Package: dominoise (via r-universe)

October 15, 2026

Title Parsimonious Noise Perturbation for Magnitude Tables

Version 0.2.0

Description Protects magnitude tables, such as tables of enterprise turnover, by multiplying each total by a Gaussian noise combining a dominance-driven component and a general-purpose one, governed by three parameters only. Information loss and disclosure risks for three attack scenarios (external inference, internal inference and differencing) are available in closed form, so that calibration requires neither simulation nor re-estimation on the data. The package guides that calibration step by step, from the differencing noise to the dominance parameters, with decision tables and risk-utility plots, applies the mechanism to an aggregated table using SHA-512 hashed random keys for consistency over time, and reports the risk and utility actually achieved. The method is described in Jamme (2027) <doi:10.1007/978-3-032-37883-5_9>.

License GPL (>= 3)

URL https://github.com/InseeFrLab/dominoise, https://inseefrlab.github.io/dominoise/

BugReports https://github.com/InseeFrLab/dominoise/issues

Encoding UTF-8

Language en-GB

Depends R (>= 4.1)

Imports stats, utils, digest, assertthat, rlang

Suggests ggplot2 (>= 3.5.0), testthat (>= 3.0.0)

RoxygenNote 8.0.0

Config/testthat/edition 3

NeedsCompilation no

Author Julien Jamme [aut, cre] (ORCID: <https://orcid.org/0009-0009-9054-7053>), Institut national de la statistique et des études économiques [cph]

Maintainer Julien Jamme <julien.jamme@insee.fr>

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

Date/Publication 2026-10-15 16:20:02 UTC

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

RemoteRef HEAD

RemoteSha c981cb3fceba0829f334a78571696dbd677607c9

Contents

assess_loss_ci . . . 2
assess_loss_expectation . . . 3
assess_risk_diff . . . 4
assess_risk_empirical . . . 5
assess_risk_I . . . 6
assess_risk_II . . . 6
assess_utility_empirical . . . 7
plot.pm_calib_diff . . . 8
plot.pm_calib_dominance . . . 9
pm_calib_diff . . . 10
pm_calib_dominance . . . 10
pm_commit_diff . . . 12
pm_commit_dominance . . . 13
pm_draws . . . 14
pm_params . . . 15
pm_perturb . . . 16
pm_plot_impact . . . 17
pm_plot_risk_max . . . 18
pm_plot_risk_profile . . . 19
pm_plot_tradeoff . . . 20
pm_sigma_eps . . . 21
pm_suggest_n . . . 22
summary.pm_calib_dominance . . . 23

Index **25**

`assess_loss_ci` *Confidence-interval bound of the relative loss Z*

Description

Upper bound `L(rho) = q * sqrt(rho^(2n) sigma_nu^2 + sigma_eps^2)` of the level-`level` confidence interval `[-L(rho), L(rho)]` of Z, with `q` the `(1+level)/2` quantile of the standard normal. The interval is symmetric, so this single value gives both bounds.

Usage

```
assess_loss_ci(rho, sigma_nu, sigma_eps, n, level = 0.95)
```

Arguments

`rho`	Dominance share X1/Y in (0,1]. Recycled.
`sigma_nu`, `sigma_eps`, `n`	Mechanism parameters. Recycled.
`level`	Confidence level (default 0.95).

Value

The upper bound L(rho) as a relative quantity.

See Also

dominoise-package paper's Definition 4

Examples

```
assess_loss_ci(rho = 1, sigma_nu = 0.3, sigma_eps = 0.03, n = 6)
```

`assess_loss_expectation` *Conditional expectation of the absolute relative loss* |Z|

Description

`E(|Z| | P = rho) = sqrt(2 (rho^(2n) sigma_nu^2 + sigma_eps^2) / pi)`: the average relative perturbation undergone by a cell of dominance `rho`. Contrary to the differencing step – where letting rho -> 0 makes the sigma_nu term vanish – the sigma_nu contribution is kept in full here.

Usage

```
assess_loss_expectation(rho, sigma_nu, sigma_eps, n)
```

Arguments

`rho`	Dominance share X1/Y in (0,1]. Recycled.
`sigma_nu`, `sigma_eps`, `n`	Mechanism parameters. Recycled.

