Package: arcstat (via r-universe)

October 13, 2026

Type Package

Title Arc-Length Statistics: Goodness of Fit, Distributions and a Bayesian Test

Version 0.3.1

Description Inference from the arc length of statistical functions. Three tools share one pure-C back-end: a goodness-of-fit test based on the arc length of the probability plot, with an analytic saddlepoint null and sensitivity to local density structure that the empirical-distribution tests miss; two constructions that build a distribution from the arc length of its defining curve, the arc-length generator and the quantile arc-length family estimated by L-moments; and a Bayesian nonparametric arc-length goodness-of-fit test on the Dirichlet-process posterior. The same C sources back the 'Python' package 'arcstat'.

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

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

License GPL-3

Encoding UTF-8

Imports stats

NeedsCompilation yes

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

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

VignetteBuilder knitr

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

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

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

Date/Publication 2026-10-12 08:00:02 UTC

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

RemoteRef HEAD

RemoteSha d617dbcf32c4d5b03cfd33d98303bdbf87b18031

Contents

al_band_model . . . 3
al_band_model_star . . . 3
al_band_sample . . . 4
al_moments . . . 5
al_pvalue . . . 5
al_scale . . . 6
al_scale_raw . . . 6
al_statistic . . . 7
al_test . . . 7
arc_generator . . . 8
arcc_c3max . . . 9
arcc_exact_ci . . . 9
arcc_gof . . . 10
arccirc . . . 11
arccirc_fr . . . 12
arcq . . . 13
bb_evidence . . . 14
bb_post_disc . . . 15
bb_ref_disc . . . 16
bc_cdf . . . 16
bc_pdf . . . 17
bc_q . . . 17
cf_arclength . . . 18
cf_arclength_family . . . 19
dtemper_vm . . . 20
eq_bstar . . . 20
eq_E . . . 21
eq_readings . . . 22
eq_ub_quad . . . 22
eqfit_bc . . . 23
fit_arccirc . . . 25
fit_arccirc_fr . . . 26
fit_arcq . . . 26
fit_arcq_cf . . . 27
k4_band_model . . . 27
k4_band_trop_mean . . . 28
k4_fit_lmom . . . 29
k4_fit_varpro . . . 30
k4_icc . . . 31
k4_locus_dist . . . 31
k4_mv_boot . . . 32

k4_q 33
k4_readings 34
k4_runmed 34
k4_tau34 35
lmom_circ 35
lmoments 36
lmratios 37
sample_lmoments 38

Index **39**

`al_band_model` *Band arc length of the normal reference curve (C back-end)*

Description

Band arc length of the normal reference curve (C back-end)

Usage

```
al_band_model(sigma, a = 0.05, b = 0.95, nodes = 400L)
```

Arguments

`sigma`	scale of the normal reference;
`a, b`	the probability band;
`nodes`	quadrature panels.

Value

the model band arc length.

`al_band_model_star` *Spacing-corrected model band arc length (C back-end)*

Description

The sample band arc length does not converge to the population band arc length `al_band_model`. Consecutive order statistics inside the band are spaced like $Q'(u)E/n$ with E standard exponential, so the polygonal sum converges instead to

$$S^*(\sigma) = \int_{z_a}^{z_b} E\sqrt{f_0(z)^2 + \sigma^2 E^2}\, dz = E\, S(\sigma E),$$

which differs from $S(\sigma)$ by a constant factor because the square root is nonlinear and $E[E^2] = 2$. The gap does not close as the sample grows. This is the functional `al_scale` matches, and matching to it is what makes that estimator consistent.

Usage

```
al_band_model_star(sigma, a = 0.05, b = 0.95, nodes = 400)
```

Arguments

sigma	scale at which to evaluate the model band length.
a, b	the probability band.
nodes	number of Simpson nodes across the band; the exponential expectation uses a 64-point Gauss-Laguerre rule, which is exact to machine precision for this integrand.

Value

the spacing-corrected model band arc length at sigma.

---

al_band_sample *Sample band arc length of the empirical distribution curve (C back-end)*

Description

Sample band arc length of the empirical distribution curve (C back-end)

Usage

```
al_band_sample(x, a = 0.05, b = 0.95)
```

Arguments

x	data;
a, b	the probability band.

Value

the sample band arc length, NA when fewer than two points fall in the band.

al_moments — *Exact mean, variance and support of the arc-length statistic (C back-end)*

Description

Exact mean, variance and support of the arc-length statistic (C back-end)

Usage

```
al_moments(n)
```

Arguments

n	sample size.

Value

list with mean, var, support.

al_pvalue — *Saddlepoint right-tail probability of the arc-length statistic (C back-end)*

Description

Saddlepoint right-tail probability of the arc-length statistic (C back-end)

Usage

```
al_pvalue(s, n)
```

Arguments

s	observed statistic value;
n	sample size.

Value

the right-tail p-value.

al_scale — *Scale by arc-length band matching, in the scale-equivariant standardised form (C back-end)*

Description

The raw matching equation fixes an aspect ratio between the probability and response axes and is therefore unit dependent; this estimator divides by the MAD, matches on the standardised scale and rescales, which is exactly scale equivariant.

Usage

```
al_scale(x, a = 0.05, b = 0.95)
```

Arguments

x	data;
a, b	the probability band, whose tail mass sets the breakdown point.

Details

The sample band length is matched to al_band_model_star, not to al_band_model. Matching to the population band length is inconsistent: the sample quantity converges to S^* rather than to S, so that estimator is biased by a constant factor that does not vanish with the sample size, about five per cent high at the standard normal on the default band. al_scale_raw retains the uncorrected form for comparison.

Value

the estimated scale, NA when the MAD vanishes or the matching equation has no root.

al_scale_raw — *The uncorrected band-matching scale estimator (C back-end)*

Description

Matches the sample band arc length to the population band length al_band_model rather than to al_band_model_star. INCONSISTENT: it estimates the wrong quantity, by a constant factor that does not vanish as the sample grows. Retained only so that the size of the correction can be measured. Use al_scale.

Usage

```
al_scale_raw(x, a = 0.05, b = 0.95)
```

Arguments

x	data;
a, b	the probability band.

Value

the uncorrected scale estimate, NA when the MAD vanishes or there is no root.

---

al_statistic *Arc-length goodness-of-fit statistic (C back-end)*

Description

Arc-length goodness-of-fit statistic (C back-end)

Usage

```
al_statistic(u)
```

Arguments

u	numeric probability-integral transforms in [0,1].

