Package: wsgnstat (via r-universe)

November 17, 2026

Type Package

Title The Wrapped Two-Piece Exponential Power Distribution

Version 0.1.1

Description The two-piece exponential power distribution on the line (Fernández and Steel, 1998, <doi:10.1080/01621459.1998.10474117>) and its wrapped version on the circle, treated as a uniform scale mixture with generalised gamma mixing (Ehlers, 2015, <doi:10.1080/01966324.2014.969460>). Given the mixing variable, the variable is uniform on an interval that contains the location. The package name follows the authors' earlier usage, the wrapped skew generalised normal distribution. Density, distribution and quantile functions, random generation from the mixture, and the characteristic function and trigonometric moments for every shape parameter, evaluated on a rotated path of integration, with the convergent series as a check. The wrapped density adds an Euler-Maclaurin tail to the image sum, so small shapes are covered. Maximum likelihood fitting of the four-parameter wrapped distribution under a lower bound on the shape, over both the regular and the spike branch, with the location searched over the observations when the shape is at most one; the grouped likelihood for rounded data; profile likelihoods; and standard errors from the expected information or a parametric bootstrap. A pure-C back-end does the numerics and also backs the 'Python' package 'wsgnstat'.

URL https://github.com/mtloots/wsgnstat

BugReports https://github.com/mtloots/wsgnstat/issues

License GPL-3

Encoding UTF-8

Language en-GB

Depends R (>= 3.5.0)

Imports stats

Suggests circular, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

NeedsCompilation yes

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Author M. Theodor Loots [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-7722-3913>)

Maintainer M. Theodor Loots <theo.loots@gmail.com>

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

Date/Publication 2026-11-17 08:40:02 UTC

RemoteUrl https://github.com/cran/wsgnstat

RemoteRef HEAD

RemoteSha 17332c43f22fcac5cdd9df848958bcd50c829cdf

Contents

cfsgn 2
dsgn 4
dwsgn 5
fit_wsgn 6
fit_wsgn_mle 10
fit_wsgn_trig 11
grouped_loglik 11
profile_wsgn 12
psgn 14
pwsgn 15
rsgn 16
rwsgn 17
trig_moments 18
wsgn_fit-methods 19

Index **20**

`cfsgn` *Characteristic function of the two-piece exponential power distribution (C back-end)*

Description

The characteristic function is `exp(i t mu) [w1 G(t sigma / lam1) + w2 Conj(G(t sigma / lam2))] / Gamma(1 + 1/beta)`, with `w1 = lam2 / (lam1 + lam2) = P(X > mu)`, `w2 = 1 - w1` and `G(w) = int_0^Inf exp(i w u - u^beta) du`. The default method evaluates `G` on the ray `u = r exp(i phi)` in the upper half-plane, with `phi` half of `min(pi, pi / (2 beta))`, by the trapezoid rule in `log r`, halving the step until successive sums agree to `tol`; the rotation removes both the oscillation and the very long range that the integral has on the real axis for small shapes.

Usage

```
cfsgn(
  t,
  mu = 0,
  sigma = 1,
  beta = 2,
  lam1 = 0.5,
  lam2 = 0.5,
  tol = 1e-12,
  method = c("quadrature", "series")
)
```

Arguments

t	numeric vector of arguments.
mu	location parameter.
sigma	scale parameter (> 0).
beta	shape parameter (> 0).
lam1	right-hand rate parameter (> 0); the right-hand rate is lam1 / sigma.
lam2	left-hand rate parameter (> 0); the left-hand rate is lam2 / sigma.
tol	tolerance relative to the size of each half (see Accuracy), for the quadrature and for the estimated rounding error of a series: one finite number in (0, 1). Values below 1e-14, beyond the rounding of the quadrature, are raised to 1e-14 with a warning.
method	"quadrature" (default) or "series".

Details

Accuracy. tol is relative to the size of each half, G(w) / Gamma(1 + 1/beta), which is of order min(1, 1/|w|). The two halves cancel to leading order at large |t|, so the error of the characteristic function is absolute, of order tol min(1,max(lam1, lam2) / (sigma |t|)) (in practice of order tol: about 1e-15 to 3e-15 for shapes up to about 3, a few times 1e-15 for shapes 5 to 20, and up to about 1e-14 for shapes up to 50, at the default tol); values much smaller than that carry no relative accuracy (for the normal member, for instance, beyond t of about 8 in standard-deviation units). Against 120-digit values of the convergent series at t = 1, 2, 3, lam1 = 0.3, lam2 = 0.7, sigma = 0.4 and sqrt(0.21), and shapes 0.1 to 3 (the file extdata/ref_cf.csv), the relative error is below 1e-14.

Series. The method "series" sums the convergent series instead, as a check: the descending series for beta < 1, the ascending (Fox-Wright) series for beta > 1, and 1 / (1 - i w) at beta = 1. Its rounding error is estimated as 1e-15 sum_k |t_k| (1 + |log t_k| + |arg t_k|) over the terms t_k, and a half is refused, with NaN and a warning, when that estimate exceeds tol relative to the half, when the series does not converge within 200000 terms, or when the sum breaks the bound |G(w)| <= Gamma(1 + 1/beta). On scans of 400 arguments per shape for shapes 0.05 to 50 the accepted values agreed with the quadrature to about tol relative to each half (to within a factor 3 at tol = 1e-8); like the quadrature, the combined value is accurate only in absolute terms at large |t|. Many arguments are refused, more for shapes near one.

