

Package ‘ldpdist’

October 25, 2026

Title Local Differential Privacy Tools for Distributional Inference

Version 0.1.1

Author Yi Liu [aut, cre],
Qirui Hu [aut]

Maintainer Yi Liu <yliu42@yorku.ca>

Description Estimates cumulative distribution functions and quantiles under local differential privacy. Provides isotonic and divide-and-conquer distribution estimators, aggregation across independent surveys, online inference for single and multiple quantiles, and asymptotic quantile confidence sequences. Also estimates category-specific subdistribution functions under asymmetric local privacy. Binary and categorical randomization mechanisms support these procedures. Distribution estimators operate on privatized reports; quantile interfaces simulate sequential local randomization internally. Published methodological sources include Liu et al. (2023) <https://proceedings.mlr.press/v202/liu23q.html>, Liu, Hu and Kong (2024) <https://proceedings.mlr.press/v235/liu24z.html>, Cai et al. (2025) [doi:10.52202/085713-3818](https://doi.org/10.52202/085713-3818), and Hu and Liu (2026) <https://proceedings.mlr.press/v306/hu26ae.html>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Suggests openssl

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-10-25 14:30:02 UTC

Contents

ldpdist-package	2
cdf-methods	3

ldp_aggregate_surveys	5
ldp_category_range_probability	6
ldp_cdf_variance	7
ldp_confidence_region	8
ldp_dc_critical	9
ldp_design	9
ldp_estimate_category_cdf	10
ldp_estimate_cdf	11
ldp_estimate_quantile	13
ldp_estimate_quantiles	15
ldp_estimate_quantile_cs	17
ldp_gaussian_cs_boundary	18
ldp_mechanism	19
ldp_median_efficiency	20
ldp_msn_critical	21
ldp_pava	22
ldp_privacy_check	22
ldp_randomize	23
ldp_randomize_category	24
ldp_rng_uniform	25
ldp_sn_critical	25
ldp_supported_methods	26
plot.ldpdist	27
Index	29

ldpdist-package	<i>Local differential privacy tools for distributional inference</i>
-----------------	--

Description

Estimation of cumulative distribution functions and quantiles under local differential privacy, with binary and categorical randomization mechanisms and methods for statistical inference.

Details

For distribution functions, `ldp_mechanism()` and `ldp_design()` specify the randomization mechanism and threshold design. `ldp_randomize()` generates privatized reports, which are passed to `ldp_estimate_cdf()` or `ldp_estimate_category_cdf()`.

CDF estimation includes isotonic likelihood estimation (Liu, Hu and Kong, 2024), divide-and-conquer inference, and aggregation across independent surveys. Single-quantile inference uses self-normalization (Liu et al., 2023); multiple-quantile inference uses a categorical reduction and a noncrossing stochastic-approximation recursion. Asymptotic quantile confidence sequences follow Cai et al. (2025). Category-specific subdistribution estimation follows the asymmetric local-privacy construction of Hu and Liu (2026).

The quantile functions simulate sequential local randomization from a numeric observation vector within one R process:

- `ldp_estimate_quantile()` estimates one quantile.
- `ldp_estimate_quantiles()` estimates several quantiles jointly.
- `ldp_estimate_quantile_cs()` constructs an asymptotic confidence sequence.

Randomization uses a secure byte source by default. On Windows this requires the suggested **openssl** package.

Use `rng = "pseudo"` with a seed for reproducible simulations.

References

Liu, Y., Hu, Q., Ding, L. and Kong, L. (2023). Online Local Differential Private Quantile Inference via Self-normalization. *Proceedings of Machine Learning Research*, 202, 21698–21714. <https://proceedings.mlr.press/v202/liu23q.html>

Liu, Y., Hu, Q. and Kong, L. (2024). Tuning-free Estimation and Inference of Cumulative Distribution Function under Local Differential Privacy. *Proceedings of Machine Learning Research*, 235, 31147–31164. <https://proceedings.mlr.press/v235/liu24z.html>

Cai, L., Hu, Q., Sun, J. and Wu, S. (2025). Time-uniform and Asymptotic Confidence Sequence of Quantile under Local Differential Privacy. *Advances in Neural Information Processing Systems*, 38. doi:10.52202/0857133818

Hu, Q. and Liu, Y. (2026). Censoring with Plausible Deniability: Asymmetric Local Privacy for Multi-Category CDF Estimation. *Proceedings of Machine Learning Research*, 306, 45290–45311. <https://proceedings.mlr.press/v306/hu26ae.html>

cdf-methods

Evaluate and invert a fitted CDF

Description

Extract fitted step heights, evaluate the right-continuous step function, or obtain quantiles and inverted pointwise limits on the fitted threshold grid.

Usage

```
## S3 method for class 'ldp_cdf_fit'
as.function(x, ...)
## S3 method for class 'ldp_cdf_fit'
coef(object, ...)
## S3 method for class 'ldp_cdf_fit'
fitted(object, ...)
## S3 method for class 'ldp_cdf_fit'
quantile(x, probs = seq(0, 1, 0.25),
  names = TRUE, bounds = NULL, ...)
ldp_quantile_interval(object, probs = c(0.25, 0.5, 0.75),
  level = 0.95, bounds = NULL, variance = c("observed", "planned"),
  critical = c("normal", "dc_finite_block"))
```

Arguments

<code>x, object</code>	An <code>ldp_cdf_fit</code> object.
<code>probs</code>	Numeric probabilities in $[0, 1]$, without missing values.
<code>names</code>	Whether to label the returned quantiles by percentage.
<code>bounds</code>	Optional finite public lower and upper bounds enclosing all fitted thresholds. See Details for the endpoint convention.
<code>level</code>	Nominal level passed to <code>confint()</code> .
<code>variance</code>	Observed counts or planned allocation for CDF variance.
<code>critical</code>	CDF calibration method or a positive finite critical value.
<code>...</code>	Unused. For <code>quantile()</code> , extra arguments raise an error.