Value

the scalar arc length of the probability-plot ogive.

---

al_test *Arc-length goodness-of-fit test*

Description

Tests H0: F = F0 via the arc length of the probability plot, with the analytic saddlepoint null. Powerful against local density structure (multimodality, clustering, oscillation, heaping) that the empirical-distribution tests miss; weak against smooth location/scale departures.

Usage

```
al_test(x, null = stats::punif, nboot = NULL, rnull = NULL)
```

Arguments

x	data;
null	CDF F0 giving the PIT (default punif);
nboot	optional bootstrap reps.
rnull	generator matching null (needed with nboot).

Value

an object of class "htest".

`arc_generator` *Arc-length generator: transform a bounded-support base into a new distribution*

Description

G(x) = S_F^[a,x] / S_F^[a,b], the normalised cumulative arc length of the base cumulative distribution function, whose density is proportional to sqrt(1 + f(x)^2). Requires bounded support.

Usage

```
arc_generator(dens, lower, upper, ngrid = 4000L)

pgen(q, obj)

dgen(q, obj)

rgen(n, obj)
```

Arguments

`dens`	base density function (vectorised).
`lower`, `upper`	bounded support.
`ngrid`	grid resolution.
`q`	quantiles.
`obj`	an "arcgen" object.
`n`	draws.

Value

an object of class "arcgen".

Functions

- `pgen()`: cumulative distribution function of the generated law
- `dgen()`: density of the generated law
- `rgen()`: random generation from the generated law

arcc_c3max — *Largest admissible shape parameter of the smooth circular family*

Description

Largest admissible shape parameter of the smooth circular family

Usage

```
arcc_c3max()
```

Value

the exact bound 3 sqrt(3) / 5 on |c3|, beyond which the quantile density goes negative.

arcc_exact_ci — *Exact confidence interval for the shape parameter of the wrapped order-three family*

Description

Inverts a Monte Carlo test on a rotation-invariant statistic. The rotation is eliminated exactly by invariance, so no nuisance parameter is profiled; the level is exact for every sample size and every number of reference draws rather than asymptotically, which matters because the admissible boundary is attained and because the likelihood of this family is unbounded. Both statistics are reflection invariants, so the interval brackets `|c3|`; the sign requires a reflection-odd statistic. Recorded resolution is handled by grouping the reference draws as the data are grouped.

Usage

```
arcc_exact_ci(
  theta,
  cgrid = seq(0, arcc_c3max(), length.out = 81),
  B = 999L,
  stat = 0L,
  group = 0L,
  level = 0.1,
  seed = 4207L
)
```

Arguments

theta	observed angles in radians
cgrid	grid over the admissible |c3| interval; the grid is the parameter space, so it is exhaustive rather than a search
B	reference draws per grid point
stat	0 for the first trigonometric moment (about twice as efficient) or 1 for the arc-length spacings functional. Both are reflection invariants.
group	0 for continuous data, or the number of equal cells the data are grouped into (36 for ten-degree grouping)
level	test level, so the interval has coverage 1 - level
seed	seed for the splitmix64 stream, so a run reproduces exactly

Value

list with the interval, the observed statistic and the p-value curve that was inverted

---

arcc_gof *Goodness of fit of a fitted circular arc-length member*

Description

The probability-integral transform through the fitted member sends the sample to uniform under the hypothesis, so the reference law of the arc-length spacings statistic is distribution-free and the Monte Carlo p-value is exact in level for any number of reference draws.

Usage

```
arcc_gof(theta, fit, B = 999L, seed = 4207L)
```

Arguments

theta	angles in radians
fit	a fitted object from fit_arccirc, supplying the member through which the probability-integral transform is taken
B	reference draws
seed	seed for the splitmix64 stream, so a run reproduces exactly

Value

A list of two numbers. stat is the observed arc-length spacings statistic of the angles after the probability-integral transform through the fitted member, and p is its Monte Carlo p-value against the distribution-free reference law. Because the transform sends the sample to uniform under the hypothesis, the p-value is exact in level for any B.

arccirc *The smooth circular arc-length family*

Description

The smooth circular arc-length family

Usage

```
arccirc(c3 = 0, mu = 0)

qd_arccirc(u, obj)

Q_arccirc(u, obj)

darccirc(theta, obj)

trigmom_arccirc(p, obj, nodes = 4096L)

rho_arccirc(obj, nodes = 4096L)

rarccirc(n, obj)
```

Arguments

c3	shape parameter, |c3| <= arcc_c3max(); c3 = 0 is the circular uniform.
mu	mean direction in radians.
u	probabilities in [0,1].
obj	an "arccirc" object.
theta	angles in radians.
p	integer order.
nodes	number of rectangle-rule nodes; the integrand is C^1 periodic, so convergence is geometric and the default is ample.
n	sample size.

Value

an object of class "arccirc".

Functions

- `qd_arccirc()`: quantile density q(u) of the underlying linear family
- `Q_arccirc()`: quantile function Q(u) of the underlying linear family
- `darccirc()`: circular density

- `trigmom_arccirc()`: trigonometric moment of order p
- `rho_arccirc()`: mean resultant length
- `rarccirc()`: random generation

`arccirc_fr` *Fejer-Riesz form of the circular arc-length family*

Description

q(u) = |P(exp(2 pi i u))|^2 normalised to integrate to one. Every such q is non-negative and infinitely differentiable around the circle by construction, so neither an admissibility condition nor a smoothness constraint arises, and the coefficients are unrestricted.

Usage

```
arccirc_fr(p, mu = 0)

qd_arccirc_fr(u, obj)

trigmom_arccirc_fr(pord, obj, nodes = 4096L)

Q_arccirc_fr(u, obj, nodes = 4096L)

darccirc_fr(theta, obj, nodes = 4096L)

rarccirc_fr(n, obj, nodes = 4096L)
```

Arguments

`p`	complex vector of polynomial coefficients.
`mu`	mean direction.
`u`	probabilities in [0,1].
`obj`	an "arccirc_fr" object.
`pord`	integer order.
`nodes`	Simpson cells used to accumulate Q.
`theta`	angles in radians.
`n`	sample size.

Value

an object of class "arccirc_fr".