Value

a complex vector of the same length as `t`.

Examples

```
## every shape is reachable: the value at zero is one, also for small shapes
sapply(c(0.1, 0.2, 0.3, 0.5, 1, 2), function(b) Mod(cfsgn(0, 0, 0.4, b, 0.3, 0.7) - 1))
## below shape one the descending series converges and agrees with the quadrature
t <- c(1, 2, 3)
cbind(quadrature = cfsgn(t, 0, 0.4, 0.3, 0.3, 0.7),
      series = cfsgn(t, 0, 0.4, 0.3, 0.3, 0.7, method = "series"))
## the normal member: exp(-t^2 / 4) for N(0, 1/2)
rbind(engine = Re(cfsgn(t, 0, 0.5, 2, 0.5, 0.5)), closed_form = exp(-t^2 / 4))
## the asymmetric Laplace member (beta = 1) in closed form
a <- 0.3 / 0.4
b <- 0.7 / 0.4
rbind(engine = cfsgn(t, 0, 0.4, 1, 0.3, 0.7),
      closed_form = a * b / ((a - 1i * t) * (b + 1i * t)))
```

`dsgn` *Density of the two-piece exponential power distribution on the line (C back-end)*

Description

The density is `c exp(-(lam1 (x - mu) / sigma)^beta)` for `x >= mu` and `c exp(-(lam2 (mu - x) / sigma)^beta)` for `x < mu`. With `lam1 = lam` and `lam2 = 1 - lam` it is the distribution of `mu + Z V`, where `Z^beta` is gamma distributed with shape `1 + 1/beta` and `V` is uniform on `[-sigma / (1 - lam), sigma / lam]`, independent of `Z`: a uniform scale mixture with generalised gamma mixing, with `P(X < mu) = lam`.

Usage

```
dsgn(x, mu = 0, sigma = 1, beta = 2, lam1 = 0.5, lam2 = 0.5)
```

Arguments

`x`	numeric vector of quantiles.
`mu`	location parameter.
`sigma`	scale parameter (> 0).
`beta`	shape parameter (> 0).
`lam1`	right-hand rate parameter (> 0); the right-hand rate is `lam1 / sigma`.
`lam2`	left-hand rate parameter (> 0); the left-hand rate is `lam2 / sigma`.

Value

the probability density function evaluated at `x`.

Examples

```
## a right-skew member with a cusp at its location, and the normal member
x <- seq(-4, 6, length.out = 401)
f_skew <- dsgn(x, mu = 0, sigma = 1, beta = 0.8, lam1 = 0.3, lam2 = 0.7)
f_norm <- dsgn(x, mu = 0, sigma = 0.5, beta = 2, lam1 = 0.5, lam2 = 0.5)
plot(x, f_skew, type = "l", ylab = "density")
lines(x, f_norm, lty = 2)
## at beta = 2 with lam1 = lam2 = 0.5 and sigma = 0.5 the distribution is N(0, 1/2)
max(abs(f_norm - dnorm(x, 0, sqrt(1 / 2))))
## the density integrates to one
integrate(dsgn, -Inf, Inf, mu = 0, sigma = 1, beta = 0.8, lam1 = 0.3, lam2 = 0.7)$value
```

`dwsgn` *Density of the wrapped two-piece exponential power distribution (C back-end)*

Description

The wrapped distribution is that of `X mod 2 pi`, with `X` the line distribution of `dsgn` with right rate `a = lam / sigma` and left rate `b = (1 - lam) / sigma`. Its four parameters are the location `mu`, the scale `sigma`, the skewness `lam` (`P(X < mu) = lam`) and the shape `beta`, and the four-parameter family is identifiable. When `sigma` is not given, the reciprocal-rate subfamily `a b = 1`, that is `sigma = sqrt(lam (1 - lam))`, is used, as in version 0.1.0; it is a three-parameter restriction of the model, kept for backward compatibility.

Usage

```
dwsgn(theta, mu = 0, beta = 2, lam = 0.5, K = 50, sigma = NULL, log = FALSE)
```

Arguments

`theta`	numeric vector of angles in radians.
`mu`	location parameter.
`beta`	shape parameter (> 0).
`lam`	skewness parameter in (0, 1); `0.5` is symmetric.
`K`	number of images summed directly on each side of `mu` before the Euler-Maclaurin tail (a positive integer); more are added directly when the formula is not yet accurate there, and fewer are used when the rest is negligible.
`sigma`	scale parameter (> 0); `NULL` gives the reciprocal-rate subfamily.
`log`	logical; return the log-density.