Details

`coef()` and `fitted()` return the step heights, named by threshold; these are not regression slopes. `as.function()` returns a callable stepfun, not an `ecdf`: the object estimates a distribution from private reports rather than retaining an empirical sample. At a fitted threshold its value is the fitted height; between thresholds it uses the preceding height. Outside the fitted range it extends the nearest height, as does `predict()`; it does not impose tails of zero and one.

`quantile()` returns the first fitted threshold with height at least the requested probability. This is a generalized inverse on the finite fitted grid, not an interpolated sample quantile. A flat region is resolved at its first crossing. Without bounds, an absent crossing is NA, and probability zero returns the first threshold. With bounds, an absent crossing returns the upper bound and probability zero returns the lower bound. The fitted CDF is not changed.

`ldp_quantile_interval()` crosses the upper CDF limit for the lower quantile limit and the lower CDF limit for the upper quantile limit. Without bounds, an absent crossing is NA. With bounds, an absent lower-limit crossing returns the lower bound, and an absent upper-limit crossing returns the upper bound. Thus unresolved limits are reported conservatively at the declared public endpoints. Probability zero returns the lower bound in all columns when bounds are supplied. CDF limits need not be monotone; each first crossing is computed directly, without imposing an additional shape constraint.

These are inversions of nominal pointwise plug-in limits, not a simultaneous confidence band or a general quantile coverage guarantee. Their interpretation depends on the fixed-grid approximation and the assumptions of the CDF confidence procedure. Supplying bounds does not add a coverage guarantee.

Value

`as.function()` returns a function of class `stepfun`.

`coef()` and `fitted()` return named numeric vectors.

`quantile()` returns a numeric vector of length `length(probs)`.

`ldp_quantile_interval()` returns a matrix with columns `estimate`, `lower`, and `upper`, with one row per probability.

See Also

[ldp_estimate_cdf](#), [stepfun](#)

Examples

```
mech <- ldp_mechanism("binary_rr", epsilon = 4)
design <- ldp_design("cdf", grid = seq(0, 1, length.out = 21))
reports <- ldp_randomize(seq(0, 1, length.out = 1000), mech, design,
                        rng = "pseudo", seed = 12)
fit <- ldp_estimate_cdf(reports)
Fhat <- as.function(fit)
Fhat(c(0.25, 0.5, 0.75))
head(coef(fit))
quantile(fit, probs = c(0.25, 0.5, 0.75))
ldp_quantile_interval(fit, probs = c(0.5, 0.9), bounds = c(0, 1),
                     variance = "planned")
```

ldp_aggregate_surveys *Aggregate CDF estimates from heterogeneous surveys*

Description

Aggregates independent survey-level CDF estimates. The theory-motivated rule uses weights proportional to $n_s^{2/3} g_s(x_0)^{2/3}$, where n_s is the survey size and $g_s(x_0)$ is the threshold design density at the target point.

Usage

```
ldp_aggregate_surveys(
  fits,
  x0,
  weights = c("equal", "theory", "sample_size", "custom"),
  design_density = NULL,
  custom_weights = NULL
)
```

Arguments

<code>fits</code>	A nonempty list of <code>ldp_cdf_fit</code> objects.
<code>x0</code>	Point at which to aggregate CDF estimates.
<code>weights</code>	Weighting rule.
<code>design_density</code>	Threshold design densities at <code>x0</code> . Required and strictly positive for <code>weights = "theory"</code> .
<code>custom_weights</code>	User-supplied nonnegative weights for <code>weights = "custom"</code> .

Value

An object of class `ldp_survey_aggregate`.

Examples

```

set.seed(1)
mech <- ldp_mechanism("binary_rr", epsilon = 2)
d <- ldp_design("cdf", grid = seq(0, 1, length.out = 11))
r1 <- ldp_randomize(stats::runif(100), mech, d, rng = "pseudo", seed = 1)
r2 <- ldp_randomize(stats::runif(100), mech, d, rng = "pseudo", seed = 2)
f1 <- ldp_estimate_cdf(r1)
f2 <- ldp_estimate_cdf(r2)
ldp_aggregate_surveys(list(f1, f2), x0 = 0.5)

```

ldp_category_range_probability

Estimate category probabilities over a value range

Description

Computes category-specific probability mass over (lower, upper] from fitted category sub-CDFs. With conditional = TRUE, the output estimates category probabilities conditional on the value falling in the requested range. An error is returned when the fitted range mass is zero.

Usage

```

ldp_category_range_probability(
  object, lower = NULL, upper = NULL, conditional = TRUE
)

```

Arguments

object	An ldp_category_cdf_fit object.
lower, upper	Finite range endpoints within the fitted threshold range. NULL selects the corresponding fitted endpoint.
conditional	Whether to normalize category masses by the total recovered mass in the range.

Value

A data frame with category probabilities. Attributes include range_total and conditional.