Functions

- `qd_arccirc_fr()`: quantile density
- `trigmom_arccirc_fr()`: trigonometric moment of order p
- `Q_arccirc_fr()`: quantile function of the underlying linear family
- `darccirc_fr()`: circular density
- `rarccirc_fr()`: random generation

`arcq` *Construct a quantile arc-length distribution*

Description

The quantile density is Q'(u) = sigma * [1 + sum_k coef[k] P_k(u)]_+, with P_k the shifted Legendre polynomials; the quantile function is Q(u) = mu + integral_0^u Q'(v) dv. Support is compact. coef = numeric(0) gives the uniform on [mu, mu+sigma].

Usage

```
arcq(coef = numeric(0), mu = 0, sigma = 1, ngrid = 4000L)

qarcq(p, obj)

parcq(x, obj)

darcq(x, obj)

rarcq(n, obj)

arclength(obj, exact = TRUE, nodes = 24L)
```

Arguments

`coef`	shape coefficients (c_1, c_2, ...); may be empty for the uniform.
`mu`	location (left endpoint of support).
`sigma`	positive scale.
`ngrid`	grid resolution for the internal numerics.
`p`	probabilities.
`obj`	an "arcq" object.
`x`	quantiles.
`n`	number of draws.
`exact`	use Gauss-Legendre quadrature (default) rather than the internal grid.
`nodes`	number of Gauss-Legendre nodes.

Value

an object of class "arcq".

A single number, the arc length of the quantile function.

Functions

- `qarcq()`: quantile function
- `parcq()`: cumulative distribution function
- `darcq()`: density
- `rarcq()`: random generation
- `arclength()`: total arc length of the quantile curve (curve complexity)

  The integrand sqrt(1+Q'(u)^2) is analytic on [0,1], so Gauss-Legendre converges geometrically where the equally spaced grid converges linearly: sixteen nodes reach machine precision on a quadratic quantile density, against a relative error of 3e-04 for four thousand grid points. Set exact = FALSE to recover the old grid value.

  The integral is generally not elementary. With Q' of degree one the integrand is the square root of a quadratic and the antiderivative is an inverse hyperbolic sine; with Q' of degree two it is the square root of a quartic, an elliptic integral, which by Liouville's theorem has no elementary antiderivative; beyond that it is hyperelliptic.

Examples

```
## an empty coefficient vector gives the uniform on [mu, mu + sigma]
arclength(arcq(numeric(0), mu = 0, sigma = 1))

## the quantile function is increasing and the distribution function inverts it
o <- arcq(0.5)
pu <- c(0.1, 0.35, 0.6, 0.9)
max(abs(parcq(qarcq(pu, o), o) - pu))
## Arc length is a SHAPE functional, so shifting the distribution cannot change it.
arclength(arcq(0.5, mu = 0, sigma = 1))
arclength(arcq(0.5, mu = 17, sigma = 1))

## The quadrature route is spectral: the value is settled at a handful of nodes, where
## the equally spaced grid it replaced converges only linearly.
o <- arcq(c(0.5, -0.3))
sapply(c(8L, 16L, 32L), function(k) arclength(o, nodes = k))
```

`bb_evidence` *Bayesian arc-length goodness-of-fit evidence*

Description

The posterior probability that the discrepancy exceeds its `(1 - level)` quantile under the null; values near one are strong evidence against the hypothesised distribution. Sensitive to local density structure (multimodality, clustering, heaping) that the vertical discrepancies miss.

Usage

```
bb_evidence(u, which = "arc", level = 0.95, M = 1500L, ref = NULL, seed = 1L)
```

Arguments

u	numeric probability-integral transforms in [0,1].
which	discrepancy to use, "arc" (default) or "ks".
level	nominal level defining the reference quantile.
M	number of posterior draws.
ref	optional precomputed reference from bb_ref_disc; computed if NULL.
seed	integer seed for the C back-end.

Value

the posterior evidence against the null, in [0,1].

---

bb_post_disc *Bayesian-bootstrap posterior arc-length discrepancies*

Description

Draws M Bayesian-bootstrap (Dirichlet-process, concentration to zero) posterior samples of the arc-length and Kolmogorov–Smirnov discrepancies of the probability-plot ogive for the probability integral transforms u.

Usage

```
bb_post_disc(u, M, seed = 1L)
```

Arguments

u	numeric probability-integral transforms in [0,1].
M	number of posterior draws.
seed	integer seed for the C back-end.

Value

an M by 2 matrix with columns arc and ks.

bb_ref_disc *Null reference distribution of the arc-length discrepancy*

Description

Pools the posterior discrepancies of `D` uniform data sets (`m` draws each) to give the distribution of the discrepancy under the null hypothesis at sample size `n`.

Usage

```
bb_ref_disc(n, D = 300L, m = 16L, seed = 7L)
```

Arguments

`n`	sample size.
`D`	number of uniform reference data sets.
`m`	posterior draws per data set.
`seed`	integer seed for the C back-end.

Value

a matrix with columns `arc` and `ks`.

bc_cdf *Beta-companion distribution function*

Description

The inverse of `bc_q`, obtained in the back-end by a continued-fraction incomplete beta and a safe-guarded root find.

Usage

```
bc_cdf(x, alpha, beta)
```

Arguments

`x`	quantiles at which to evaluate.
`alpha, beta`	the two exponents of the quantile density.

Value

A numeric vector of probabilities, the same length as `x`.

Examples

```
bc_cdf(bc_q(c(0.3, 0.7), -0.60, -0.35), -0.60, -0.35)
```

bc_pdf *Beta-companion density*

Description

The derivative of bc_cdf, equal to the reciprocal of the quantile density evaluated at the corresponding probability.

Usage

```
bc_pdf(x, alpha, beta)
```

Arguments

x	points at which to evaluate the density.
alpha, beta	the two exponents of the quantile density.

Value

A numeric vector of density values, the same length as x.

Examples

```
## the density is the derivative of the distribution function
xx <- bc_q(0.3, -0.60, -0.35); eps <- 1e-6
(bc_cdf(xx + eps, -0.60, -0.35) - bc_cdf(xx - eps, -0.60, -0.35)) / (2 * eps)
bc_pdf(xx, -0.60, -0.35)
```

bc_q *Beta-companion quantile function (standardised support)*

Description

The two-exponent quantile density $q(u) = u^{\alpha}(1-u)^{\beta}$ integrated from the origin, so bc_q is the quantile function of the beta-companion family on its standardised support. Both exponents are negative on the family of interest.

Usage

```
bc_q(u, alpha, beta)
```

Arguments

u	probabilities in $(0, 1)$.
alpha, beta	the two exponents of the quantile density.

Value

A numeric vector of quantiles, the same length as u.