Details

The density is the image sum c sum_k [exp(-(a (t + 2 pi k))^beta) + exp(-(b (2 pi - t + 2 pi k))^beta)], t = (theta - mu) mod 2 pi, with c = lam (1 - lam) / (sigma Gamma(1 + 1/beta)). On each side of mu images are summed directly, and the remainder is added by the Euler-Maclaurin formula, with the exact incomplete-gamma integral and Bernoulli terms through B_12, as soon as it is accurate; it is tried after 8, 16, 32 and K images and after each further one, so the result does not depend on K. The tail is kept in log space, so very small shapes are covered (at beta = 0.005 and 0.001 the density differs from 1/(2 pi) by less than 1e-13 relative, the limit set by cancelling lgamma(1/beta)). Against 40-digit image sums at six angles, lam = 0.3, sigma = 0.4 and sqrt(0.21), and shapes 0.1, 0.2, 0.3, 0.5, 1 and 2 (the file extdata/ref_dw.csv), the relative error is below 1e-14.

Value

the wrapped density (or its logarithm) evaluated at theta.

Examples

```
## a skew member with a cusp, drawn on the circle as a radial curve about the unit circle
th <- seq(0, 2 * pi, length.out = 721)
d <- dwsgn(th, mu = pi / 2, beta = 0.8, lam = 0.3, sigma = 0.4)
plot(cos(th), sin(th), type = "l", asp = 1, xlim = c(-2.5, 2.5), ylim = c(-2.5, 2.5),
     xlab = "", ylab = "", lty = 3)
lines((1 + d) * cos(th), (1 + d) * sin(th))
## the density integrates to one over the circle, also at a very small shape
integrate(dwsgn, 0, 2 * pi, mu = 1, beta = 0.2, lam = 0.3, sigma = 0.4,
          subdivisions = 1000)$value
## at beta = 2 with lam = 0.5 the circular distribution is the wrapped normal
## with standard deviation sqrt(2) sigma
tt <- c(0.5, 2.0, 4.5)
wn <- vapply(tt, function(t) sum(dnorm(t + 2 * pi * (-80:80), 0, sqrt(2) * 0.6)), 0)
rbind(wsgnstat = dwsgn(tt, 0, 2, 0.5, sigma = 0.6), wrapped_normal = wn)
```

---

fit_wsgn *Fit the wrapped two-piece exponential power distribution by maximum likelihood (C back-end)*

Description

Maximum likelihood fitting of the full four-parameter wrapped distribution (mu, sigma, lam, beta). Every step runs in C: the optimiser, the starting values, the location search and the bootstrap refits.

Usage

```
fit_wsgn(
  theta,
  beta_floor = NULL,
  resolution = NULL,
```

```
  origin = 0,
  symmetric = FALSE,
  se = c("auto", "information", "bootstrap", "none"),
  B = 200,
  window = 0.4,
  K = 50,
  ngrid = 20000,
  ncores = 1
)
```

Arguments

theta	numeric vector of angles in radians.
beta_floor	lower bound on the shape; NULL gives 0.1 for exact data and 0.001 for grouped data.
resolution	width of the recording cells in radians (for instance pi / 18 for 10 degrees), dividing the circle into a whole number of cells, at most 1e6 of them (a finer resolution is refused); NULL for exact data.
origin	centre of one recording cell, in radians.
symmetric	logical; fit the symmetric member lam = 0.5.
se	"auto", "information", "bootstrap" or "none".
B	number of bootstrap replicates.
window	half-width (radians) of the location search about the fitted location.
K	largest number of images summed directly, as in dwsgn.
ngrid	number of points of the circle in the quadrature for the information, a whole number from 100 to 1e8.
ncores	number of threads, a whole number of at least one (at most 256; more are reduced to 256 with a message), used when the package is built with OpenMP: for the independent trials inside a fit and, with se = "bootstrap", for the refits. The result does not depend on it. Keep it at 1 in examples and tests.

Details

Exact data. With the shape free the likelihood is unbounded as beta tends to zero, so it is maximised under a lower bound beta >= beta_floor (default 0.1), under which it is bounded and the estimator is consistent whenever the floor does not exceed the true shape. The estimate is the maximiser of the likelihood over beta_floor <= beta <= 20, |log a|, |log b| <= R, where R = max(30, 3 (1 - log beta_floor) / beta_floor) (99 at the default floor). Two searches are made and the best result is returned. The first, for the regular branch, runs Nelder-Mead from 27 starting values about the circular mean and 6 one-sided starts (one log-rate near the bound, the location at either end of the widest gap between observations); when its shape is at most one, every maximising location is an observation, so the location is then searched over the stored observations within window of it, followed by a free polish. The second is for the spike branch, whose likelihood grows as the shape falls and whose profile in the shape can be ragged just above the floor: the same procedure, with a spike start on the most-tied observation, is run at the fixed shapes beta_floor times 1, 1.25 and 1.6 with the location searched over every observation, each followed by a free polish (the 1.25 and 1.6