Examples

```

set.seed(1)
x <- stats::runif(100)
y <- sample(c("A", "B"), 100, replace = TRUE)
design <- ldp_design("category_cdf", grid = seq(0, 1, length.out = 11),
  categories = c("A", "B"), y = y)
mech <- ldp_mechanism("uldp_censor", epsilon = 2, domain_size = 2)
reports <- ldp_randomize(x, mech, design, rng = "pseudo", seed = 1)
fit <- ldp_estimate_category_cdf(reports)
ldp_category_range_probability(fit, lower = 0.25, upper = 0.75)

```

ldp_cdf_variance	<i>Finite-grid variance for LDP CDF estimates</i>
------------------	---

Description

Computes the finite-grid pointwise variance induced by binary randomized response and the public threshold design. For isotonic CDF fits, the formula uses the affine randomized-response inverse and aggregates either observed or planned threshold sample sizes over active PAVA pools. Planned allocation is unavailable when report thresholds override the design. For divide-and-conquer fits, it returns the block-studentized variance used by `confint()`.

Usage

```
ldp_cdf_variance(
  object,
  newdata = NULL,
  source = c("observed", "planned"),
  as_matrix = FALSE
)
```

Arguments

<code>object</code>	An <code>ldp_cdf_fit</code> object.
<code>newdata</code>	Optional points at which to map the finite-grid variance.
<code>source</code>	Whether to use observed threshold counts or planned design allocation probabilities.
<code>as_matrix</code>	Whether to return a covariance matrix. Cells in the same active PAVA pool have a shared nonzero covariance.

Value

A named numeric vector, or a covariance matrix when `as_matrix = TRUE`.

Examples

```
x <- stats::runif(200)
mech <- ldp_mechanism("binary_rr", epsilon = 2)
design <- ldp_design("cdf", grid = seq(0, 1, length.out = 11))
reports <- ldp_randomize(x, mech, design, rng = "pseudo", seed = 3)
fit <- ldp_estimate_cdf(reports)
ldp_cdf_variance(fit, newdata = c(0.25, 0.75))
```

ldp_confidence_region *Self-normalized confidence region for many quantiles*

Description

Builds a self-normalized ellipsoidal confidence region from a many-quantile fit. The returned object contains the center, shape matrix, critical value, and helper functions for evaluating the statistic and checking membership.

Usage

```
ldp_confidence_region(
  object,
  level = 0.95,
  critical = NULL,
  n_sim = 5000L,
  n_grid = 400L,
  seed = 1L
)
```

Arguments

object	An ldp_quantiles_fit object.
level	Confidence level.
critical	Optional numeric critical value.
n_sim, n_grid, seed	Simulation controls when critical is not supplied.

Value

An object of class ldp_confidence_region.

Examples

```
set.seed(1)
x <- stats::rnorm(300)
fit <- ldp_estimate_quantiles(x, c(0.25, 0.5, 0.75),
                             epsilon = 2, bounds = c(-4, 4),
                             rng = "pseudo", seed = 2)
region <- ldp_confidence_region(fit, level = 0.8, n_sim = 200, n_grid = 50)
region$contains(coef(fit))
```

ldp_dc_critical	<i>Finite-block critical values for D&C CDF inference</i>
-----------------	---

Description

Returns finite-block critical values used by the divide-and-conquer CDF studentized interval. Values are interpolated on the log block-count scale from the 2026 D&C CDF supplementary implementation.

Usage

```
ldp_dc_critical(blocks, level = 0.95)
```

Arguments

blocks	Number of divide-and-conquer blocks.
level	Confidence level. Currently tabulated finite-block values are available for 95 percent intervals.

Value

A numeric vector of critical values.

Examples

```
ldp_dc_critical(c(8, 32, 128))
```

ldp_design	<i>Create an LDP data-collection design</i>
------------	---

Description

Creates a design object describing the public queries used during local data collection. The design is separate from the privacy mechanism so that the same estimator can be paired with different randomizers or threshold schemes.

Usage

```
ldp_design(
  task = c("cdf", "quantile", "quantiles", "category_cdf"),
  grid = NULL,
  tau = NULL,
  bounds = NULL,
  categories = NULL,
  assignment = c("random", "cyclic"),
  weights = NULL,
  ...
)
```

Arguments

<code>task</code>	Statistical task.
<code>grid</code>	Threshold grid for CDF designs. The grid is stored in increasing order.
<code>tau</code>	Quantile levels for quantile designs.
<code>bounds</code>	Optional finite support or projection bounds.
<code>categories</code>	Optional category labels. Category-CDF labels may be character, factor-like, or numeric values; reports store their internal 0-based indices.
<code>assignment</code>	How thresholds are assigned to users.
<code>weights</code>	Optional threshold sampling probabilities. If <code>grid</code> is not supplied in increasing order, weights are reordered with the grid. The length must match <code>grid</code> before normalization.
<code>...</code>	Reserved for task-specific options.

Value

An object of class `ldp_design`.

Examples

```
design <- ldp_design("cdf", grid = seq(0, 1, length.out = 11))
design
```

`ldp_estimate_category_cdf`

Estimate category-specific CDFs from censored ULDP reports

Description

Recovers category-specific sub-CDFs from asymmetrically censored randomized response reports. The default implementation uses componentwise marginal isotonic regression. The optional projected procedure uses Dykstra projection onto the shared feasible set and is not presented as an exact ICM likelihood maximizer.

