

Package ‘ldpdist’

October 13, 2026

Title Local Differential Privacy Tools for Distributional Inference

Version 0.1.0

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 binary and categorical randomization mechanisms, isotonic distribution estimators, and simulation interfaces for sequential quantile estimation with self-normalized inference and asymptotic confidence sequences. Distribution estimators operate on privatized reports; quantile interfaces simulate local randomization internally. Methods include the quantile procedure of Liu et al. (2023) <<https://proceedings.mlr.press/v202/liu23q.html>> and the distribution estimator of Liu, Hu and Kong (2024) <<https://proceedings.mlr.press/v235/liu24z.html>>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Suggests openssl

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-10-13 13:30:02 UTC

Contents

ldpdist-package	2
ldp_aggregate_surveys	3
ldp_category_range_probability	4
ldp_cdf_variance	4
ldp_confidence_region	5
ldp_dc_critical	6

ldp_design	7
ldp_estimate_category_cdf	8
ldp_estimate_cdf	9
ldp_estimate_quantile	10
ldp_estimate_quantiles	12
ldp_estimate_quantile_cs	14
ldp_gaussian_cs_boundary	16
ldp_mechanism	17
ldp_median_efficiency	17
ldp_msn_critical	18
ldp_pava	19
ldp_privacy_check	20
ldp_randomize	20
ldp_randomize_category	21
ldp_rng_uniform	22
ldp_sn_critical	22
ldp_supported_methods	23
plot.ldpdist	24
Index	25

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().

The quantile functions ldp_estimate_quantile(), ldp_estimate_quantiles(), and ldp_estimate_quantile_cs() are simulation interfaces: they receive a numeric data vector and carry out the sequential local randomization internally. They do not implement a distributed client-server service.

Randomization uses a secure byte source by default. On Windows this requires the suggested **openssl** package. Seeded pseudo-random generation is available for reproducible simulations through rng = "pseudo".

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>

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>	Positive threshold design density values at <code>x0</code> , required 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 $(\text{lower}, \text{upper}]$ 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

<code>object</code>	An <code>ldp_category_cdf_fit</code> object.
<code>lower, upper</code>	Finite range endpoints within the fitted threshold range. <code>NULL</code> selects the corresponding fitted endpoint.
<code>conditional</code>	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`
Finite-grid variance for LDP CDF estimates

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

Create an LDP data-collection design

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

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

```

ldp_estimate_cdf

*Estimate a CDF from local threshold reports***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	Points at which to evaluate the fitted CDF or confidence interval.
critical	Critical value method for confint(): normal or finite-block D&C calibration.
variance	Variance source for finite-grid isotonic fits: observed threshold counts or planned design allocation.
...	Unused.

Value

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

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))
```

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'
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>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`.

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, ...)
```

Arguments

<code>x</code>	Private numeric observations.
<code>tau</code>	Strictly increasing vector of target quantile levels.
<code>mechanism</code>	KRR mechanism with domain size $\text{length}(\text{tau}) + 1$.
<code>epsilon</code>	Privacy budget used when mechanism is omitted.
<code>bounds</code>	Known projection bounds for the target quantiles.
<code>init</code>	Optional initial noncrossing quantile vector.
<code>step_scale, step_power</code>	Step-size controls. When <code>step_scale</code> is <code>NULL</code> , the default is $\text{diff}(\text{bounds}) / (\text{length}(\text{tau}) + 1)$, which matches the initial interval width induced by the quantile partition. 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 average paths.
<code>object</code>	An <code>ldp_quantiles_fit</code> object.
<code>parm</code>	Unused.
<code>level</code>	Confidence level.
<code>simultaneous</code>	Unused compatibility argument. Marginal Gaussian intervals are not supplied by the multivariate self-normalization theory.
<code>...</code>	Unused.

Details

Joint self-normalized inference assumes independent observations and local randomization, unique target quantiles with positive continuous densities, and the regularity and stability conditions of the projected recursion. The true quantile vector must be strictly ordered and lie inside the supplied bounds. Numerical positive definiteness of the observed self-normalizer is also required to construct the region.

Value

An object of class `ldp_quantiles_fit`.

Examples