shapes are skipped when they reach half the upper bound, that is when the floor is 6.25 or more). In every location search each candidate observation is screened after a short optimisation of the other parameters (from the current fit and from a spike as wide as the gap to its nearest neighbour); all candidates are screened when there are at most 200 of them, otherwise the 15 best at the current fit, the 15 with the smallest gap to a neighbour, the 15 with the smallest gap to the further neighbour and every tied value. The 12 best screened candidates and every tied value are refined further, and the best two fully. `profile_wsgn` uses the same search at its fixed shapes, so the profile at the floor equals the fit's floor search. With no ties and a moderate sample the spike often wins, and the fit is then on the floor or just above it: `at_floor` reports a fit on the floor. `at_upper` reports a fit on the upper bound 20 and `at_rate_bound` one on the rate bound `R` (a degenerate fit, for instance a spike or a one-sided limit, where `lam` may round to 0 or 1 and `sigma` to almost 0; the log-rates in `log_rates` are exact). `at_one_sided` reports a one-sided fit that stops short of the rate bound (one rate effectively infinite, `lam` within about `1e-12` of 0 or 1, so the two log-rates differ by more than `log(1e12)`), for exact and grouped data alike, judged from the log-rates; it is the true maximum on the boundary, kept as such. The search is not exhaustive. On the cases checked (the review cases, the periwinkles and the wind directions) it matched or beat independent searches, but more distant branches cannot be excluded; compare with `profile_wsgn`. Without standard errors a fit takes about 2.5 seconds for the 31 periwinkles, 5 to 10 seconds for 60 observations, 20 for 150 and 40 for 300 on a current laptop.

Exact data need at least three distinct angles, and all angles must be finite.

Rounded data. When the data are recorded to a resolution (whole degrees, 5 or 10 degrees, compass points), give `resolution` (in radians): the observations are assigned to cells of that width centred at `origin + j * resolution`, and the grouped likelihood `sum_j N_j log P(Theta in C_j)`, the likelihood of what was observed, is maximised. It is bounded with no constraint on the shape (the default floor is `0.001`; the upper bound is 50 and the rate bound 60), with the cell probabilities computed exactly from the incomplete-gamma survival functions. Starting values are 30 fixed-seed random points, every fifth a spike on the modal cell. The data must occupy at least two cells.

Standard errors. With `se = "auto"`, a continuous fit with `beta > 1` that is on none of the bounds gets standard errors from the expected Fisher information `I(theta)^-1 / n`, which is finite for `beta > 1/2`, the normal approximation being proved for a true shape above one; in a simulation at `n = 200` the 95 per cent Wald intervals covered 87 to 92 per cent, so they are somewhat optimistic at moderate `n`. In every other case `se = "auto"` returns no standard errors, with a message saying why: on a bound they have no meaning, and at a fitted shape at most one, or for grouped data, the bootstrap is available on request. A one-sided fit (`at_one_sided`: `min(lam, 1 - lam)` below about `1e-12`, so that one rate is effectively infinite, a maximum on the boundary that the optimiser can reach short of the rate bound) counts as on a bound. Standard errors are never returned as `NaN`, except the `lam` entry of a symmetric fit, where `lam` is fixed: if the information matrix is not finite and positive definite, `se = "auto"` gives none, with a message. `se = "information"` is refused for grouped data, a fit on a bound (the floor, the upper bound or the rate bound), a one-sided fit, and when the information matrix cannot be inverted. `se = "bootstrap"` draws `B` samples from the fitted rates (rounded to the same cells when grouped) and refits each by the same procedure; a refit takes as long as a fit, so `B = 200` takes minutes to hours (`ncores` helps where the package is built with OpenMP). At a fitted shape below one these are descriptive.

Speed. A fit is a search from many starting values, a screen of candidate locations and, at the smallest shapes, a spike search, and these trials are independent. With `ncores > 1` and a package built with OpenMP they run in parallel, and the best is chosen afterwards in a fixed order (ties to the lowest index), so the fit is identical, to the last bit, for every number of threads. A bootstrap spreads

its refits over the threads instead, each refit running on one thread. `ncores = 1` (the default, as CRAN requires) runs everything serially. A likelihood evaluation with 128 or more observations (or cells) is itself split over the threads and summed in a fixed order. On a laptop with four performance and six efficiency cores, busy with other work, a fit of 1500 observations took 160 seconds on one thread, 86 on two, 57 on four and 55 on eight, and one of 400 observations 36, 18, 11 and 13 seconds: about a factor of 2.8 on four threads, and little more beyond. The gain stops there because some of the work still runs in sequence (the first pass of the location search, the Nelder-Mead runs and the final polish), and because the extra threads of such a machine run on its slower cores. A fit takes minutes from about a thousand observations on. Where the package is built without OpenMP the argument is accepted and has no effect.

Value

an object of class `"wsgn_fit"`: a list with `par` (`mu`, `sigma`, `lam`, `beta`), `rates` (`a`, `b`), `log_rates` (`log_a`, `log_b`, exact also on a degenerate fit), `logLik`, `se`, `vcov` (information-based standard errors only), `boot` (the bootstrap fits, if any), `frac_beta_lt1` (fraction of bootstrap fits with shape below one), `se_method`, `beta_floor`, `at_floor`, `at_upper`, `at_rate_bound`, `at_one_sided`, `n`, `grouped`, `resolution`, `symmetric` and `nfev`.