Usage

```
ldp_estimate_category_cdf(
  reports,
  method = c("marginal", "projected", "global_icm"),
  cap_total = TRUE,
  max_iter = 100L,
  tol = 1e-07
)
## S3 method for class 'ldp_category_cdf_fit'
predict(object, newdata = NULL, ...)
```

Arguments

reports	An ldp_reports object from a category_cdf design and an uldp_censor mechanism.
method	Estimation method. "marginal" fits each revealed-category indicator by isotonic regression. "projected" applies a likelihood-increasing projected optimizer as numerical regularization. "global_icm" is a deprecated alias for "projected".
cap_total	Whether to cap the total recovered CDF at one by freezing sub-CDFs after the first crossing for "marginal", or by enforcing the corresponding distorted-CDF cap for "projected".
max_iter	Maximum number of projected iterations; must be a positive integer.
tol	Convergence tolerance for the projected updates.
object	A fitted category CDF object.
newdata	Points at which to evaluate the fitted sub-CDFs.
...	Unused.

Value

An object of class `ldp_category_cdf_fit`; `predict()` returns a data frame containing the total CDF and one sub-CDF column per category. Prediction is restricted to the fitted threshold range. Projected fits also store convergence diagnostics in `diagnostics`.

Examples

```
set.seed(1)
x <- stats::runif(100)
y <- sample(c("A", "B"), 100, replace = TRUE)
design <- ldp_design("category_cdf", grid = seq(0, 1, length.out = 11),
  categories = c("A", "B"), y = y)
mech <- ldp_mechanism("uldp_censor", epsilon = 2, domain_size = 2)
reports <- ldp_randomize(x, mech, design, rng = "pseudo", seed = 1)
fit <- ldp_estimate_category_cdf(reports)
predict(fit, c(0.25, 0.5))
projected <- ldp_estimate_category_cdf(reports, method = "projected")
```

<code>ldp_estimate_cdf</code>	<i>Estimate a CDF from local threshold reports</i>
-------------------------------	--

Description

Fits the monotone current-status CDF estimator from binary randomized response threshold reports. The implementation applies weighted isotonic regression to the privatized threshold means, then maps the fitted curve back to the original CDF scale and clips it to $[0, 1]$. The divide-and-conquer method fits the same estimator independently over blocks and averages the fitted values. `vcov()`, `ldp_cdf_variance()`, and `confint()` use finite-grid covariance formulas that account for randomized-response inflation, active PAVA pools, and public threshold allocation. The normal limits use the realized active pools as fixed and do not adjust for active-set selection or simultaneous coverage.

Usage

```
ldp_estimate_cdf(
  reports, grid = NULL, method = c("isotonic", "dc"), blocks = NULL
)
## S3 method for class 'ldp_cdf_fit'
confint(
  object,
  parm,
  level = 0.95,
  critical = c("normal", "dc_finite_block"),
  newdata = NULL,
  variance = c("observed", "planned"),
  ...
)
## S3 method for class 'ldp_cdf_fit'
predict(object, newdata = NULL, ...)
## S3 method for class 'ldp_cdf_fit'
summary(object, ...)
## S3 method for class 'ldp_cdf_fit'
vcov(object, ...)
```

Arguments

reports	An ldp_reports object from a CDF design.
grid	Optional prediction grid.
method	Estimator type. "isotonic" fits one monotone estimator; "dc" averages monotone estimates over disjoint blocks and requires untied continuous thresholds supplied through threshold=.
blocks	Number of blocks for method = "dc". The supported range starts at four and does not exceed $\text{floor}(n^{0.2})$.
object	An ldp_cdf_fit object.
parm	Unused.
level	Nominal confidence level for pointwise plug-in Wald limits.
newdata	Numeric points at which to evaluate the fitted CDF or confidence interval. For predict(), a data frame with a numeric column named x is also accepted.
critical	Critical value method for confint(): normal or finite-block D&C calibration.
variance	Use observed threshold counts or planned allocation to compute the finite-grid variance of an isotonic fit.
...	Unused.

Value

An object of class ldp_cdf_fit; predict() returns a data frame.

See Also

[cdf-methods](#) for function conversion, fitted-value extraction, and grid quantiles.

Examples

```

set.seed(1)
x <- stats::runif(200)
mech <- ldp_mechanism("binary_rr", epsilon = 2)
design <- ldp_design("cdf", grid = seq(0, 1, length.out = 21))
reports <- ldp_randomize(x, mech, design, rng = "pseudo", seed = 2)
fit <- ldp_estimate_cdf(reports)
predict(fit, c(0.25, 0.5, 0.75))
predict(fit, data.frame(x = c(0.25, 0.5, 0.75)))

```

ldp_estimate_quantile *Estimate one quantile with online binary LDP queries*

Description

Runs the online binary-response quantile recursion and returns the Polyak-Ruppert averaged estimate. The fit stores the self-normalizer needed for inference without estimating the density at the target quantile. The stored normalizer is not a covariance estimate, so `vcov()` returns an informative error.