Value

E(|Z| | rho) as a relative quantity (multiply by 100 for percent).

See Also

dominoise-package paper's proposition 3

Examples

```
assess_loss_expectation(rho = 1, sigma_nu = 0.3, sigma_eps = 0.03, n = 6)
```

---

assess_risk_diff *Compute the upper bound of the differencing risk.*

Description

Compute the upper bound of the differencing risk.

Usage

```
assess_risk_diff(sigma_eps, beta)
```

Arguments

sigma_eps	double
beta	double

Value

A numeric vector: the upper bound of the differencing risk, vectorised over `sigma_eps` and `beta`.

See Also

dominoise-package paper's proposition 7

Examples

```
assess_risk_diff(sigma_eps = 0.1, beta = 0.2)
```

assess_risk_empirical *Observed disclosure risk*

Description

For each cell, evaluates whether the attack of a given scenario would in fact have succeeded on the realised perturbation – scenario I: `|Y' - X1| / X1 < beta`; scenario II: `|(Y' - X2) - X1| / X1 < beta` – and compares the observed success rate with the theoretical probability `mu(rho)` averaged over the table. The two should agree: the realised rate is one draw from the probability the mechanism guarantees.

Usage

```
assess_risk_empirical(x, scenario = c("I", "II"), beta = NULL, by = NULL)
```

Arguments

`x`	A table returned by `pm_perturb()`.
`scenario`	`"I"` (default) or `"II"`.
`beta`	Accuracy threshold. Defaults to the matching policy of the parameter object.
`by`	Optional column name(s) to break the summary down by.

Details

Scenario II requires the `x2` column to have been declared in `pm_perturb()`, which adds the `rho2` share the theoretical measure needs. The differencing scenario is not assessed here: it bears on pairs of cells, not on single cells, and is controlled a priori by the upper bound of the metric.

Value

A `data.frame` with the number of cells, the observed success rate (in percent), the mean theoretical risk, and the maximum theoretical risk.

See Also

dominoise-package paper's proposition 7

Examples

```
set.seed(123)
para <- suppressMessages(
  pm_commit_dominance(pm_commit_diff(pm_params()), sigma_nu = 0.4, n = 4)
)
tab <- data.frame(turnover = runif(500, 100, 1000), ck = runif(500))
tab$x1 <- tab$turnover * runif(500, 0.2, 1)
tab$x2 <- (tab$turnover - tab$x1) * runif(500)
res <- pm_perturb(tab, "turnover", "x1", "ck", para, x2 = "x2")
assess_risk_empirical(res, scenario = "I")
assess_risk_empirical(res, scenario = "II")
```

assess_risk_I — *Scenario-I (dominance) risk measure*

Description

Probability that an external attacker infers the dominant contribution X1 to within +/- beta : `mu_I(rho) = F_Z((1+beta)rho` with Z the relative loss at dominance rho.

Usage

```
assess_risk_I(rho, sigma_nu, sigma_eps, n, beta)
```

Arguments

`rho`	Dominance share X1/Y in (0,1]. Recycled.
`sigma_nu`, `sigma_eps`, `n`	Mechanism parameters. Recycled.
`beta`	Accuracy threshold of the inference.

Value

mu_I(rho), vectorised.

See Also

dominoise-package paper's proposition 5

Examples

```
assess_risk_I(rho = 0.85, sigma_nu = 0.3, sigma_eps = 0.03, n = 6, beta = 0.2)
```

assess_risk_II — *Scenario-II (p%-rule) risk measure*

Description

Probability that an insider contributing `X2` to the aggregate infers the dominant contribution `X1` to within +/- beta, by subtracting their own contribution from the disseminated total : `mu_II(rho, rho2) = F_Z((1+beta)rho -` with `Z` the relative loss at dominance `rho`.

Usage

```
assess_risk_II(rho, rho2, sigma_nu, sigma_eps, n, beta)
```

Arguments

`rho`	Share of the largest contribution, X1/Y, in (0,1]. Recycled.
`rho2`	Share of the second contribution, X2/Y. Recycled.
`sigma_nu`, `sigma_eps`, `n`	Mechanism parameters. Recycled.
`beta`	Accuracy threshold of the inference.

