a coefficient of variation (sd/mean). where 0 is no variability.

Details

The log-normal distribution has density

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

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

Value

`dlnorm` gives the density, `plnorm` is the cumulative distribution function, and `qlnorm` is the quantile function of the log-normal distribution. `rlnorm` generates random deviates.

The length of the result is determined by `n` for `rlnorm`, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than `n` are recycled to the length of the result. Only the first elements of the logical arguments are used.

Note

The cumulative hazard $H(t) = -\log(1 - F(t))$ is `-plnorm(t, r, lower = FALSE, log = TRUE)`.

Source

`dlnorm` is calculated from the definition (in 'Details'). `[pqr]lnorm` are based on the relationship to the normal.

Consequently, they model a single point mass at `exp(meanlog)` for the boundary case `sdlog = 0`.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions*, volume 1. Wiley, New York. ISBN 978-0-471-58495-7.
Chapter 14.

See Also

Distributions for other standard distributions, including `dnorm` for the normal distribution.

Examples

```
rlnorm4(10, 2, 1)
```

rlogitnorm *Logit-normal distribution*

Description

The logit-normal distribution has a support of 0 to 1.

Usage

```
rlogitnorm(n, meanlogit = 0, sdlogit = 1)
```

Arguments

n	number of observations
meanlogit	the mean on the logit scale
sdlogit	the sd on the logit scale

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
rlogitnorm(10, 0, 1)
```

rlogitnorm2 *Logit-normal distribution*

Description

The logit-normal distribution has a support of 0 to 1.

Usage

```
rlogitnorm2(n, prob.0.5 = 0.5, kappa = 1 - exp(-1))
```

Arguments

n	number of observations
prob.0.5	the median on the true scale
kappa	a dispersion parameter from 0 (none) to 1 maximum dispersion

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
mean(rlogitnorm2(10000, 0.75, 0.2))
```

---

rnbinom2 *The Negative Binomial Distribution*

Description

Density, distribution function, quantile function and random generation for the negative binomial distribution with parameters `size` and `prob`.

Usage

```
rnbinom2(n, mean, sd = sqrt(mean))
```

Arguments

`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`mean`	the mean value on the true scale (vectorised)
`sd`	the standard deviation on the true scale (vectorised)

Details

The negative binomial distribution with `size` $= n$ and `prob` $= p$ has density

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

for $x = 0, 1, 2, \ldots$, $n > 0$ and $0 < p \leq 1$.

This represents the number of failures which occur in a sequence of Bernoulli trials before a target number of successes is reached. The mean is $\mu = n(1-p)/p$ and variance $n(1-p)/p^2$.

A negative binomial distribution can also arise as a mixture of Poisson distributions with mean distributed as a gamma distribution (see pgamma) with scale parameter `(1 - prob)/prob` and shape parameter `size`. (This definition allows non-integer values of `size`.)

An alternative parametrization (often used in ecology) is by the *mean* `mu` (see above), and `size`, the *dispersion parameter*, where `prob = size/(size+mu)`. The variance is `mu + mu^2/size` in this parametrization.

If an element of `x` is not integer, the result of `dnbinom` is zero, with a warning.

The case `size == 0` is the distribution concentrated at zero. This is the limiting distribution for `size` approaching zero, even if `mu` rather than `prob` is held constant. Notice though, that the mean of the limit distribution is 0, whatever the value of `mu`.

The quantile is defined as the smallest value x such that $F(x) \geq p$, where F is the distribution function.

Value

dnbinom gives the density, pnbinom is the cumulative distribution function, and qnbinom is the quantile function of the negative binomial distribution. rnbinom generates random deviates.

Invalid size or prob will result in return value NaN, with a warning.

The length of the result is determined by n for rnbinom, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

rnbinom returns a vector of type integer unless generated values exceed the maximum representable integer when double values are returned.

Source

dnbinom computes via binomial probabilities, using code contributed by Catherine Loader (see dbinom).

pnbinom uses pbeta.

qnbinom uses the Cornish–Fisher Expansion to include a skewness correction to a normal approximation, followed by a search.

rnbinom uses the derivation as a gamma mixture of Poisson distributions, see Devroye (1986, page 480).

References

Devroye L (1986). *Non-Uniform Random Variate Generation*. Springer, New York. doi:10.1007/9781461386438. https://luc.devroye.org/handbooksimulation1.pdf.

See Also

Distributions for standard distributions, including dbinom for the binomial, dpois for the Poisson and dgeom for the geometric distribution, which is a special case of the negative binomial.