See Also

`profile_wsgn` for the profile likelihood in the shape, and `fit_wsgn_mle` for the reciprocal-rate subfamily of version 0.1.0.

Examples

```
## simulate from a skew member with a shape above one and recover it
set.seed(4)
th <- rwsgn(40, mu = 1, beta = 1.6, lam = 0.3, sigma = 0.4)
fit <- fit_wsgn(th)
fit

## the same fit on two threads (where the package is built with OpenMP) is identical
identical(fit$par, fit_wsgn(th, ncores = 2)$par)
## the same data recorded to whole multiples of 10 degrees: the grouped likelihood
h <- 10 * pi / 180
th10 <- (round(th / h) * h) %% (2 * pi)
fit10 <- fit_wsgn(th10, resolution = h, se = "none")
rbind(exact = coef(fit), grouped = coef(fit10))
## a shape below one: the location is an observation (se = "bootstrap" would add
## bootstrap standard errors, at the cost of one such fit per sample)
th2 <- rwsgn(60, mu = 2, beta = 0.9, lam = 0.4, sigma = 0.3)
fit2 <- fit_wsgn(th2, se = "none")
coef(fit2)
any(th2 %% (2 * pi) == fit2$par["mu"])
```

`fit_wsgn_mle` *Fit the reciprocal-rate subfamily by maximum likelihood*

Description

Nelder-Mead over the three parameters (`mu`, `beta`, `lam`) of the reciprocal-rate subfamily `sigma = sqrt(lam (1 - lam))`, a restriction of the model on which the likelihood is bounded, with optional parametric-bootstrap standard errors. Kept from version 0.1.0 for backward compatibility; `fit_wsgn` fits the full four-parameter distribution. Since version 0.1.1 the wrapped density includes the Euler-Maclaurin tail, so log-likelihoods at shapes below about 0.6 differ from those of version 0.1.0, which were truncated.

Usage

```
fit_wsgn_mle(theta, start = NULL, K = 50, B = 0)
```

Arguments

`theta`	numeric vector of angles in radians.
`start`	numeric starting value `c(mu, beta, lam)`; default is the trigonometric-moment estimate.
`K`	largest number of images summed directly, as in `dwsgn`.
`B`	number of parametric-bootstrap replicates for standard errors; `0` skips them.

Value

a list with `par`, `logLik`, and (if `B > 0`) `se`.

Examples

```
set.seed(12)
truth <- c(mu = 1, beta = 1.5, lam = 0.3)
th <- rwsgn(500, truth["mu"], truth["beta"], truth["lam"])
fit <- fit_wsgn_mle(th)
round(rbind(truth = truth, estimate = fit$par), 3)
fit$logLik

## bootstrap standard errors refit the model once per replicate
fit_b <- fit_wsgn_mle(th, B = 30)
round(rbind(estimate = fit_b$par, se = fit_b$se), 3)
```

fit_wsgn_trig *Fit the reciprocal-rate subfamily by trigonometric moments*

Description

Minimum-distance matching of the first two sample trigonometric moments to those of the three-parameter reciprocal-rate subfamily (`sigma = sqrt(lam (1 - lam))`), with standard errors from the minimum-distance sandwich. Kept from version 0.1.0 for backward compatibility. Matching trigonometric moments is useful as a diagnostic and for starting values, but it is not a reliable estimator for this family; `fit_wsgn` fits the full four-parameter distribution by maximum likelihood.

Usage

```
fit_wsgn_trig(theta, starts = NULL)
```

Arguments

`theta`	numeric vector of angles in radians.
`starts`	list of numeric starting values `c(mu, beta, lam)`.

Value

a list with the estimate `par = c(mu, beta, lam)`, its standard errors `se`, and the objective value `value`.

Examples

```
set.seed(11)
truth <- c(mu = 1, beta = 1.5, lam = 0.3)
th <- rwsgn(1000, truth["mu"], truth["beta"], truth["lam"])
fit <- fit_wsgn_trig(th)
round(rbind(truth = truth, estimate = fit$par, se = fit$se), 3)
```

grouped_loglik *Grouped log-likelihood of rounded circular data (C back-end)*

Description

The likelihood of what is observed when angles are recorded to a resolution: the observations are assigned to cells of width `resolution` centred at `origin + j * resolution`, and the grouped log-likelihood is `sum_j N_j log P(Theta in C_j)`, with the cell probabilities computed exactly as in `pwsgn`. It is at most zero and bounded for every parameter value, and `fit_wsgn` maximises it when `resolution` is given.

Usage

```
grouped_loglik(theta, resolution, mu, sigma, lam, beta, origin = 0, K = 50)
```

Arguments

theta	numeric vector of angles in radians.
resolution	width of the recording cells in radians (for instance pi / 18 for 10 degrees), dividing the circle into a whole number of cells, at most 1e6 of them (a finer resolution is refused); NULL for exact data.
mu, sigma, lam, beta	parameters of the wrapped distribution, as in dwsgn.
origin	centre of one recording cell, in radians.
K	largest number of images summed directly, as in dwsgn.

Value

the grouped log-likelihood.