Examples

```
## the quantile function is increasing, and the distribution function inverts it
u <- c(0.05, 0.25, 0.5, 0.75, 0.95)
x <- bc_q(u, alpha = -0.60, beta = -0.35)
all(diff(x) > 0)
max(abs(bc_cdf(x, -0.60, -0.35) - u))
```

---

cf_arclength *Arc length of the empirical characteristic function (C back-end)*

Description

The empirical characteristic function of a sample traces a curve in the complex plane; this returns the arc length of that curve over a window $[0, T]$, doubled for the symmetric half. The sample is standardised internally, so the value is scale free. It is a descriptive, asymmetry-sensitive summary: a symmetric law tends to the value two, and asymmetry adds length.

Usage

```
cf_arclength(x, T = 6, ngrid = 1200L)
```

Arguments

x	numeric data vector.
T	upper end of the frequency window.
ngrid	number of grid points on $[0, T]$.

Value

the windowed arc length of the empirical characteristic function.

`cf_arclength_family` *Closed-form characteristic-function arc length for named families*

Description

Total arc length of the characteristic-function curve for the families with a closed form: `"exponential"` (π), `"gamma"` (shape `k`), `"normal"` (drift ratio `delta`), `"cauchy"` (drift ratio `delta`), `"skewstable"` (index `alpha`, skewness `beta`), and `"poisson"` (rate `lambda`, per period). Total arc length is scale free; a symmetric monotone (Polya) law carries the value two.

Usage

```
cf_arclength_family(
  family = c("exponential", "gamma", "normal", "cauchy", "skewstable", "poisson"),
  k = 1,
  delta = 0,
  alpha = 1.5,
  beta = 0,
  lambda = 1
)
```

Arguments

`family`	one of the names above.
`k`	gamma shape.
`delta`	location-to-scale ratio for the normal and Cauchy.
`alpha`, `beta`	stability index and skewness for the skew-stable family.
`lambda`	Poisson rate (per-period arc length).

Value

the closed-form arc length.

Examples

```
## two closed forms that anchor the whole construction
cf_arclength_family("normal")                    # exactly 2
cf_arclength_family("exponential", lambda = 1)   # exactly pi

## total arc length is scale free, so the exponential rate cannot matter
cf_arclength_family("exponential", lambda = 2)
```

dtemper_vm *Arc-length tempering of a von Mises base*

Description

The generator of the linear theory applied to a circular base, with the scale the construction implicitly carries. Arc length adds a length to a density and is therefore not scale invariant; on a circle of circumference 2π a density is of order $1/(2\pi)$, so without the scale the constant dominates and the transform returns the circular uniform. With it, the family interpolates from the uniform at $s \to 0$ to the base itself as $s \to \infty$, monotonically in concentration, preserving the mode throughout.

Usage

```
dtemper_vm(theta, kappa, mu = 0, s = 1, nodes = 4096L)

trigmom_temper_vm(p, kappa, mu = 0, s = 1, nodes = 4096L)
```

Arguments

theta	angles in radians.
kappa	von Mises concentration of the base.
mu	mean direction.
s	tempering scale.
nodes	quadrature nodes for the normaliser.
p	integer order.

Value

density values at theta.

Functions

- `trigmom_temper_vm()`: trigonometric moment of the tempered law

eq_bstar *Equivalence curve*

Description

Solves `eq_E` for β at a given α by a deterministic scan followed by bisection, so the returned curve is reproducible rather than dependent on a starting value.

Usage

```
eq_bstar(alpha)
```

Arguments

alpha	the first exponent of the quantile density. Documented here rather than inherited, because a combined two-name parameter tag cannot be inherited one name at a time.

Value

A single number, $\beta^*(\alpha)$.

Examples

```
## the solution curve is increasing in alpha, as the global theorem states
als <- seq(-0.62, -0.54, by = 0.02)
bss <- sapply(als, eq_bstar)
all(diff(bss) > 0)
```

---

eq_E *Closed-form equivalence discrepancy*

Description

The signed amount by which the area identity $\int q = u_c q(u_c)$ fails at (α, β). Equivalence holds where it vanishes.

Usage

```
eq_E(alpha, beta)
```

Arguments

alpha, beta	the two exponents of the quantile density.

Value

A single number, the discrepancy.

Examples

```
## the discrepancy vanishes on the equivalence curve and changes sign across it
bs <- eq_bstar(-0.60)
eq_E(-0.60, bs)
eq_E(-0.60, bs - 0.05) * eq_E(-0.60, bs + 0.05) < 0
```

eq_readings *Quantile-domain induction readings and equivalence discrepancy for tilted beta-kernel quantile densities q(u) = u^alpha (1-u)^beta exp(sum theta_j P_j(u))*

Description

Quantile-domain induction readings and equivalence discrepancy for tilted beta-kernel quantile densities q(u) = u^alpha (1-u)^beta exp(sum theta_j P_j(u))

Usage

```
eq_readings(alpha, beta, theta = numeric(0), ngrid = 20001L)

eq_readings_vsl(lambda, delta, ngrid = 20001L)
```

Arguments

alpha, beta	kernel exponents
theta	tilt coefficients on shifted Legendre polynomials (orders 1 to 3)
ngrid	evaluation grid size
lambda, delta	van Staden-Loots parameters (kurtosis and skew weights)

Value

c(D, a, b, u_c, u_b); NA when the geometry is invalid

eq_ub_quad *Quadratic shoulder and mode of the kernel family*

Description

The shoulder is the root of $3q'^2 = qq''$, transcendental on a general family. On this one it reduces to a quadratic $Au^2 + Bu + C = 0$, whose constant term $C = \alpha(2\alpha + 1)$ is positive exactly when $\alpha < -1/2$: that inequality is the family's existence condition for a shoulder.

Usage

```
eq_ub_quad(alpha, beta)
```

Arguments

alpha, beta	the two exponents of the quantile density.

Value

A length-two vector, the shoulder u_b and the mode u_c.