Details

The worst case (IIa) has the two largest contributors sharing the whole cell (`rho + rho2 = 1`); the relaxed case (IIb) leaves a residual share to the other contributors, typically `rho + rho2 = 0.95`.

Value

mu_II(rho, rho2), vectorised.

See Also

dominoise-package paper's proposition 5

Examples

```
# worst case IIa at rho = 0.5
assess_risk_II(0.5, 0.5, sigma_nu = 0.4, sigma_eps = 0.031, n = 4, beta = 0.1)
```

---

assess_utility_empirical

Observed information loss

Description

Measures the perturbation actually undergone by the table, and compares it with the theoretical expectation evaluated at each cell's own dominance. A close match is the natural consistency check: the mechanism is analytical, so the realised loss should track the predicted one.

Usage

```
assess_utility_empirical(x, by = NULL, thresholds = c(5, 10, 20))
```

Arguments

`x`	A table returned by `pm_perturb()`.
`by`	Optional column name(s) to break the summary down by (e.g. a publication stratum, or a dominance band built beforehand).
`thresholds`	Relative deviations (in percent) whose exceedance rate is reported.

Value

A `data.frame`: number of cells, mean and median absolute relative deviation, quantiles, maximum, relative RMSE, exceedance rates, and the theoretical mean absolute loss averaged over the observed dominance.

See Also

dominoise-package paper's proposition 3

Examples

```
set.seed(123)
para <- suppressMessages(
  pm_commit_dominance(pm_commit_diff(pm_params()), sigma_nu = 0.4, n = 4)
)
tab <- data.frame(turnover = runif(500, 100, 1000), ck = runif(500))
tab$x1 <- tab$turnover * runif(500, 0.2, 1)
tab$x2 <- (tab$turnover - tab$x1) * runif(500)
res <- pm_perturb(tab, "turnover", "x1", "ck", para, x2 = "x2")
assess_utility_empirical(res)
```

`plot.pm_calib_diff` *Plot method for a differencing calibration grid*

Description

Draws the risk-utility frontier of the differencing step: the level risk ceiling `tau` (guaranteed maximum differencing risk) against the relative loss (Z) level it costs (either expectation of |Z| or upper bound of the confidence interval of it), one curve per `beta`.

Usage

```
## S3 method for class 'pm_calib_diff'
plot(x, loss = c("CI", "EZ"), ...)
```

Arguments

`x`	A `pm_calib_diff` grid, as returned by `pm_calib_diff()`.
`loss`	One of `"CI"` (for upper bound of the confidence interval), `"EZ"` (for expectation of the absolute relative loss |Z|): which loss metric to use on the x-axis.
`...`	Ignored, for compatibility with the `plot()` generic.

Value

A `ggplot` object.

Examples

```
plot(pm_calib_diff())
```

---

plot.pm_calib_dominance

Plot method for a dominance calibration grid

Description

Thin dispatcher over the three calibration plots.

Usage

```
## S3 method for class 'pm_calib_dominance'
plot(x, type = c("tradeoff", "profile", "worst"), ...)
```

Arguments

x	A pm_calib_dominance table.
type	"tradeoff" (Figure 4, default), "profile" (Figure 2) or "worst" (Figure 3, middle panel).
...	Passed on to the underlying plotting function.

Value

A ggplot object.

Examples

```
grid <- pm_calib_dominance(sigma_eps = 0.031, beta = 0.2)
plot(grid)
plot(grid, type = "profile")
```

pm_calib_diff — *Calibration of the differencing noise (sigma_epsilon)*

Description

Calibration of the differencing noise (sigma_epsilon)

Usage

```
pm_calib_diff(params = NULL, beta = NULL, tau = NULL, level = 0.95)
```

Arguments

params	object pm_params
beta	real vector of beta values (accuracy level of the inference)
tau	real vector of tau values (ceiling risk level)
level	confidence level for the confidence interval of the relative loss

Details

By default, the function returns the results for some beta and tau values beta in (0.05, 0.1, 0.15, 0.2, 0.25) and tau in (0.5, 0.55, 0.6, ..., 0.90, 0.95)

Value

A data.frame of class pm_calib_diff, one row per (beta, tau) couple, with columns beta, tau, sigma_eps, and the loss floor it implies at rho -> 0: EZ (mean absolute loss) and CI (upper bound of the level confidence interval), both in percent. The confidence level is stored in the level attribute.