Examples

```
## ants' directions recorded to the nearest 10 degrees (Fisher, 1993, data set B.7)
if (requireNamespace("circular", quietly = TRUE)) {
  e <- new.env()
  utils::data("fisherB7", package = "circular", envir = e)
  ants <- e$fisherB7 * pi / 180
  grouped_loglik(ants, pi / 18, mu = pi, sigma = 0.1, lam = 0.5, beta = 0.6)
}
## the grouped likelihood of a single cell holding all the data is zero at a spike
grouped_loglik(rep(1, 20), pi / 18, mu = 1, sigma = 1e-3, lam = 0.5, beta = 2, origin = 1)
```

---

profile_wsgn *Profile log-likelihood of the wrapped distribution in the shape (C back-end)*

Description

M(beta), the supremum of the log-likelihood over (mu, sigma, lam) at each fixed shape, for exact data or (with resolution) for the grouped likelihood. For exact data the supremum tends to infinity as beta tends to zero, at the leading rate n log(n / (n - m)) / beta, with m the largest number of tied observations, so the profile shows where that degenerate branch overtakes the ordinary fit. At shapes up to one the location is searched over the stored observations within window of the best free fit, as described for fit_wsgn (every observation screened when there are at most 200); the default window = pi covers all of them. The rate bound is that of fit_wsgn with the same beta_floor, max(30, 3 (1 - log b) / b) with b the smaller of beta_floor and the shape, so the profile at the floor equals the floor search of the fit and the profile at any shape at or above the floor uses the fit's bound (the bound is computed from beta_floor, not from each profiled shape). With ncores > 1 and a package built with OpenMP the shapes, and the trials inside each fit, share the threads; the profile is identical for every number of threads.

Usage

```
profile_wsgn(
  theta,
  beta,
  resolution = NULL,
  origin = 0,
  symmetric = FALSE,
  window = pi,
  K = 50,
  beta_floor = NULL,
  ncores = 1
)
```

Arguments

theta	numeric vector of angles in radians.
beta	numeric vector of shapes.
resolution	width of the recording cells in radians (for instance pi / 18 for 10 degrees), dividing the circle into a whole number of cells, at most 1e6 of them (a finer resolution is refused); NULL for exact data.
origin	centre of one recording cell, in radians.
symmetric	logical; fit the symmetric member lam = 0.5.
window	half-width (radians) of the location search about the fitted location.
K	largest number of images summed directly, as in dwsgn.
beta_floor	the shape floor of the fit to compare with, which sets the rate bound (NULL: as in fit_wsgn).
ncores	number of threads, a whole number of at least one (at most 256; more are reduced to 256 with a message), used when the package is built with OpenMP for the shapes and the trials inside each fit. The result does not depend on it.

Value

a data frame with columns beta, logLik, mu, sigma and lam.

Examples

```
set.seed(22)
th <- rwsgn(50, mu = 1, beta = 1.2, lam = 0.35, sigma = 0.4)
pr <- profile_wsgn(th, beta = c(0.8, 1.2, 2), window = 0.4)
pr
plot(pr$beta, pr$logLik, type = "b", xlab = "shape", ylab = "profile log-likelihood")
```

psgn *Distribution function of the two-piece exponential power distribution on the line*

Description

Evaluated in closed form through the regularised incomplete gamma function: the mass to the left of `mu` is `lam1 / (lam1 + lam2)`, and each tail is a scaled gamma tail in `(lam * |x - mu| / sigma)^beta`.

Usage

```
psgn(q, mu = 0, sigma = 1, beta = 2, lam1 = 0.5, lam2 = 0.5)
```

Arguments

`q`	numeric vector of quantiles.
`mu`	location parameter.
`sigma`	scale parameter (> 0).
`beta`	shape parameter (> 0).
`lam1`	right-hand rate parameter (> 0); the right-hand rate is `lam1 / sigma`.
`lam2`	left-hand rate parameter (> 0); the left-hand rate is `lam2 / sigma`.

Value

the distribution function evaluated at `q`.

Examples

```
## agreement with numerical integration of the density
q <- c(-1, 0, 0.7, 2.5)
closed <- psgn(q, 0, 1, 1.3, 0.3, 0.8)
numeric <- vapply(q, function(u)
  integrate(dsgn, -Inf, u, mu = 0, sigma = 1, beta = 1.3, lam1 = 0.3, lam2 = 0.8)$value, 0)
rbind(closed, numeric)
## the normal member: psgn equals pnorm with variance 1/2
psgn(0.4, 0, 0.5, 2, 0.5, 0.5) - pnorm(0.4, 0, sqrt(1 / 2))
```

pwsgn *Distribution and quantile functions of the wrapped two-piece exponential power distribution*

Description

The distribution function is measured anticlockwise from angle `0`: `pwsgn(theta)` is the probability of the arc `[0, theta]`. It is computed from the incomplete-gamma survival functions of the two sides, summed over images, with the Euler-Maclaurin tail beyond the `K`-th image. The arc is described by its start and its width, so a short arc keeps its relative accuracy: for an arc shorter than 0.01 the first image is taken from the incomplete gamma function where that is accurate and by Gauss-Legendre quadrature of the density otherwise, and the incomplete gamma functions are evaluated from the logarithm of their argument, so that `(rate x)^beta` never underflows next to the location, at any shape. Small probabilities therefore keep their relative accuracy down to the underflow limit, on both sides of the location and for large shapes too (checked for shapes 0.3 to 50). The quantile function inverts it by bisection with geometric midpoints while the bracket is wide, so that small probabilities get quantiles with relative accuracy (for instance `qwsgn(1e-300)`). The parameters are as in dwsgn.