Examples

```
## the returned root satisfies the ORIGINAL transcendental equation, in the log form
## 2g'^2 = g'', not merely the quadratic that replaced it
al <- -0.60; be <- -0.35
ub <- eq_ub_quad(al, be)[1]
gp <- al / ub - be / (1 - ub); gpp <- -al / ub^2 - be / (1 - ub)^2
abs(2 * gp^2 - gpp)
```

---

eqfit_bc — *The equivalence-system paper's fitting and simulation loops, in the back end*

Description

`eqfit_bc` fits the beta-companion sigmoid by deterministic Nelder–Mead from a matrix of starts (free: six columns; constrained to the equivalence curve: five), returning the fitted six-parameter theta, the SSE and both readings. `eqfit_bc_boot` runs the iid residual bootstrap of the constrained fit, warm-started, OpenMP over replicates with per-replicate splitmix64 streams. `eqfit_k4eq` and `eqfit_k4eq_boot` are the kappa-family analogues constrained to a locus table h -> k. `eqfit_msim` runs the complete manifold-test simulation (truth and two displacement arms, with the optimiser audit certificate). `eqfit_estsim` runs the three-estimator study. All outputs are byte-identical across thread counts and between the R and Python fronts.

Usage

```
eqfit_bc(x, y, starts, curve, maxit = 4000L)

eqfit_bc_boot(x, y, th, curve, B = 30L, seed = 4207L, maxit = 2000L)

eqfit_k4eq(x, y, starts, locus, maxit = 4000L)

eqfit_k4eq_boot(x, y, th, p0, locus, B = 25L, seed = 4207L, maxit = 2000L)

eqfit_msim(
  nM,
  sde,
  Rm,
  th0,
  st5,
  curve,
  awin,
  thref,
  seed = 4207L,
```

```
  maxit = 2000L
)

eqfit_score_at(x, y, mu, sg, al, be)

eqfit_blocklen(res)

eqfit_nullT(nM, sde, R2, th0, st5, curve, awin, seed = 4207L, maxit = 2000L)

eqfit_estsim(a0, b0, R, nsizes, seed = 4207L, maxit = 500L)

k4_b_sweep(k, h)

eqfit_taus(a0, b0, R, n, seed = 4207L)

k4_ab_sweep(k, h)

eq_E_sweep(alpha, betas)
```

Arguments

x, y	the curve being fitted
starts	matrix of optimiser starts, one per row
curve	two-column matrix (alpha, beta) of the traced equivalence curve
maxit	Nelder–Mead iteration cap per pass
th	a fitted theta, used to warm-start the bootstrap
B	bootstrap or reference replicates
seed	seed for the per-replicate splitmix64 streams, so a run reproduces exactly
locus	two-column matrix (h, k) of the kappa equivalence locus
p0	warm-start parameter vector for the bootstrap refits
nM, sde, Rm	design size, noise standard deviation and replicates of the manifold study
th0	the true manifold member; st5 the three profiling start transforms (mu, sigma, alpha-centre); thref a 3 x 4 matrix of reference manifold members (mu, sigma, alpha, beta) for the three displaced arms, used for the power certificate; awin the profiling alpha window c(alo, ahi)
st5	the constrained five-parameter start, held on the equivalence curve
awin	two-element admissible window in alpha, passed as lower and upper bounds
thref	three by four matrix of reference thetas, one row per reference member
mu, sg, al, be	a single parameter point for eqfit_score_at
res	a residual vector
a0, b0	beta parameters of the sampled member
R, R2	replicate counts for the estimator simulation and for the null simulation
k, h	shape grid for the wall sweep; betas the beta grid of an E section
n, nsizes	the sample size, and the vector of sample sizes swept over
alpha, betas	a single alpha, and the vector of betas swept across it

Value

All of these return plain R objects rather than a class, so they can be compared element by element against the Python front end. `eqfit_bc` returns a list with `th`, the fitted six-parameter vector, `sse`, the residual sum of squares at that fit, and `a` and `b`, the two curve readings; `eqfit_k4eq` returns the same list without `b`. `eqfit_bc_boot`, `eqfit_k4eq_boot` and `eqfit_nullT` return a numeric vector holding one statistic per replicate, of length `B` for the first two and `R2` for the third. `eqfit_msim` returns a list of eight numeric vectors, each of length `Rm`: the test statistic under the truth (`Tlev`), under the displacement arms (`Taud`, `Tp2`, `Tp15`, `Tp4`) and at the three reference members (`Tref2`, `Tref15`, `Tref4`). `eqfit_estsim` returns a numeric array of dimension `c(R, 3, 2, length(nsizes))`, indexed by replicate, estimator, parameter and sample size. `eqfit_taus` returns a numeric matrix of `R` rows with columns `t3`, `t4`, `al` and `be`, the two L-moment ratios and the parameter pair they map to. `k4_ab_sweep` returns a list of the two numeric vectors `a` and `b`, one entry per element of `k`; `k4_b_sweep` and `eq_E_sweep` return a single numeric vector of the same length as `k` and `betas` respectively. `eqfit_score_at` returns a single number, the score at the given parameter point, and `eqfit_blocklen` a single integer, the selected moving-block length.

`fit_arccirc` *Method-of-moments fit of the smooth circular arc-length family*

Description

The modulus of the first trigonometric moment is even in c3 and strictly increasing in |c3|, so it identifies the magnitude; the rotation-invariant psi = arg(phi_2) - 2 arg(phi_1), reduced to the principal branch, identifies the sign. Reliable sign recovery needs a few thousand observations.

Usage

```
fit_arccirc(theta, nodes = 4096L)
```

Arguments

`theta`	angles in radians.
`nodes`	rectangle-rule nodes used inside the inversion.

Value

a list with mu, c3, the observed |phi_1|, and whether the inversion stayed interior.

fit_arccirc_fr *Closed-form fit of the Fejer-Riesz circular arc-length family*

Description

Sort, apply the circular L-moment weights, invert the exact linear map, factorise: no numerical search enters anywhere. The pair (p, mu) is not identified, because shifting the cut is absorbed exactly by a phase ramp on the coefficients, so the gauge is fixed by cutting at the sample mean direction; compare two fits by the density they imply, not by their coefficients. Following the practice of the linear family, an estimate that leaves the admissible set is reported rather than silently replaced; the shrunken spectrum is returned as a labelled fallback because, unlike the linear case, the factorisation itself fails without one.

Usage

```
fit_arccirc_fr(theta, nm = 3L, margin = 0, nodes = 4096L)
```

Arguments

theta	angles in radians.
nm	number of circular L-moments, equal to the degree of the fitted family.
margin	required minimum of the implied quantile density.
nodes	quadrature nodes.

Value

a list with the coefficients, mean direction, shrink factor and admissibility.

fit_arcq *Fit an arcq distribution to data by matching L-moments*

Description

Matches the first (order+2) L-moments: mu and sigma absorb L_1 and L_2, and the ‘order‘ shape coefficients are chosen to match tau_3, tau_4, ... by least squares.