Examples

```
# default behaviour
pm_calib_diff()
```

pm_calib_dominance — *Calibration of the dominance noise (sigma_nu and n)*

Description

Theoretical, data-free. For every candidate (sigma_nu, n, beta, sigma_eps), evaluates the scenario-I risk over an internal grid of dominance levels rho, keeps its worst case (max over rho), and reports the maximum information loss (reached at rho = 1, hence set by sigma_nu alone). The returned table is the decision surface of step 2: feed it to plot() for the risk heat-map, to pm_suggest_n() for the largest admissible n, then lock the choice with pm_commit_dominance().

Usage

```
pm_calib_dominance(
  params = NULL,
  sigma_nu = NULL,
  n = NULL,
  beta = NULL,
  sigma_eps = NULL,
  level = 0.95,
  rho = NULL
)
```

Arguments

params	Optional pm_params. When supplied, sigma_eps and beta default to its values.
sigma_nu, n	Numeric vectors of candidate values. Default to (1:5) / 10 and seq(3, 12, by = 3).
beta	Accuracy threshold(s) of scenario I. Default to the dominance policy of params, or to c(0.1, 0.2).
sigma_eps	Fixed differencing noise. Default to params$mechanism$sigma_eps, or 0 when no params is given.
level	Confidence level for the CI loss metric (default 0.95).
rho	Internal grid of dominance levels used to locate the worst case. Default (1:50) / 50.

Details

Like pm_calib_diff(), calling it with everything at NULL returns a default exploration grid.

Value

A data.frame of class pm_calib_dominance, one row per (rho, sigma_nu, sigma_eps, n, beta), with columns risk (scenario-I risk), EZ and CI (information loss, in percent). The confidence level is stored in the level attribute. Use summary() for the worst case over rho of each parameter combination.

Examples

```
# default exploration
grid <- pm_calib_dominance()
grid
summary(grid)

# at a committed sigma_eps
para <- pm_commit_diff(pm_params())
grid <- pm_calib_dominance(para)
```

pm_commit_diff *Commit the differencing decision into the parameter object*

Description

Once beta_diff and tau_diff are chosen (typically after exploring with pm_calib_diff()), this fills sigma_eps in the pm_params object and records the two policy values. It is the single place where step 1 mutates params.

Usage

```
pm_commit_diff(
  params,
  beta = params$policy$diff$beta,
  tau = params$policy$diff$tau,
  level = 0.95
)
```

Arguments

params	A pm_params object.
beta, tau	The chosen threshold and ceiling. Default to the values already stored in the diff policy. Must be single values.
level	Confidence level used only for the informative message.

Value

The updated pm_params (returned invisibly).

Examples

```
parameters <- pm_params()

# beta and tau already set in the pm_params object:
parameters <- pm_commit_diff(parameters)
parameters

# if an another choice is preferred (after calibration search)
parameters <- pm_commit_diff(parameters, beta = 0.1, tau = 0.9)
parameters
```

`pm_commit_dominance` *Commit the dominance decision (sigma_nu and n) into the parameter object*

Description

Closes calibration step 2: records the chosen `sigma_nu` and `n`, together with the scenario-I policy (`beta`, `tau`), into the `pm_params` object. This is the single place where step 2 mutates `params`.