Usage

```

ldp_estimate_quantile(
  x,
  tau = 0.5,
  mechanism = ldp_mechanism("binary_rr", epsilon),
  epsilon,
  init = 0,
  step_scale = 1,
  step_power = 0.6,
  rng = c("secure", "pseudo"),
  seed = NULL,
  keep_path = FALSE
)
## S3 method for class 'ldp_quantile_fit'
coef(object, ...)
## S3 method for class 'ldp_quantile_fit'
fitted(object, ...)
## S3 method for class 'ldp_quantile_fit'
predict(object, newdata = NULL, ...)
## S3 method for class 'ldp_quantile_fit'
vcov(object, ...)
## S3 method for class 'ldp_quantile_fit'
confint(
  object,
  parm,
  level = 0.95,

```

```

critical = "self_normalized",
n_sim = 20000L,
n_grid = 1000L,
seed = 1L,
...
)
## S3 method for class 'ldp_quantile_fit'
summary(object, ...)

```

Arguments

<code>x</code>	Numeric observations used to simulate sequential local randomization.
<code>tau</code>	Target quantile level.
<code>mechanism</code>	A binary randomized-response mechanism.
<code>epsilon</code>	Privacy budget used when mechanism is omitted.
<code>init</code>	Initial query value.
<code>step_scale, step_power</code>	Step-size parameters for $\text{step_scale} * t^{(-\text{step_power})}$; the supported exponent lies in $(0.5, 1)$.
<code>rng</code>	Randomness source.
<code>seed</code>	Optional pseudo-random seed for simulation.
<code>keep_path</code>	Whether to retain iterate and averaged paths.
<code>object</code>	An <code>ldp_quantile_fit</code> object.
<code>newdata</code>	Must be NULL. This estimator has no covariates and does not predict conditional quantiles for new observations.
<code>parm</code>	Unused.
<code>level</code>	Confidence level.
<code>critical</code>	The self-normalized method or a user-supplied numeric critical value.
<code>n_sim, n_grid</code>	Simulation controls for the self-normalized critical value.
<code>...</code>	Unused.

Details

This function simulates the sequential protocol in one R process. It receives the observation vector and performs local randomization internally.

The self-normalized inference assumes independent observations and local randomization, a unique target quantile with positive continuous density, and the regularity and stability conditions of the online quantile procedure. For a continuous CDF at the target, the quantile solves $F(q) = \tau$. With discrete data the generalized quantile need not solve this equation; computing a fit does not establish the applicability of the inferential limit.

Value

An object of class `ldp_quantile_fit`. `coef()`, `fitted()`, and `predict()` return the same named scalar: the estimated unconditional quantile on the original measurement scale.

Examples

```

set.seed(1)
x <- stats::rnorm(500)
fit <- ldp_estimate_quantile(x, tau = 0.5, epsilon = 2,
                           init = 0, rng = "pseudo", seed = 10)
coef(fit)
confint(fit)

```

ldp_estimate_quantiles

Jointly estimate multiple quantiles under categorical LDP

Description

Reduces the many-quantile problem to KRR interval-label estimation. At each step the current quantile vector partitions the real line, the user's interval label is privatized by KRR, and the server uses the unbiased one-hot decoder in a projected Robbins-Monro update. The projection enforces noncrossing at every iteration. This function is a simulation wrapper that receives the private numeric vector and performs the local randomization internally. Joint inference is provided by `ldp_confidence_region()`; `confint()` returns an informative error rather than unsupported marginal intervals.

Usage

```

ldp_estimate_quantiles(
  x,
  tau,
  mechanism = ldp_mechanism("krr", epsilon, domain_size = length(tau) + 1L),
  epsilon,
  bounds,
  init = NULL,
  step_scale = NULL,
  step_power = 0.6,
  rng = c("secure", "pseudo"),
  seed = NULL,
  keep_path = FALSE
)
## S3 method for class 'ldp_quantiles_fit'
coef(object, ...)
## S3 method for class 'ldp_quantiles_fit'
confint(object, parm, level = 0.95, simultaneous = TRUE, ...)
## S3 method for class 'ldp_quantiles_fit'
summary(object, ...)

```

ldp_estimate_quantile_cs

Asymptotic confidence sequence for one LDP quantile

Description

Runs a dynamic-chain parallel SGD procedure for one LDP quantile and combines the chain-wise variance estimator with a Gaussian confidence-sequence boundary.

Usage

```
ldp_estimate_quantile_cs(
  x,
  tau = 0.5,
  mechanism = ldp_mechanism("binary_rr", epsilon),
  epsilon,
  init = 0,
  initial_chains = 5L,
  growth = "log",
  log_base = 10,
  step_scale = 1,
  step_power = 0.6,
  alpha = 0.05,
  boundary = c("stitched", "mixture"),
  rho = 0.01,
  rng = c("secure", "pseudo"),
  seed = NULL,
  checkpoints = NULL
)
## S3 method for class 'ldp_quantile_cs'
confint(object, parm, level = 1 - object$alpha, ...)
## S3 method for class 'ldp_quantile_cs'
summary(object, ...)
```

Arguments

x	Private numeric observations.
tau	Target quantile level.
mechanism	A binary randomized-response mechanism.
epsilon	Privacy budget used when mechanism is omitted.
init	Initial value for each chain.
initial_chains	Initial number of parallel chains.
growth	The supported logarithmic chain-growth schedule.
log_base	Base for logarithmic chain growth; it must exceed $\max(\text{initial_chains}, 2 + 1 / \text{initial_chains})$.

step_scale, step_power	Per-chain step-size controls, with exponent in $(0.5, 1)$.
alpha	Miscoverage level.
boundary	Gaussian confidence-sequence boundary.
rho	Mixture-boundary tuning parameter.
rng	Randomness source.
seed	Optional pseudo-random seed for simulation.
checkpoints	Optional integer times at which to store the sequence.
object	An ldp_quantile_cs object.
parm, level, ...	Standard confint() arguments.