Usage

```
fit_arcq(x, order = 2L)
```

Arguments

x	data.
order	number of shape coefficients (>=1).

Value

An `arcq` object with the fitted location, scale and shape coefficients.

Examples

```
set.seed(1)
fit <- fit_arcq(rnorm(500), order = 2L)
arclength(fit)
```

`fit_arcq_cf` *Closed-form L-moment fit of the order-two arcq family*

Description

The family's L-moments are an exact linear function of its shape coefficients, so matching them inverts explicitly: c2 = 35 t4 / (3 + 7 t4) and c1 = t3 (5 - c2), with sigma and mu then absorbing the sample L-mean and L-scale. No optimisation, starting value or tolerance is involved, and the estimator is consistent and asymptotically normal by the delta method. This is the estimator of record for the order-two family; 'fit_arcq' remains available for higher orders, where the inversion is done numerically.

Usage

```
fit_arcq_cf(x)
```

Arguments

`x`	data.

Value

an 'arcq' object, with a 'fit' component holding the coefficients, 'mu', 'sigma' and an 'admissible' flag; a warning is issued when the estimate leaves the admissible set.

`k4_band_model` *Band arc lengths of a scaled kappa curve and of a data polyline*

Description

The arc-length element is a norm of the vector (dx, dy). Which norm is a modelling choice: `p = 1` gives the ordinary sum, `p = 2` the Euclidean arc length, and `p = Inf` the max-plus (tropical) form $\max(|dx|, |dy|)$. The three agree to within a factor of $\sqrt{2}$.

Usage

```
k4_band_model(theta, breaks, nodes = 60L, p = 2)

k4_band_sample(x, y, breaks, p = 2)
```

Arguments

theta	c(g0, g1, mu, sg, k, h)
breaks	band break points (length J+1)
nodes	number of Gauss-Legendre nodes per band
p	the norm used for the arc-length element; 2 is the Euclidean default, Inf the tropical.
x, y	data ordered in x

Value

A numeric vector of band arc lengths, one per band.

---

k4_band_trop_mean *Exact mean of the tropical band arc length under Gaussian error*

Description

The tropical (max-plus) arc-length element $\max(dx, |dy|)$ has an expectation elementary in Φ and φ when the curve is observed with independent Gaussian error, whereas the Euclidean element's expectation is a confluent hypergeometric function. The observed band arc length can therefore be compared with its own mean rather than with a clean-curve quantity it does not estimate.

Usage

```
k4_band_trop_mean(x, theta, sigma, breaks)
```

Arguments

x	ordered design points.
theta	the six-vector (g0, g1, mu, sg, k, h).
sigma	error standard deviation of a single observation.
breaks	the J+1 band edges.

Details

Only the mean is returned. The variance of a band sum is not the sum of the elements' variances: consecutive increments share an observation, and although the noise increments correlate at $-1/2$ the absolute value discards the sign and leaves the elements positively correlated, so summing as if independent understates a band's variance by roughly a fifth. That correction is not elementary and is deliberately not supplied.

Value

the J expected band arc lengths.

`k4_fit_lmom` *Deterministic kappa fits: L-moment shape inversion, quantile-domain arc-length shape fit, curve-domain NLS and NALR (banded arc lengths of the running-median presmoothed polyline)*

Description

Deterministic kappa fits: L-moment shape inversion, quantile-domain arc-length shape fit, curve-domain NLS and NALR (banded arc lengths of the running-median presmoothed polyline)

Usage

```
k4_fit_lmom(t3, t4, nodes = 200L)

k4_fit_aleq(y, bands, start = c(0, 0, log(0.3)))

k4_fit_nls(x, y, start, drift = FALSE)

k4_fit_nalr(x, y, start, J = 12L, lambda = 1, w = 9L, p = 2)
```

Arguments

`t3`, `t4`	target sample L-moment ratios
`nodes`	number of Gauss-Legendre nodes for the theoretical ratios
`y`	response vector (sorted internally where required)
`bands`	two-column matrix of quantile bands
`start`	transformed start vector
`x`	data vector
`drift`	model the slow linear rise of the Rancimat water-trap conductivity, `y = g0 + m x + g1 F(x)`. Both induction-period readings are invariant to it, so the equivalence theory is unaffected and only the fit improves.
`J`	number of curve bands
`lambda`	anchor weight
`w`	running-median window
`p`	the norm used for the arc-length element; 2 is the Euclidean default, Inf the tropical

Value

A list of fitted parameters, whose components depend on which fitting routine is called; all return the six-vector `theta` of the scaled kappa curve.

`k4_fit_varpro` *Variable-projection fit of the drifted kappa response*

Description

Fits $y = g_0 + mx + g_1 F(x; \mu, \sigma, k, h)$. The mean curve is linear in the three coefficients, so they are solved exactly for any shape and only the four shape parameters are searched; the multi-start is run in the C back end, in parallel over starts, and the best is chosen serially so the answer does not depend on the thread count.

Usage

```
k4_fit_varpro(
  x,
  y,
  starts,
  maxit = 1500L,
  bounds = c(-0.98, 0.95, 0, 4),
  hfix = NA_real_
)
```

Arguments

`x, y`	the curve.
`starts`	a matrix with four columns, `(mu, log sigma, k, h)`, one row per start; `h` is carried directly so negative values are admissible.
`maxit`	iterations per Nelder-Mead descent.
`bounds`	admissible shape box `c(k_lo, k_hi, h_lo, h_hi)`. The default lower bound on k is the one the edible-oil analysis settled on: a wider box lowers the residual sum of squares but moves the fitted induction periods away from the laboratory values, so the choice is the analyst's and is stated here rather than fixed in the back end. A lower bound of zero on h is admissible and meaningful: it admits the generalised extreme value distribution, the $h \to 0$ member of the family, and with $k = 0$ the Gumbel.
`hfix`	hold h at this value and fit the remaining three parameters, which is how a submodel is fitted in its own right; `NA` (the default) fits all four. Because h is searched on the log scale, $h = 0$ is unreachable by the free search, which creeps toward it and reports a spurious small value instead of naming the submodel. Choosing between the two fits is model selection between a nested pair, and is left to the caller.