Usage

```
pm_commit_dominance(
  params,
  sigma_nu,
  n,
  beta = NULL,
  tau = NULL,
  level = 0.95,
  rho = NULL
)
```

Arguments

`params`	A `pm_params` object with `sigma_eps` set.
`sigma_nu`, `n`	The chosen values (single values). Recall the calibration rule of step 2: keep the *largest* `n` whose worst-case risk still meets the ceiling, so as to spare cells of intermediate dominance (see `pm_calib_dominance()` and the trade-off map `pm_plot_tradeoff()`).
`beta`, `tau`	Scenario-I accuracy threshold and risk ceiling. Default to the `dominance` policy already stored in `params`.
`level`	Confidence level for the reported CI loss (default 0.95).
`rho`	Grid used to locate the worst case over rho. Default `(1:1000) / 1000`.

Details

Requires `sigma_eps` to be already set (step 1), since both the scenario-I risk and the information loss depend on it. The function reports the resulting worst-case risk over `rho`, where that worst case occurs, and the range of the information loss – from its floor (rho -> 0, driven by `sigma_eps` alone) to its ceiling (rho = 1, driven by `sigma_nu`). A warning is raised when the worst-case risk exceeds `tau`: the values are still recorded, the producer being free to accept the overshoot knowingly.

Value

The updated `pm_params`, invisibly.

Examples

```
para <- pm_commit_diff(pm_params())
para <- pm_commit_dominance(para, sigma_nu = 0.3, n = 6)
para
```

`pm_draws` *Derive the two Gaussian draws from a cell key*

Description

Implements Definitions 6-7 of the paper. From the cell key `CK`, the name of the indicator and the aggregation operation, two distinct strings are built – one per draw – and hashed with `SHA-512`. Each digest is mapped to a uniform draw in]0;1[, then turned into a Gaussian draw by quantile inversion: `nu` from `N(0, sigma_nu^2)` and `eps` from `N(0, sigma_eps^2)`.

Usage

```
pm_draws(ck, params, indicator, operation = "sum", key_digits = 9L)
```

Arguments

`ck`	Numeric vector of cell keys in [0;1].
`params`	A fully calibrated `pm_params` object.
`indicator`	Name of the indicator the aggregate is computed from (e.g. `"turnover"`). Enters the hashed string.
`operation`	Name of the aggregation (default `"sum"`). Enters the hashed string, so that several statistics on the same indicator get independent draws.
`key_digits`	Number of decimals used to turn the key into a string. Fixed for reproducibility; changing it changes every draw.

Details

Determinism, avalanche effect and uniformity of `SHA-512` ensure that the same cell always receives the same perturbation, while draws remain uncorrelated across indicators and between `nu` and `eps`.

Value

A `data.frame` with columns `ck_nu`, `ck_eps`, `nu`, `eps`.

See Also

dominoise-package paper's proposition 8

Examples

```
para <- pm_commit_dominance(pm_commit_diff(pm_params()), sigma_nu = 0.4, n = 4)
pm_draws(c(0.12, 0.87), para, indicator = "turnover")
```

pm_params *Create the perturbation-parameter object containing all the parameters of the noise.*

Description

Create the perturbation-parameter object containing all the parameters of the noise.

Usage

```
pm_params(
  beta_dominance = 0.2,
  tau_dominance = 0.9,
  beta_prule = 0.1,
  tau_prule = 0.9,
  s_prule = 0.95,
  beta_diff = 0.05,
  tau_diff = 0.95,
  sigma_nu = NA_real_,
  sigma_eps = NA_real_,
  n = NA_real_
)
```

Arguments

`beta_dominance`, `tau_dominance`
: Accuracy threshold and risk ceiling for scenario I (dominance rule). Defaults to 0.2 / 0.9.

`beta_prule`, `tau_prule`, `s_prule`
: Threshold, ceiling and cumulated share of the two largest contributors for scenario II (p%-rule). `s_prule = 0.95` is the relaxed case IIb in the paper; `1` would be the worst case IIa. Defaults to 0.1 / 0.9 / 0.95.

`beta_diff`, `tau_diff`
: Threshold and ceiling for the differencing scenario. Defaults: 0.05 / 0.95.

`sigma_nu`, `sigma_eps`, `n`
: Mechanism parameters, filled in during calibration. Left as `NA` until decided.

Value

An object of class `pm_params`: a list with two elements, `policy` (sub-lists `dominance`, `prule` and `diff`, each holding the accuracy threshold `beta` and the risk ceiling `tau`, plus `s` for `prule`) and `mechanism` (`sigma_nu`, `sigma_eps`, `n`).