Usage

```
pwsgn(q, mu = 0, beta = 2, lam = 0.5, K = 50, sigma = NULL)

qwsgn(p, mu = 0, beta = 2, lam = 0.5, K = 50, sigma = NULL)
```

Arguments

`q`	numeric vector of angles in radians.
`mu`	location parameter.
`beta`	shape parameter (> 0).
`lam`	skewness parameter in (0, 1); `0.5` is symmetric.
`K`	number of images summed directly on each side of `mu` before the Euler-Maclaurin tail (a positive integer); more are added directly when the formula is not yet accurate there, and fewer are used when the rest is negligible.
`sigma`	scale parameter (> 0); `NULL` gives the reciprocal-rate subfamily.
`p`	numeric vector of probabilities.

Value

`pwsgn` gives probabilities in `[0, 1]`: `0` for `q <= 0` and `1` for `q >= 2 * pi`. `qwsgn` gives angles in `[0, 2 * pi]`.

Examples

```
## the distribution function is the integral of the density
pwsgn(2, mu = 1, beta = 1.5, lam = 0.3, sigma = 0.5)
integrate(dwsgn, 0, 2, mu = 1, beta = 1.5, lam = 0.3, sigma = 0.5)$value
## the quantile function inverts it
p <- c(0.05, 0.25, 0.5, 0.75, 0.95)
q <- qwsgn(p, mu = 1, beta = 1.5, lam = 0.3, sigma = 0.5)
rbind(p, pwsgn(q, mu = 1, beta = 1.5, lam = 0.3, sigma = 0.5))
## the curve rises from 0 to 1 around the circle
th <- seq(0, 2 * pi, length.out = 361)
plot(th, pwsgn(th, mu = 1, beta = 1.5, lam = 0.3, sigma = 0.5), type = "l",
     xlab = "angle (radians)", ylab = "distribution function")
```

`rsgn` *Random generation from the two-piece exponential power distribution on the line*

Description

Uses the uniform scale mixture: `Z^beta` is drawn from the gamma distribution with shape `1 + 1/beta`, and the draw is uniform on `[mu - Z sigma / lam2, mu + Z sigma / lam1]`, an interval that contains `mu`; a single uniform variable decides the side. The draws are reduced modulo `2 pi` when `wrap = TRUE`.

Usage

```
rsgn(n, mu = 0, sigma = 1, beta = 2, lam1 = 0.5, lam2 = 0.5, wrap = FALSE)
```

Arguments

`n`	sample size.
`mu`	location parameter.
`sigma`	scale parameter (> 0).
`beta`	shape parameter (> 0).
`lam1`	right-hand rate parameter (> 0); the right-hand rate is `lam1 / sigma`.
`lam2`	left-hand rate parameter (> 0); the left-hand rate is `lam2 / sigma`.
`wrap`	logical; reduce modulo `2 pi`.

Value

a numeric vector of draws.

See Also

`rwsgn` for draws from the wrapped distribution.

Examples

```
set.seed(2)
x <- rsgn(5000, mu = 0, sigma = 1, beta = 1.3, lam1 = 0.3, lam2 = 0.7)
hist(x, breaks = 60, freq = FALSE, main = "")
curve(dsgn(x, 0, 1, 1.3, 0.3, 0.7), add = TRUE)
## the mass to the left of the location is lam1 / (lam1 + lam2) = 0.3
mean(x < 0)
```

`rwsgn` *Random generation from the wrapped two-piece exponential power distribution*

Description

Draws from the wrapped distribution with the parameters of dwsgn, that is `rsgn(n, mu, sigma, beta, lam1 = lam, lam2 = 1 - lam, wrap = TRUE)`. When `sigma` is not given, the reciprocal-rate subfamily `sigma = sqrt(lam (1 - lam))` is used.

Usage

```
rwsgn(n, mu = 0, beta = 2, lam = 0.5, sigma = NULL)
```

Arguments

`n`	number of draws.
`mu`	location parameter.
`beta`	shape parameter (> 0).
`lam`	skewness parameter in (0, 1); `0.5` is symmetric.
`sigma`	scale parameter (> 0); `NULL` gives the reciprocal-rate subfamily.

Value

a numeric vector of angles in `[0, 2 * pi)`.