Examples

```
rnbinom2(10, 5, sqrt(5))
```

---

rnorm4 *The Normal Distribution*

Description

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to mean and standard deviation equal to sd.

Usage

```
rnorm4(n, median = 0, kappa = 1)
```

Arguments

n	number of observations. If length(n) > 1, the length is taken to be the number required.
median	the median value on the true scale (vectorised)
kappa	a coefficient of variation (sd/mean). where 0 is no variability.

Details

If mean or sd are not specified they assume the default values of 0 and 1, respectively.

The normal distribution has density

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

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

Value

dnorm gives the density, pnorm is the cumulative distribution function, and qnorm is the quantile function of the normal distribution. rnorm generates random deviates.

The length of the result is determined by n for rnorm, and is the maximum of the lengths of the numerical arguments for the other functions.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

For sd = 0 this gives the limit as sd decreases to 0, a point mass at mu. sd < 0 is an error and returns NaN.

Source

For pnorm, based on Cody (1993).

For qnorm, the code is based on a C translation of Wichura (1988) which provides precise results up to about 16 digits for log.p=FALSE. For log scale probabilities in the extreme tails, since R version 4.1.0, extensively since 4.3.0, asymptotic expansions are used which have been derived and explored in Mächler (2022).

For rnorm, see RNG for how to select the algorithm and for references to the supplied methods.

References

Becker RA, Chambers JM, Wilks AR (1988). *The New S Language*. Chapman and Hall/CRC, London. ISBN 053409192X.

Cody WJ (1993). "Algorithm 715: SPECFUN—A Portable FORTRAN Package of Special Function Routines and Test Drivers." *ACM Transactions on Mathematical Software*, **19**(1), 22–30. doi:10.1145/151271.151273.

Johnson NL, Kotz S, Balakrishnan N (1994). *Continuous Univariate Distributions*, volume 1. Wiley, New York. ISBN 978-0-471-58495-7.
Chapter 13.

Mächler M (2022). *Asymptotic Tail Formulas for Gaussian Quantiles*. Vignette, CRAN package **DPQ**, https://CRAN.R-project.org/package=DPQ/vignettes/qnorm-asymp.pdf.

Wichura MJ (1988). "Algorithm AS 241: The Percentage Points of the Normal Distribution." *Applied Statistics*, **37**(3), 477. doi:10.2307/2347330.

See Also

Distributions for other standard distributions, including `dlnorm` for the *log*-normal distribution.

Examples

```
require(graphics)

dnorm(0) == 1/sqrt(2*pi)
dnorm(1) == exp(-1/2)/sqrt(2*pi)
dnorm(1) == 1/sqrt(2*pi*exp(1))

## Using "log = TRUE" for an extended range :
par(mfrow = c(2,1))
plot(function(x) dnorm(x, log = TRUE), -60, 50,
     main = "log { Normal density }")
curve(log(dnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("dnorm(x, log=TRUE)", adj = 0)
mtext("log(dnorm(x))", col = "red", adj = 1)

plot(function(x) pnorm(x, log.p = TRUE), -50, 10,
     main = "log { Normal Cumulative }")
curve(log(pnorm(x)), add = TRUE, col = "red", lwd = 2)
mtext("pnorm(x, log=TRUE)", adj = 0)
mtext("log(pnorm(x))", col = "red", adj = 1)

## if you want the so-called 'error function'
erf <- function(x) 2 * pnorm(x * sqrt(2)) - 1
## (see Abramowitz and Stegun 29.2.29)
## and the so-called 'complementary error function'
erfc <- function(x) 2 * pnorm(x * sqrt(2), lower = FALSE)
## and the inverses
erfinv <- function (x) qnorm((1 + x)/2)/sqrt(2)
erfcinv <- function (x) qnorm(x/2, lower = FALSE)/sqrt(2)
```

`rnull` *Null distributions always returns NA*

Description

Null distributions always returns NA

Usage

```
rnull(n, ...)
```

Arguments

n	number of observations
...	passed onto named functions

Examples

```
rnull(3, c(0.1, 0.5, 0.9), 0.1)
```

---

rwedge *Wedge distribution*

Description

The wedge distribution has a support of 0 to 1 and has a linear probability density function over that support.

Usage

```
rwedge(n, a)
```

Arguments

n	number of observations
a	a gradient from -2 (left skewed) to 2 (right skewed)

Details

The rwedge can be combined with quantile functions to skew standard distributions, or introduce correlation or down weight certain parts of the distribution.