Examples

```
para <- pm_params()
para
```

pm_perturb — *Perturb an aggregated table*

Description

Applies the mechanism of Definition 2 to a table of sums: `Y' = Y (1 + rho^n nu + eps)` with `rho = X1 / Y`. The draws are obtained from the cell keys via `pm_draws()`, so the perturbation is reproducible: re-running on the same cells and the same indicator returns the same values.

Usage

```
pm_perturb(
  data,
  total,
  x1,
  ck_var,
  params,
  indicator = NULL,
  operation = "sum",
  x2 = NULL,
  reduce_key = FALSE,
  key_digits = 9L
)
```

Arguments

`data`	A `data.frame` of aggregated data.
`total`, `x1`	Column names (strings): the aggregate and the largest contribution, in the same unit.
`ck_var`	Column name of the aggregated cell key.
`params`	A fully calibrated `pm_params` object.
`indicator`	Name of the indicator; defaults to the `total` column name.
`operation`	Aggregation name entering the hash (default `"sum"`).
`x2`	Optional column name of the second largest contribution. When given, the share `rho2 = X2 / Y` is added, enabling the scenario-II assessment.
`reduce_key`	If `TRUE`, reduce `ck_var` to its fractional part (use it when the column holds the raw sum of individual keys, Definition 6). If `FALSE` (default), keys must already lie in [0;1].
`key_digits`	Decimals used to stringify the key before hashing (default 9).

Details

The mechanism is defined for strictly positive totals. Cells with a non-positive total are left unperturbed and flagged, with a warning: being multiplicative, the noise protects neither zeros nor near-zero values.

Value

data with added columns rho, rho2 (when x2 is given), ck_nu, ck_eps, and the perturbed total named <total>_pert. Calibration parameters and hashing settings are attached as the pm_meta attribute.

Examples

```
para <- pm_commit_dominance(pm_commit_diff(pm_params()), sigma_nu = 0.4, n = 4)
tab <- data.frame(turnover = c(1000, 500), x1 = c(850, 120), ck = c(0.12, 0.87))
pm_perturb(tab, total = "turnover", x1 = "x1", ck_var = "ck", params = para)
```

---

pm_plot_impact *Observed perturbation against dominance*

Description

Plots the relative deviation actually undergone by each cell against its dominance level, and overlays the theoretical envelope predicted before any data were touched: the level-level confidence band of the relative loss (Definition 4) and the mean absolute loss.

Usage

```
pm_plot_impact(x, level = 0.95, colour_by = NULL, alpha = 0.35, n_grid = 200)
```

Arguments

x	A table returned by pm_perturb().
level	Confidence level of the band (default 0.95).
colour_by	Optional column name used to colour the points (e.g. a publication stratum).
alpha	Point transparency; lower it on large tables.
n_grid	Number of dominance values used to draw the envelope.

Details

The mechanism being analytical, the cloud of points should fill the band and straddle the mean lines. That agreement is the natural consistency check between what was promised at calibration time and what the table received.

Value

A ggplot object.

See Also

dominoise-package paper's proposition 3

Examples

```
set.seed(123)
N = 1000
params <- pm_commit_dominance(pm_commit_diff(pm_params()), sigma_nu = 0.4, n = 4)
rhos <- runif(N, 0, 1)
tab <- data.frame(turnover = runif(N, 0,1000), ck = runif(N, 0, 1))
tab$x1 <- tab$turnover * rhos
res <- pm_perturb(tab, "turnover", "x1", "ck", params)
pm_plot_impact(res)
```

---

pm_plot_risk_max *Worst-case risk against a mechanism parameter*

Description

Reduces a calibration table to the worst case over `rho` of the disclosure risk, at a fixed `sigma_eps`, and plots it against one of the two mechanism parameters. Panels are split by `scenario` when several are present.