Details

This function simulates the sequential protocol in one R process. It receives the observation vector and performs local randomization internally.

Inference assumes independent observations and local randomization, a unique target quantile with positive continuous density, a bounded density derivative, and the supported step-size and chain-growth schedules. The guarantee is asymptotic, not finite-sample time-uniform coverage.

For chain counts n_j , iterate sums S_j , and the overall averaged estimate $\hat{q}_t$, the scale is the weighted variance of $(S_j - n_j \hat{q}_t) / \sqrt{n_j}$, with weights n_j / t . Centering before scaling makes the result invariant to a common translation of the data and initialization, including while a new chain catches up.

Value

An object of class ldp_quantile_cs.

Examples

```
set.seed(1)
x <- stats::rnorm(500)
cs <- ldp_estimate_quantile_cs(x, tau = 0.5, epsilon = 2,
                             rng = "pseudo", seed = 13)
confint(cs)
```

ldp_gaussian_cs_boundary

Gaussian confidence-sequence boundaries

Description

Computes Gaussian boundaries used by the asymptotic confidence-sequence construction.

Usage

```
ldp_gaussian_cs_boundary(
  time,
  alpha = 0.05,
  type = c("stitched", "mixture"),
  m = 1,
  rho = 0.01
)
```

Arguments

time	Positive time index or vector of time indices.
alpha	Miscoverage level.
type	Boundary family.
m	Starting time for stitched boundaries.
rho	Mixture-boundary tuning parameter.

Value

A numeric vector of boundary widths on the unit-variance mean scale.

Examples

```
ldp_gaussian_cs_boundary(c(100, 1000), alpha = 0.05)
```

ldp_mechanism	<i>Create a local privacy mechanism object</i>
---------------	--

Description

Constructs an ldp_mechanism object with validated privacy and domain parameters.

Usage

```
ldp_mechanism(
  type = c("binary_rr", "krr", "oue", "olh", "hadamard", "uldp_censor"),
  epsilon,
  domain_size = NULL,
  ...
)
```

Arguments

type	Mechanism family.
epsilon	Positive local privacy budget.
domain_size	Number of categories for categorical mechanisms. Hadamard response currently supports padded size up to 4096.
...	Reserved for mechanism-specific options.

Value

An object of class `ldp_mechanism`.

Examples

```
mech <- ldp_mechanism("binary_rr", epsilon = 1)
mech
```

`ldp_median_efficiency` *Median efficiency diagnostic under binary randomized response*

Description

Computes the median asymptotic variance lower bound for binary randomized-response sequential LDP mechanisms, $1 / (4 * r^2 * n * f(m)^2)$, where $r = \tanh(\epsilon / 2)$ and $f(m)$ is the density at the median. The diagnostic reports the privacy variance inflation relative to the non-private median bound and, when an estimator variance is supplied, the ratio of the lower bound to that variance.

Usage

```
ldp_median_efficiency(
  object = NULL,
  density_at_median,
  epsilon = NULL,
  n = NULL,
  estimator_variance = NULL
)
```

Arguments

<code>object</code>	Optional <code>ldp_quantile_fit</code> object with $\tau = 0.5$.
<code>density_at_median</code>	Value of the population density at the median.
<code>epsilon</code>	Privacy budget. Required when <code>object</code> is omitted.
<code>n</code>	Sample size. Required when <code>object</code> is omitted.
<code>estimator_variance</code>	Optional variance to compare with the lower bound. When omitted, the function returns the bounds and privacy inflation without an efficiency ratio. Supplying <code>object</code> does not supply a variance.

Value

An object of class `ldp_median_efficiency`.

Examples

```
ldp_median_efficiency(density_at_median = stats::dnorm(0),
                      epsilon = 2, n = 1000)
x <- stats::rnorm(500)
fit <- ldp_estimate_quantile(x, tau = 0.5, epsilon = 2,
                           rng = "pseudo", seed = 1)
ldp_median_efficiency(fit, density_at_median = stats::dnorm(0))
```

ldp_msn_critical	<i>Multivariate self-normalized Brownian critical values</i>
------------------	--

Description

Approximates the critical value of the multivariate self-normalized Brownian statistic used by the many-quantile confidence region.

Usage

```
ldp_msn_critical(k, level = 0.95, n_sim = 5000L, n_grid = 400L, seed = 1L)
```

Arguments

k	Dimension of the Brownian motion.
level	Quantile level.
n_sim	Number of simulated Brownian paths.
n_grid	Discretization grid size.
seed	Pseudo-random seed. A supplied integer seed is used locally and the previous random state is restored afterwards. With NULL, simulation uses and advances the current random state. Fractional or missing seeds are rejected.

Value

A numeric critical value.

Examples

```
ldp_msn_critical(2, level = 0.8, n_sim = 200, n_grid = 50)
```

ldp_pava

*Weighted pool-adjacent-violators algorithm***Description**

Computes the weighted isotonic regression fit using the pool-adjacent-violators algorithm. This is an internal workhorse exposed because many LDP distributional estimators require monotone projections.

Usage

```
ldp_pava(y, weights = NULL, decreasing = FALSE, backend = c("auto", "C", "R"))
```

Arguments

<code>y</code>	Numeric vector to be projected onto the monotone cone.
<code>weights</code>	Optional nonnegative weights.
<code>decreasing</code>	Whether to fit a decreasing sequence.
<code>backend</code>	Computation backend. "auto" uses the compiled C backend when available and falls back to R; "C" requires the compiled backend; "R" uses the pure R implementation.