Value

a list with `theta` in the six-vector convention `(g0, g1, mu, sg, k, h)`, the fitted `drift`, and the residual sum of squares `rss`.

k4_icc *One-way variance components and the intraclass correlation*

Description

Decomposes the variance of y into within-group and between-group parts under a one-way random-effects model, and returns the intraclass correlation. Groups may be of unequal size; the between-group mean square is divided by the unbalanced constant $k_0 = (N - \sum n_i^2/N)/(G-1)$ rather than by the mean group size, which is the balanced-design shortcut and biases the ratio when group sizes differ.

Usage

```
k4_icc(y, g)
```

Arguments

y	numeric observations.
g	grouping vector; coerced with `factor`, so any labels will do.

Details

A negative between-group variance estimate is truncated at zero, the usual convention.

Value

named numeric vector: `icc`, `sd_within`, `sd_between`.

Examples

```
## replicate runs on the same specimen agree more closely than runs on different specimens
y <- c(1.1, 1.2, 1.0,  5.1, 5.3, 4.9,  9.0, 9.2, 8.8)
g <- rep(1:3, each = 3)
k4_icc(y, g)["icc"] > 0.9
```

k4_locus_dist *Distance from fitted shapes to the equivalence locus*

Description

Shortest Euclidean distance in the (h, k) shape plane from each fitted shape to the equivalence locus, supplied as a polyline. Distance is measured to the segments of the polyline rather than to its vertices: a vertex-only search overstates the distance by up to half the vertex spacing, which matters when the distance is compared against the price of the constraint.

Usage

```
k4_locus_dist(h, k, locus_h, locus_k)
```

Arguments

h, k	numeric vectors of fitted shape parameters, of equal length.
locus_h, locus_k	the locus polyline, of equal length.

Value

numeric vector of distances, NA where the shape is not finite.

Examples

```
## a shape sitting on the locus is at distance zero
lh <- seq(0.05, 0.39, length.out = 20); lk <- seq(-0.04, 0.81, length.out = 20)
k4_locus_dist(lh[5], lk[5], lh, lk) < 1e-12
```

---

k4_mv_boot *The fitting-method multiverse of one induction run: eight admissible pipelines under one moving-block residual bootstrap*

Description

All eight pipelines report the standard tangent reading from the same trace: the variable-projection least-squares fit with its GEV submodel selection (LS), running medians of window nine and twenty-one followed by transformed-parameter NLS (med9-LS, med21-LS), the banded arc-length estimator (arc), the GEV submodel in its own right (GEV-LS), the selection fit past the run's measured transient cutoff (transient-excised), and the selection fit on each half-density index grid (grid-odd, grid-even). Replicates start from the base fit; point estimates use the full start grid. Resampling indices come from per-replicate splitmix64 streams, so the result is identical whatever the OpenMP thread count and identical between the R and Python fronts.

Usage

```
k4_mv_boot(
  x,
  y,
  cut = -Inf,
  B = 50L,
  seed = 4207L,
  bounds = c(-0.98, 0.95, 0, 4),
  maxit = 1500L
)
```

Arguments

`x`, `y`	the logged curve
`cut`	transient cutoff in the units of `x`; `-Inf` excises nothing
`B`	bootstrap replicates; `0` returns point estimates only
`seed`	integer seed for the per-replicate streams
`bounds`	the admissible shape box `(k_lo, k_hi, h_lo, h_hi)`
`maxit`	Nelder–Mead iteration cap per start

Value

`list(a_pt, A)`: the eight named point estimates, and the `B` by eight matrix of replicate readings (`NULL` when `B = 0`)

---

k4_q *Four-parameter kappa quantile, distribution and density functions*

Description

Four-parameter kappa quantile, distribution and density functions

Usage

```
k4_q(u, mu = 0, sg = 1, k, h)

k4_cdf(x, mu = 0, sg = 1, k, h)

k4_pdf(x, mu = 0, sg = 1, k, h)
```

Arguments

`u`, `x`	numeric vectors of probabilities or quantiles
`mu`, `sg`, `k`, `h`	kappa parameters (location, scale, two shapes)

Value

numeric vector

k4_readings *The two standard induction-period readings of a fitted kappa curve*

Description

The two standard induction-period readings of a fitted kappa curve

Usage

```
k4_readings(theta, grid = 4000L)
```

Arguments

theta	c(g0, g1, mu, sg, k, h)
grid	number of grid points for the dense evaluation

Value

c(a = tangent reading, b = third-derivative reading, mode)

k4_runmed *Running median with shrinking symmetric windows at the edges*

Description

Running median with shrinking symmetric windows at the edges

Usage

```
k4_runmed(y, w = 9L)
```

Arguments

y	numeric vector
w	odd window width

Value

A numeric vector the same length as y.

Examples

```
## the window shrinks symmetrically at the ends rather than padding, so the smoothed
## series keeps the length of the original
y <- c(1, 8, 2, 9, 3, 10, 4)
length(k4_runmed(y, 3L)) == length(y)
```

k4_tau34 *Theoretical L-moment ratios of the standard kappa distribution*

Description

Theoretical L-moment ratios of the standard kappa distribution

Usage

```
k4_tau34(k, h, nodes = 200L)
```

Arguments

k, h	shape parameters
nodes	number of Gauss-Legendre nodes

Value

c(tau3, tau4, l1, l2)

lmom_circ *Circular L-moments*

Description

On the line the L-moments are Legendre projections of the quantile function and are linear in the Legendre coefficients of the quantile density. On the circle the natural basis is Fourier, and the m-th circular L-moment is the Fourier projection $\ell_m = \int_0^1 Q(u) e^{-2\pi i m u}\, du$. With the empirical quantile function this is a linear combination of order statistics with fixed complex weights, hence a genuine L-statistic. The map to the quantile-density spectrum is exact and inverts, $\ell_m = i(1-\rho_m)/(2\pi m)$, equivalently $\rho_m = 1 + 2\pi i m \ell_m$.

Usage

```
lmom_circ(x, nm = 3L)

rho_from_lmom(ell)

qmin_rho(rho, nodes = 4096L)

admiss_rho(rho, margin = 0, nodes = 4096L)

factorise_rho(rho, margin = 0, nodes = 4096L)
```

Arguments

x	observations on the unit interval, obtained by cutting the circle at the mean direction.
nm	number of L-moments.
ell	complex vector of circular L-moments.
rho	complex vector of spectrum values.
nodes	grid points.
margin	required minimum of the implied quantile density.

Details

The general device of projecting a quantile function onto an orthogonal basis is not new: it is Sillitto (1969) in one dimension and Decurninge (2014) for multivariate quantile maps. What is specific here is the Fourier form and its exact inversion to the Fejer-Riesz parameters.