Usage

```
pm_plot_risk_max(
  x,
  x_axis = c("sigma_nu", "n"),
  sigma_nu = NULL,
  n = NULL,
  beta = NULL,
  sigma_eps = NULL,
  scenario = NULL,
  tau = NULL,
  mark_frontier = NULL,
  marks = NULL,
  thresholds = c(0.5, 0.8)
)
```

Arguments

`x`	A calibration table (e.g. from `pm_calib_dominance()`). Needs a fine grid on the x-axis parameter to draw smooth curves.
`x_axis`	Parameter on the x-axis: `"sigma_nu"` (default) or `"n"`.
`sigma_nu`, `n`, `beta`, `sigma_eps`, `scenario`	Optional values to keep; `NULL` keeps everything present in the table.
`tau`	Optional risk ceiling, drawn as an extra dashed line on top of `thresholds`. A single value is required to mark the frontier.

mark_frontier	Whether to mark the largest admissible n of each sigma_nu. NULL (default) marks it whenever it is defined, i.e. when x_axis = "n" and a single tau is given. TRUE forces it and warns when it cannot be drawn; FALSE never draws it.
marks	Values of the x-axis parameter highlighted with a point; NULL (default) for none.
thresholds	Risk levels drawn as dashed horizontal lines.

Details

With x_axis = "sigma_nu" (default) this reproduces the middle panel of Figure 3 of the paper, generalised: the risk decreases with sigma_nu, one curve per n.

With x_axis = "n" it is the view that accompanies pm_suggest_n(): the risk increases monotonically with n, one curve per sigma_nu. Each crossing of the ceiling tau is the n_max of its sigma_nu – the very root that pm_suggest_n() solves for. Those points are overlaid when mark_frontier allows it, computed by the same function so that the plot and the table always agree.

Value

A ggplot object.

See Also

pm_suggest_n(), plot.pm_calib_dominance().

Examples

```
# Figure 3 (middle panel): worst-case risk against sigma_nu
grid <- pm_calib_dominance(sigma_nu = (2:100)/200,
                           n = c(3, 6, 9, 12), beta = 0.2)
pm_plot_risk_max(grid, marks = c(0.05, 0.1, 0.2, 0.3, 0.4, 0.5))

# Against n, with the frontier found by pm_suggest_n()
grid <- pm_calib_dominance(sigma_eps = 0.031, beta = 0.2,
                           sigma_nu = c(0.3, 0.4, 0.5), n = (5:60)/5)
pm_plot_risk_max(grid, x_axis = "n", tau = 0.5)
```

---

pm_plot_risk_profile *Risk profile as a function of dominance*

Description

Reproduces Figure 2 of the paper, generalised: the disclosure risk against the dominance level rho, one curve per shape parameter n, panels split by sigma_nu (and by scenario when the table carries several).

Usage

```
pm_plot_risk_profile(
  x,
  sigma_nu = NULL,
  n = NULL,
  beta = NULL,
  sigma_eps = NULL,
  scenario = NULL,
  thresholds = c(0.5, 0.8)
)
```

Arguments

x	A calibration table (e.g. from pm_calib_dominance()).
sigma_nu, n, beta, sigma_eps, scenario	Optional values to keep; NULL keeps everything present in the table. Filtering is usually needed on beta and sigma_eps to keep the panel grid readable.
thresholds	Risk levels drawn as dashed horizontal lines.

Value

A ggplot object.

Examples

```
grid <- pm_calib_dominance(beta = c(0.1,0.2))
pm_plot_risk_profile(grid, sigma_nu = c(0.1, 0.2, 0.4, 0.5), beta = 0.2)
```

---

pm_plot_tradeoff *Risk-utility trade-off map*

Description

Reproduces Figure 4 of the paper: for each (sigma_nu, n) the curve traces the (information loss, risk) couple as rho runs over]0;1]. Loss on the x-axis, risk on the y-axis, colour by sigma_nu, panels by n, and open symbols marking a few reference dominance levels.