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
pwedge(seq(0, 1, 0.1), a = 1)
dwedge(seq(0, 1, 0.1), a = 1)
qwedge(c(0.25, 0.5, 0.75), a = -1)

stats::cor(
  stats::qnorm(rwedge(1000, a = 2)),
  stats::qnorm(rwedge(1000, a = -2))
)
```

std_size *Standard image and paper sizes*

Description

The width and height of images to fit scientific publication standards.

Usage

```
std_size
```

Format

A list with width and height in inches

u_pit_significance *Significance test for the bias-adjusted undirected PIT-Wasserstein metric*

Description

Tests the weakest of the three null hypotheses in the PIT-Wasserstein family: that the predictive distributions have the **right shape given an arbitrary location error**. The bias adjustment absorbs any common shift, so a rejection here means the shape of the predictive distribution is wrong — skew, tail weight, or a mixture of dispersion errors — and not that the forecasts are merely off-centre. One-sided, as `U_pit` is a distance.

Usage

```
u_pit_significance(U_pit, n_predictions)
```

Arguments

`U_pit`	the bias-adjusted undirected PIT-Wasserstein metric, as returned by `pit_w()` or `prediction_scores()`
`n_predictions`	the number of predictions each `U_pit` was computed from, as returned by `prediction_scores()`

Details

Because the adjustment estimates a parameter from the same data, the null is **not** the baseline $\int_0^1 |B^\circ|$ of `b_pit_significance()`: it is the bridge with the estimated direction projected out (see `.u_pit_null`), giving a 5% critical value about `0.66` times the baseline one at every `n`. Using the baseline null here would be badly conservative. The adjusted null is still distribution-free — it depends on `n` and nothing else — but has no published c.d.f., so it is simulated once per session and cached.

`U_pit` says nothing about the *direction* of a dispersion error; pair it with `d_pit_significance()`, which does.

Value

a dataframe with one row per input value and columns:

- `U_pit`, `n_predictions` — as supplied
- `statistic` — $\sqrt{n}\,U_{pit}$, the quantity with a fixed null
- `expected` — the null mean of `U_pit`
- `critical_0.05` — the one-sided 5% critical value of `U_pit`
- `p_value` — the probability of a `U_pit` at least this large under the location-free null

Unit tests

```
# the adjusted null is strictly tighter than the baseline one
testthat::expect_lt(
  u_pit_significance(0, 100)$critical_0.05,
  b_pit_significance(0, 100)$critical_0.05
)

# the critical value scales as 1/sqrt(n)
testthat::expect_equal(
  u_pit_significance(0, 400)$critical_0.05 * 2,
  u_pit_significance(0, 100)$critical_0.05,
  tolerance = 1e-6
)

# a statistic at the null mean is not significant
testthat::expect_gt(u_pit_significance(u_pit_significance(0, 100)$expected, 100)$p_value, 0.3)

# the 5% critical value returns a p value of about 0.05
crit = u_pit_significance(0, 100)$critical_0.05
testthat::expect_equal(u_pit_significance(crit, 100)$p_value, 0.05, tolerance = 0.005)

# vectorised over both arguments
testthat::expect_equal(nrow(u_pit_significance(c(0.02, 0.09), c(76, 76))), 2L)
```

---

wedge *Wedge distribution*

Description

The wedge distribution has a support of 0 to 1 and has a linear probability density function over that support.

Arguments

n	number of observations
x	vector of quantiles
q	vector of quantiles
p	vector of probabilities
log	logical; if TRUE, probabilities p are given as log(p).
log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are P[X<=x] otherwise P[X>x].
a	a gradient from -2 (left skewed) to 2 (right skewed)

Details

The rwedge can be combined with quantile functions to skew standard distributions, or introduce correlation or down weight certain parts of the distribution.

Value

a vector of probabilities, quantiles, densities or samples.

Examples