Value

complex vector of length nm.

a list with the coefficient vector p, the shrink factor applied to the spectrum (one when nothing was needed), and the admissible flag for the spectrum AS SUPPLIED.

Functions

- rho_from_lmom(): the exact linear inversion to the quantile-density spectrum
- qmin_rho(): smallest implied quantile density; negative means the estimate has left the admissible set
- admiss_rho(): shrink a spectrum towards the circular uniform until admissible
- factorise_rho(): Fejer-Riesz spectral factorisation, recovering the coefficient vector

  A factorisation exists only where the implied quantile density is non-negative, which is what qmin_rho measures; for a single moment with $\rho_0 = 1$ the condition reduces to $|\rho| \leq 1/2$. Asked for an inadmissible spectrum this routine used to run the root finder anyway and return the coefficients of a factorisation that does not exist, silently and at any modulus. It now does what the fitting path in the C back-end has always done: shrink towards the circular uniform until the spectrum is admissible, and REPORT that it did, so a caller can tell an answer to the question asked from an answer to a nearby one.

---

lmoments *Theoretical L-moments of an arcq distribution*

Description

L_r is the shifted-Legendre projection of Q (Hosking's convention): L_r = integral_0^1 Q(u) P*_(r-1)(u) du, r>=1, with P*_r the shifted Legendre polynomial (P*_r(1)=1).

Usage

```
lmoments(obj, nmom = 4L)
```

Arguments

obj	an "arcq" object.
nmom	number of L-moments.

Value

the first nmom L-moments; ratios tau_r = L_r/L_2 for r>=3 via lmratios().

---

lmratios *L-moment ratios (tau_3, tau_4, ...) from a vector of L-moments*

Description

L-moment ratios (tau_3, tau_4, ...) from a vector of L-moments

Usage

```
lmratios(L)
```

Arguments

L	L-moments.

Value

A numeric vector: the first two entries are the location and scale L-moments themselves, followed by the ratios $\tau_3, \tau_4, \ldots$.

Examples

```
## a symmetric sample has vanishing L-skewness
u <- (seq_len(20000) - 0.5) / 20000
lmratios(sample_lmoments(u, 4L))[3]
```

`sample_lmoments` *Sample L-moments of data*

Description

Sample L-moments of data

Usage

```
sample_lmoments(x, nmom = 4L)
```

Arguments

`x`	data.
`nmom`	number of L-moments.

Value

A numeric vector of `nmom` sample L-moments.

Examples

```
## for a uniform grid on (0,1) the first L-moment is 1/2 and the second is 1/6
u <- (seq_len(20000) - 0.5) / 20000
sample_lmoments(u, 4L)[1:2]
```

Index

admiss_rho *(*lmom_circ*)*, 35
al_band_model, 3, *3*, *6*
al_band_model_star, 3, *6*
al_band_sample, 4
al_moments, 5
al_pvalue, 5
al_scale, *3*, 6, *6*
al_scale_raw, 6, *6*
al_statistic, 7
al_test, 7
arc_generator, 8
arcc_c3max, 9
arcc_exact_ci, 9
arcc_gof, 10
arccirc, 11
arccirc_fr, 12
arclength *(*arcq*)*, 13
arcq, 13

bb_evidence, 14
bb_post_disc, 15
bb_ref_disc, 16
bc_cdf, 16, *17*
bc_pdf, 17
bc_q, *16*, 17

cf_arclength, 18
cf_arclength_family, 19

darccirc *(*arccirc*)*, 11
darccirc_fr *(*arccirc_fr*)*, 12
darcq *(*arcq*)*, 13
dgen *(*arc_generator*)*, 8
dtemper_vm, 20

eq_bstar, 20
eq_E, *20*, 21
eq_E_sweep *(*eqfit_bc*)*, 23
eq_readings, 22
eq_readings_vsl *(*eq_readings*)*, 22
eq_ub_quad, 22
eqfit_bc, 23
eqfit_bc_boot *(*eqfit_bc*)*, 23
eqfit_blocklen *(*eqfit_bc*)*, 23
eqfit_estsim *(*eqfit_bc*)*, 23
eqfit_k4eq *(*eqfit_bc*)*, 23
eqfit_k4eq_boot *(*eqfit_bc*)*, 23
eqfit_msim *(*eqfit_bc*)*, 23
eqfit_nullT *(*eqfit_bc*)*, 23
eqfit_score_at *(*eqfit_bc*)*, 23
eqfit_taus *(*eqfit_bc*)*, 23

factorise_rho *(*lmom_circ*)*, 35
fit_arccirc, 25
fit_arccirc_fr, 26
fit_arcq, 26
fit_arcq_cf, 27

k4_ab_sweep *(*eqfit_bc*)*, 23
k4_b_sweep *(*eqfit_bc*)*, 23
k4_band_model, 27
k4_band_sample *(*k4_band_model*)*, 27
k4_band_trop_mean, 28
k4_cdf *(*k4_q*)*, 33
k4_fit_aleq *(*k4_fit_lmom*)*, 29
k4_fit_lmom, 29
k4_fit_nalr *(*k4_fit_lmom*)*, 29
k4_fit_nls *(*k4_fit_lmom*)*, 29
k4_fit_varpro, 30
k4_icc, 31
k4_locus_dist, 31
k4_mv_boot, 32
k4_pdf *(*k4_q*)*, 33
k4_q, 33
k4_readings, 34
k4_runmed, 34
k4_tau34, 35

lmom_circ, 35
lmoments, 36

lmratios, 37

parcq *(arcq)*, 13
pgen *(arc_generator)*, 8

Q_arccirc *(arccirc)*, 11
Q_arccirc_fr *(arccirc_fr)*, 12
qarcq *(arcq)*, 13
qd_arccirc *(arccirc)*, 11
qd_arccirc_fr *(arccirc_fr)*, 12
qmin_rho *(lmom_circ)*, 35

rarccirc *(arccirc)*, 11
rarccirc_fr *(arccirc_fr)*, 12
rarcq *(arcq)*, 13
rgen *(arc_generator)*, 8
rho_arccirc *(arccirc)*, 11
rho_from_lmom *(lmom_circ)*, 35

sample_lmoments, 38

trigmom_arccirc *(arccirc)*, 11
trigmom_arccirc_fr *(arccirc_fr)*, 12
trigmom_temper_vm *(dtemper_vm)*, 20