Value

A numeric vector of fitted monotone values.

Examples

```
ldp_pava(c(0.1, 0.4, 0.3, 0.8))
ldp_pava(c(0.1, 0.4, 0.3, 0.8), backend = "R")
```

ldp_privacy_check

*Check analytic likelihood-ratio bounds for LDP mechanisms***Description**

Reports the analytic likelihood-ratio bound associated with a mechanism. For ordinary LDP mechanisms this checks the symmetric epsilon-LDP bound. For the censored ULDP mechanism, the report explicitly states that the guarantee is asymmetric and applies to the protected censored output rather than to all original-value pairs.

Usage

```
ldp_privacy_check(mechanism)
```

Arguments

mechanism An ldp_mechanism object.

Value

An object of class ldp_privacy_check.

Examples

```
ldp_privacy_check(ldp_mechanism("binary_rr", epsilon = 1))
ldp_privacy_check(ldp_mechanism("uldp_censor", epsilon = 1, domain_size = 2))
```

ldp_randomize	<i>Locally randomize private observations</i>
---------------	---

Description

Applies a local randomizer to private data. The current implementation supports threshold CDF reports with binary randomized response and censored category-CDF reports with the ULDP censoring mechanism.

Usage

```
ldp_randomize(
  x,
  mechanism,
  design,
  rng = c("secure", "pseudo"),
  seed = NULL,
  threshold = NULL
)
```

Arguments

x Private user observations.

mechanism An ldp_mechanism object.

design An ldp_design object.

rng Randomness source.

seed Optional pseudo-random seed for simulations.

threshold Optional preassigned threshold vector for CDF designs.

Value

An object of class ldp_reports.

Examples

```
x <- c(0.2, 0.6, 0.9)
mech <- ldp_mechanism("binary_rr", epsilon = 2)
design <- ldp_design("cdf", grid = seq(0, 1, by = 0.25), assignment = "cyclic")
ldp_randomize(x, mech, design, rng = "pseudo", seed = 1)
```

ldp_randomize_category

Categorical local randomizers and unbiased decoders

Description

Applies categorical LDP mechanisms and decodes the resulting reports into unbiased one-hot estimates. Implemented mechanisms are direct randomized response ("krr"), optimized unary encoding ("oue"), optimized local hashing ("olh"), and Hadamard response ("hadamard").

Usage

```
ldp_randomize_category(
  category, mechanism, rng = c("secure", "pseudo"), seed = NULL
)
ldp_decode_category(reports, mechanism = reports$mechanism)
```

Arguments

category	Integer vector of 0-based category labels.
mechanism	Categorical mechanism created by ldp_mechanism().
rng	Randomness source.
seed	Optional pseudo-random seed for simulation.
reports	Categorical privatized reports.

Value

ldp_randomize_category() returns an ldp_category_reports object.

ldp_decode_category() returns a numeric matrix with one row per report and one column per category.

Examples

```
cat <- c(0, 1, 2, 1)
mech <- ldp_mechanism("krr", epsilon = 2, domain_size = 3)
reports <- ldp_randomize_category(cat, mech, rng = "pseudo", seed = 1)
ldp_decode_category(reports)
```

ldp_rng_uniform	<i>Generate uniform random numbers for LDP workflows</i>
-----------------	--

Description

Generates uniforms using either a cryptographically secure byte source or R's pseudo-random generator. Secure generation uses **openssl** when available and otherwise `/dev/urandom` on Unix-like systems.

Usage

```
ldp_rng_uniform(n, rng = c("secure", "pseudo"), seed = NULL)
```

Arguments

n	Nonnegative integer number of uniforms.
rng	Randomness source. Use "secure" for local randomization in real deployments and "pseudo" for reproducible simulation.
seed	Optional integer seed for rng = "pseudo". Secure generation rejects seeds.

Value

A numeric vector in $[\emptyset, 1)$.

Examples

```
ldp_rng_uniform(3, rng = "pseudo", seed = 1)
```

ldp_sn_critical	<i>Simulate a scalar self-normalized Brownian critical value</i>
-----------------	--

Description

Approximates the quantile of the scalar self-normalized Brownian statistic used by the online quantile confidence interval.

Usage

```
ldp_sn_critical(level = 0.95, n_sim = 20000L, n_grid = 1000L, seed = 1L)
```

Arguments

level	Quantile level.
n_sim	Number of Monte Carlo Brownian paths.
n_grid	Discretization grid size for Brownian paths.
seed	Pseudo-random seed. A supplied integer seed is used locally and the previous random state is restored afterwards. With NULL, simulation uses and advances the current random state. Fractional or missing seeds are rejected.

Value

A numeric critical value.

Examples

```
ldp_sn_critical(level = 0.8, n_sim = 200, n_grid = 50)
```

ldp_supported_methods *List registered mechanisms, tasks, and estimators*

Description

Returns the mechanisms, task families, estimators, and public functions available in **ldpdist**.

Usage

```
ldp_supported_methods()
```

Value

A named list of character vectors.

Examples

```
ldp_supported_methods()
```

plot.ldpdist	<i>Plot ldpdist fitted objects</i>
--------------	------------------------------------

Description

Base graphics plotting methods for fitted distributions and quantiles.