Examples

```
set.seed(3)
th <- rwsgn(2000, mu = 1, beta = 0.9, lam = 0.3, sigma = 0.4)
hist(th, breaks = seq(0, 2 * pi, length.out = 49), freq = FALSE, main = "",
     xlab = "angle (radians)")
curve(dwsgn(x, mu = 1, beta = 0.9, lam = 0.3, sigma = 0.4), 0, 2 * pi, add = TRUE, n = 721)
```

`trig_moments` *Trigonometric moments of the wrapped distribution (C back-end)*

Description

The trigonometric moments of the wrapped distribution are the values of the characteristic function of the line distribution at the integers, `phi_p = E exp(i p Theta) = A_p + i B_p`, computed as in `cfsgn`, so their error is absolute, of order `tol / p` (in practice of order `tol`: about `1e-15` to `3e-15` for shapes up to about 3, a few times `1e-15` for shapes 5 to 20, and up to about `1e-14` for shapes up to 50, at the default `tol`); moments much smaller than that (the far moments of a smooth member) carry no relative accuracy. `trig_decay` gives the constant `C` of the exact decay rate `|phi_p| ~ C p^-(1 + beta)` as `p` grows, which holds unless `lam = 0.5` and `beta` is an even integer (then `C = 0` and the decay is faster than any power).

Usage

```
trig_moments(P = 2, mu = 0, beta = 2, lam = 0.5, sigma = NULL, tol = 1e-12)

trig_decay(beta = 2, lam = 0.5, sigma = NULL)
```

Arguments

`P`	highest order `p`, a whole number from 1 to `1e7`.
`mu`	location parameter.
`beta`	shape parameter (> 0).
`lam`	skewness parameter in (0, 1); `0.5` is symmetric.
`sigma`	scale parameter (> 0); `NULL` gives the reciprocal-rate subfamily.
`tol`	tolerance, as in `cfsgn`: one finite number in (0, 1), raised to `1e-14` if smaller.

Value

`trig_moments`: a list with components `A` and `B`, the cosine and sine moments for `p = 1, ..., P`. `trig_decay`: a named vector with the constant `C` and the exponent `1 + beta`.

Examples

```
## the reduction to the wrapped normal: exp(-sigma^2 p^2) at beta = 2, lam = 0.5
tm <- trig_moments(4, mu = 0, beta = 2, lam = 0.5, sigma = 0.5)
rbind(engine = tm$A, closed_form = exp(-0.25 * (1:4)^2))
## a skew member with a small shape: the moments approach C p^-(1 + beta)
tm <- trig_moments(400, mu = 0, beta = 0.3, lam = 0.3, sigma = 0.4)
dc <- trig_decay(beta = 0.3, lam = 0.3, sigma = 0.4)
p <- c(10, 100, 400)
Mod(complex(real = tm$A[p], imaginary = tm$B[p])) * p^dc["exponent"] / dc["C"]
## the first moment against a large simulated sample
tm1 <- trig_moments(1, mu = 1, beta = 1.5, lam = 0.3, sigma = 0.5)
set.seed(1)
```

```
th <- rwsgn(1e5, mu = 1, beta = 1.5, lam = 0.3, sigma = 0.5)
rbind(population = c(tm1$A, tm1$B), sample = c(mean(cos(th)), mean(sin(th))))
```

`wsgn_fit-methods` *Methods for fitted wrapped two-piece exponential power distributions*

Description

Print, coefficients, covariance matrix and log-likelihood of a fit from `fit_wsgn`.

Usage

```
## S3 method for class 'wsgn_fit'
print(x, digits = 4, ...)

## S3 method for class 'wsgn_fit'
coef(object, ...)

## S3 method for class 'wsgn_fit'
vcov(object, ...)

## S3 method for class 'wsgn_fit'
logLik(object, ...)
```

Arguments

`x, object`	a `"wsgn_fit"` object.
`digits`	number of significant digits to print.
`...`	unused.

Value

`print` returns `x` invisibly; `coef` the estimates (`mu`, `sigma`, `lam`, `beta`); `vcov` the information-based covariance matrix (`NULL` for bootstrap standard errors, whose replicates are in `object$boot`); `logLik` an object of class `"logLik"` (the grouped log-likelihood for grouped data).

Examples

```
set.seed(7)
fit <- fit_wsgn(rwsgn(30, mu = 1, beta = 2, lam = 0.4, sigma = 0.5))
print(fit)
coef(fit)
vcov(fit)
logLik(fit)
AIC(fit)
```

Index

cfsgn, 2, *18*
coef.wsgn_fit *(*wsgn_fit-methods*)*, 19

dsgn, 4, *5*
dwsgn, 5, *7*, *10*, *12, 13*, *15*, *17*

fit_wsgn, 6, *10–13*, *19*
fit_wsgn_mle, *9*, 10
fit_wsgn_trig, 11

grouped_loglik, 11

logLik.wsgn_fit *(*wsgn_fit-methods*)*, 19

print.wsgn_fit *(*wsgn_fit-methods*)*, 19
profile_wsgn, *8, 9*, 12
psgn, 14
pwsgn, *11*, 15

qwsgn *(*pwsgn*)*, 15

rsgn, 16
rwsgn, *16*, 17

trig_decay *(*trig_moments*)*, 18
trig_moments, 18

vcov.wsgn_fit *(*wsgn_fit-methods*)*, 19

wsgn_fit-methods, 19