```
set.seed(1)
x <- stats::rnorm(300)
mech <- ldp_mechanism("krr", epsilon = 2, domain_size = 4)
fit <- ldp_estimate_quantiles(x, tau = c(0.25, 0.5, 0.75),
                             mechanism = mech, bounds = c(-4, 4),
                             rng = "pseudo", seed = 2)
coef(fit)
```

`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

<code>x</code>	Private numeric observations.
<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 value for each chain.
<code>initial_chains</code>	Initial number of parallel chains.
<code>growth</code>	The supported logarithmic chain-growth schedule.
<code>log_base</code>	Base for logarithmic chain growth; it must exceed $\max(\text{initial_chains}, 2 + 1 / \text{initial_chains})$.
<code>step_scale, step_power</code>	Per-chain step-size controls, with exponent in $(0.5, 1)$.
<code>alpha</code>	Miscoverage level.
<code>boundary</code>	Gaussian confidence-sequence boundary.
<code>rho</code>	Mixture-boundary tuning parameter.
<code>rng</code>	Randomness source.
<code>seed</code>	Optional pseudo-random seed for simulation.
<code>checkpoints</code>	Optional integer times at which to store the sequence.
<code>object</code>	An <code>ldp_quantile_cs</code> object.
<code>parm, level, ...</code>	Standard <code>confint()</code> 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	Create a local privacy mechanism object
---------------	---

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

object Optional ldp_quantile_fit object with $\tau = 0.5$.

density_at_median Value of the population density at the median.

epsilon Privacy budget. Required when object is omitted.

n Sample size. Required when object is omitted.

estimator_variance Optional variance to compare with the lower bound. When omitted, the function returns the bounds and privacy inflation without an efficiency ratio. Supplying object 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

Multivariate self-normalized Brownian critical values

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

<code>k</code>	Dimension of the Brownian motion.
<code>level</code>	Quantile level.
<code>n_sim</code>	Number of simulated Brownian paths.
<code>n_grid</code>	Discretization grid size.
<code>seed</code>	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)
```

<code>ldp_pava</code>	<i>Weighted pool-adjacent-violators algorithm</i>
-----------------------	---

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	<i>Check analytic likelihood-ratio bounds for LDP mechanisms</i>
-------------------	--

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

<code>x</code>	Private user observations.
<code>mechanism</code>	An <code>ldp_mechanism</code> object.
<code>design</code>	An <code>ldp_design</code> object.
<code>rng</code>	Randomness source.
<code>seed</code>	Optional pseudo-random seed for simulations.
<code>threshold</code>	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

<code>category</code>	Integer vector of 0-based category labels.
<code>mechanism</code>	Categorical mechanism created by <code>ldp_mechanism()</code> .
<code>rng</code>	Randomness source.
<code>seed</code>	Optional pseudo-random seed for simulation.
<code>reports</code>	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("kr", 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

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

Value

A numeric vector in $[0, 1)$.

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 CDF fits, category CDF fits, and quantile confidence sequences.

Usage

```
## S3 method for class 'ldp_cdf_fit'
plot(x, ..., conf.int = FALSE, level = 0.95,
     xlab = "x", ylab = "CDF", main = "LDP CDF estimate")
## 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 confidence bands for CDF fits.
<code>level</code>	Confidence level.
<code>xlab, ylab, main</code>	Base graphics labels.

Value

The input object, invisibly.

Index

- * **package**
 - ldpdist-package, 2
- coef.ldap_quantile_fit
 - (ldap_estimate_quantile), 10
- coef.ldap_quantiles_fit
 - (ldap_estimate_quantiles), 12
- confint.ldap_cdf_fit(ldap_estimate_cdf), 9
- confint.ldap_quantile_cs
 - (ldap_estimate_quantile_cs), 14
- confint.ldap_quantile_fit
 - (ldap_estimate_quantile), 10
- confint.ldap_quantiles_fit
 - (ldap_estimate_quantiles), 12
- ldap_aggregate_surveys, 3
- ldap_category_range_probability, 4
- ldap_cdf_variance, 4
- ldap_confidence_region, 5
- ldap_dc_critical, 6
- ldap_decode_category
 - (ldap_randomize_category), 21
- ldap_design, 7
- ldap_estimate_category_cdf, 8
- ldap_estimate_cdf, 9
- ldap_estimate_quantile, 10
- ldap_estimate_quantile_cs, 14
- ldap_estimate_quantiles, 12
- ldap_gaussian_cs_boundary, 16
- ldap_mechanism, 17
- ldap_median_efficiency, 17
- ldap_msn_critical, 18
- ldap_pava, 19
- ldap_privacy_check, 20
- ldap_randomize, 20
- ldap_randomize_category, 21
- ldap_rng_uniform, 22
- ldap_sn_critical, 22
- ldap_supported_methods, 23
- ldpdist(ldpdist-package), 2
- ldpdist-package, 2
- plot.ldap_category_cdf_fit
 - (plot.ldpdist), 24
- plot.ldap_cdf_fit(plot.ldpdist), 24
- plot.ldap_quantile_cs(plot.ldpdist), 24
- plot.ldpdist, 24
- predict.ldap_category_cdf_fit
 - (ldap_estimate_category_cdf), 8
- predict.ldap_cdf_fit(ldap_estimate_cdf), 9
- summary.ldap_cdf_fit(ldap_estimate_cdf), 9
- summary.ldap_quantile_cs
 - (ldap_estimate_quantile_cs), 14
- summary.ldap_quantile_fit
 - (ldap_estimate_quantile), 10
- summary.ldap_quantiles_fit
 - (ldap_estimate_quantiles), 12
- vcov.ldap_cdf_fit(ldap_estimate_cdf), 9
- vcov.ldap_quantile_fit
 - (ldap_estimate_quantile), 10