The CDF method uses `plot.stepfun()`, marking the attained value at each threshold. Its ... arguments include `xlim`, `col.hor`, `col.vert`, `col.points`, and `cex.points`.

The scalar-quantile method plots the averaged-iterate path when the fit was created with `keep_path = TRUE`; otherwise it plots the final estimate. An optional interval is shown only at the final time. It is not a confidence sequence for the plotted path.

Usage

```
## S3 method for class 'ldp_cdf_fit'
plot(x, ..., conf.int = FALSE, level = 0.95,
     xlab = "x", ylab = "CDF", main = "LDP CDF estimate", ylim = c(0, 1),
     add = FALSE, verticals = TRUE, do.points = TRUE, pch = 19,
     variance = c("observed", "planned"))
## S3 method for class 'ldp_quantile_fit'
plot(x, ..., conf.int = FALSE, level = 0.95,
     critical = "self_normalized", n_sim = 20000L, n_grid = 1000L, seed = 1L,
     xlab = "Time", ylab = "Quantile", main = "LDP quantile estimate",
     ylim = NULL, add = FALSE)
## S3 method for class 'ldp_category_cdf_fit'
plot(x, ..., xlab = "x", ylab = "Sub-CDF",
     main = "LDP category CDF estimates")
## S3 method for class 'ldp_quantile_cs'
plot(x, ..., xlab = "Time", ylab = "Quantile",
     main = "LDP quantile confidence sequence")
```

Arguments

<code>x</code>	A fitted object from ldpdist .
<code>...</code>	Additional graphics arguments.
<code>conf.int</code>	Whether to add pointwise CDF limits or a final-time scalar quantile interval.
<code>level</code>	Confidence level.
<code>xlab, ylab, main</code>	Base graphics labels.
<code>ylim</code>	Vertical plotting limits.
<code>add</code>	Whether to add the estimate to an existing plot.
<code>verticals, do.points, pch</code>	Draw vertical steps and points at the fitted thresholds, with the specified point symbol; passed to <code>plot.stepfun()</code> .
<code>variance</code>	CDF variance source.
<code>critical, n_sim, n_grid, seed</code>	Scalar-quantile confidence calibration, passed to <code>confint()</code> .

Value

The input object, invisibly.

Examples

```
qfit <- ldp_estimate_quantile(seq(-1, 1, length.out = 200), epsilon = 2,  
                             rng = "pseudo", seed = 8, keep_path = TRUE)  
plot(qfit)
```

Index

- * **package**
 - ldpdist-package, 2
- as.function.ldp_cdf_fit (cdf-methods), 3
- cdf-methods, 3
- coef.ldp_cdf_fit (cdf-methods), 3
- coef.ldp_quantile_fit
 - (ldp_estimate_quantile), 13
- coef.ldp_quantiles_fit
 - (ldp_estimate_quantiles), 15
- confint.ldp_cdf_fit (ldp_estimate_cdf), 11
- confint.ldp_quantile_cs
 - (ldp_estimate_quantile_cs), 17
- confint.ldp_quantile_fit
 - (ldp_estimate_quantile), 13
- confint.ldp_quantiles_fit
 - (ldp_estimate_quantiles), 15
- fitted.ldp_cdf_fit (cdf-methods), 3
- fitted.ldp_quantile_fit
 - (ldp_estimate_quantile), 13
- ldp_aggregate_surveys, 5
- ldp_category_range_probability, 6
- ldp_cdf_variance, 7
- ldp_confidence_region, 8
- ldp_dc_critical, 9
- ldp_decode_category
 - (ldp_randomize_category), 24
- ldp_design, 9
- ldp_estimate_category_cdf, 10
- ldp_estimate_cdf, 5, 11
- ldp_estimate_quantile, 13
- ldp_estimate_quantile_cs, 17
- ldp_estimate_quantiles, 15
- ldp_gaussian_cs_boundary, 18
- ldp_mechanism, 19
- ldp_median_efficiency, 20
- ldp_msn_critical, 21
- ldp_pava, 22
- ldp_privacy_check, 22
- ldp_quantile_interval (cdf-methods), 3
- ldp_randomize, 23
- ldp_randomize_category, 24
- ldp_rng_uniform, 25
- ldp_sn_critical, 25
- ldp_supported_methods, 26
- ldpdist (ldpdist-package), 2
- ldpdist-package, 2
- plot.ldp_category_cdf_fit
 - (plot.ldpdist), 27
- plot.ldp_cdf_fit (plot.ldpdist), 27
- plot.ldp_quantile_cs (plot.ldpdist), 27
- plot.ldp_quantile_fit (plot.ldpdist), 27
- plot.ldpdist, 27
- predict.ldp_category_cdf_fit
 - (ldp_estimate_category_cdf), 10
- predict.ldp_cdf_fit (ldp_estimate_cdf), 11
- predict.ldp_quantile_fit
 - (ldp_estimate_quantile), 13
- quantile.ldp_cdf_fit (cdf-methods), 3
- stepfun, 5
- summary.ldp_cdf_fit (ldp_estimate_cdf), 11
- summary.ldp_quantile_cs
 - (ldp_estimate_quantile_cs), 17
- summary.ldp_quantile_fit
 - (ldp_estimate_quantile), 13
- summary.ldp_quantiles_fit
 - (ldp_estimate_quantiles), 15
- vcov.ldp_cdf_fit (ldp_estimate_cdf), 11
- vcov.ldp_quantile_fit
 - (ldp_estimate_quantile), 13