```
pwedge(seq(0, 1, 0.1), a = 1)
dwedge(seq(0, 1, 0.1), a = 1)
qwedge(c(0.25, 0.5, 0.75), a = -1)

stats::cor(
  stats::qnorm(rwedge(1000, a = 2)),
  stats::qnorm(rwedge(1000, a = -2))
)
```

Index

∗ **distributions**
dbeta2, 10
dcgamma, 11
ddist, 12
dgamma2, 12
dgamma4, 13
dlnorm2, 14
dlnorm4, 15
dlogitnorm, 17
dlogitnorm2, 17
dnbinom2, 18
dnorm4, 20
dnull, 22
dwedge, 22
pbeta2, 25
pcgamma, 26
pdist, 27
pgamma2, 27
pgamma4, 28
plnorm2, 30
plnorm4, 31
plogitnorm, 33
plogitnorm2, 34
pnbinom2, 34
pnorm4, 36
pnull, 38
pwedge, 40
qbeta2, 41
qcgamma, 42
qdist, 43
qgamma2, 43
qgamma4, 44
qlnorm2, 45
qlnorm4, 46
qlogitnorm, 48
qlogitnorm2, 48
qnbinom2, 49
qnorm4, 51
qnull, 53
qwedge, 53
rbern, 54
rbeta2, 55
rcategorical, 56
rcgamma, 56
rdist, 57
reparam_fn, 58
rexpgrowth, 58
rexpgrowthI0, 59
rgamma2, 60
rgamma4, 61
rlnorm2, 62
rlnorm4, 63
rlogitnorm, 65
rlogitnorm2, 65
rnbinom2, 66
rnorm4, 67
rnull, 69
rwedge, 70
wedge, 72
∗ **output**
std_size, 71
∗ **scores**
crps_fn_generator, 7
pit_w, 29
prediction_scores, 39
∗ **significance**
b_pit_significance, 3
d_pit_significance, 8
p_l1_bridge, 24
u_pit_significance, 71
∗ **simulations**
calibrated_locations, 5
calibrated_locations_general, 6
configuration_plot, 6
generate_simulation, 23

b_pit_significance, 3
b_pit_significance(), *71*

calibrated_locations, 5
calibrated_locations_general, 6
configuration_plot, 6
crps_fn_generator, 7

d_pit_significance, 8
d_pit_significance(), *4*, *71*
dbeta2, 10
dbinom, *19*, *36*, *50*, *67*
dcgamma, 11
ddist, 12
dgamma2, 12
dgamma4, 13
dgeom, *19*, *36*, *50*, *67*
Distributions, *15, 16*, *19*, *21*, *31*, *33*, *36, 37*, *46, 47*, *50*, *52*, *63, 64*, *67*, *69*
dlnorm, *21*, *37*, *52*, *69*
dlnorm2, 14
dlnorm4, 15
dlogitnorm, 17
dlogitnorm2, 17
dnbinom2, 18
dnorm, *15, 16*, *31*, *33*, *46, 47*, *63, 64*
dnorm4, 20
dnull, 22
double, *19*, *35*, *50*, *67*
dpois, *19*, *36*, *50*, *67*
dwedge, 22

generate_simulation, 23

integer, *19*, *35*, *50*, *67*

p_l1_bridge, 24
p_l1_bridge(), *4*
pbeta, *19*, *36*, *50*, *67*
pbeta2, 25
pcgamma, 26
pdist, 27
pgamma, *18*, *35*, *50*, *66*
pgamma2, 27
pgamma4, 28
pit_w, 29
pit_w(), *4*, *8*, *71*
plnorm2, 30
plnorm4, 31
plogitnorm, 33
plogitnorm2, 34
pnbinom2, 34
pnorm4, 36
pnull, 38
prediction_scores, 39
prediction_scores(), *4*, *8*, *71*
pwedge, 40

q_l1_bridge *(*p_l1_bridge*)*, 24
qbeta2, 41
qcgamma, 42
qdist, 43
qgamma2, 43
qgamma4, 44
qlnorm2, 45
qlnorm4, 46
qlogitnorm, 48
qlogitnorm2, 48
qnbinom2, 49
qnorm4, 51
qnull, 53
qwedge, 53

rbern, 54
rbeta2, 55
rcategorical, 56
rcgamma, 56
rdist, 57
reparam_fn, 58
rexpgrowth, 58
rexpgrowthI0, 59
rgamma2, 60
rgamma4, 61
rlnorm2, 62
rlnorm4, 63
rlogitnorm, 65
rlogitnorm2, 65
rnbinom2, 66
RNG, *20*, *37*, *52*, *68*
rnorm4, 67
rnull, 69
rwedge, 70

stats::dbeta(), *10*
stats::dgamma(), *11*, *13*
stats::pbeta(), *26*
stats::pgamma(), *27–29*
stats::qbeta(), *41*
stats::qgamma(), *42*, *44*
stats::rbeta(), *55*
stats::rgamma(), *57*, *60*, *61*

std_size, 71

u_pit_significance, 71
u_pit_significance(), *4*, *8*

wedge, 72