Usage

```
pm_plot_tradeoff(
  x,
  loss = c("EZ", "CI"),
  sigma_nu = NULL,
  n = NULL,
  beta = NULL,
  sigma_eps = NULL,
```

```
  scenario = NULL,
  rho_marks = c(0.8, 0.9, 0.95),
  thresholds = c(0.5, 0.8)
)
```

Arguments

x	A calibration table (e.g. from pm_calib_dominance()).
loss	Loss metric on the x-axis: "EZ" (mean absolute loss, default) or "CI" (upper confidence bound).
sigma_nu, n, beta, sigma_eps, scenario	Optional values to keep.
rho_marks	Dominance levels highlighted with a symbol.
thresholds	Risk levels drawn as dashed horizontal lines.

Details

Generalised on the choice of the risk metric only: pick the scenario with scenario when the table carries several, since a single risk column can be plotted at a time.

Value

A ggplot object.

Examples

```
grid <- pm_calib_dominance(sigma_eps = 0.031, beta = 0.2)
pm_plot_tradeoff(grid)
```

---

pm_sigma_eps *Smallest* sigma_eps *guaranteeing the differencing-risk ceiling*

Description

Closed-form inversion of the worst-case upper bound of the differencing risk.

Usage

```
pm_sigma_eps(beta, tau)
```

Arguments

beta, tau	Numeric. Accuracy threshold and risk ceiling.

Value

A numeric vector of sigma_eps values, vectorised over beta/tau.

See Also

dominoise-package (Proposition 7 of the paper).

Examples

```
pm_sigma_eps(beta = 0.05, tau = 0.95)
pm_sigma_eps(beta = c(0.05, 0.1), tau = 0.9)
```

---

pm_suggest_n *Largest admissible shape parameter n*

Description

Implements the step-2 calibration rule. The worst-case scenario-I risk increases with `n`, while the information loss decreases with `n` at every `rho < 1` and is unchanged at `rho = 1`. Both objectives are therefore monotone and opposite in `n`: there is no interior optimum, and the best choice is the upper edge of the admissible set, `n_max = sup{n : max_rho mu_I <= tau}`. Because the risk is monotone, that edge is found by root-finding rather than by a grid search.

Usage

```
pm_suggest_n(
  params = NULL,
  sigma_nu = NULL,
  beta = NULL,
  tau = NULL,
  sigma_eps = NULL,
  margin = 0,
  n_range = c(1, 30),
  integer = FALSE,
  level = 0.95,
  rho = NULL
)
```

Arguments

`params`	Optional `pm_params`; supplies `sigma_eps`, `beta` and `tau`.
`sigma_nu`	Numeric vector of candidate values. Default `(1:10) / 20`.
`beta, tau`	Scenario-I threshold and ceiling. Default to the `dominance` policy of `params`.
`sigma_eps`	Fixed differencing noise. Defaults to `params$mechanism$sigma_eps`.
`margin`	Safety margin in [0,1): the effective ceiling is `tau * (1 - margin)`. `n_max` is a boundary solution with zero slack, so a small margin guards against later revisions of the policy. Default 0.
`n_range`	Search interval for `n`. Default `c(1, 30)`.

`integer`	If `TRUE`, round `n_max` down to an integer – the only conservative rounding, the risk being increasing in `n`.
`level`	Confidence level for the reported CI loss (default 0.95).
`rho`	Grid used to locate the worst case. Default `(1:1000) / 1000`.

Details

The rule holds whatever the dominance profile of the table: since utility improves at every `rho`, no weighting by the actual distribution of `rho` could favour a smaller `n`.

Note this determines `n` *given* `sigma_nu`. Every point of the returned frontier meets the ceiling exactly, so choosing among them is a pure utility arbitrage – larger `sigma_nu` costs more at `rho = 1` but spares cells of intermediate dominance. The reported losses are there to settle it.

Value

A `data.frame` with one row per `sigma_nu`: `n_max`, `risk_at_n_max`, and the information loss at `rho = 1` and at the dominance threshold `1 - beta`. `n_max` is `NA` when no `n` meets the ceiling (raise `sigma_nu`), and the upper end of `n_range` when the whole range qualifies.

Examples

```
para <- pm_commit_diff(pm_params())
pm_suggest_n(para, sigma_nu = c(0.3, 0.4, 0.5))
```

`summary.pm_calib_dominance`

Worst-case risk and loss range of a dominance calibration grid

---

Description

Reduces the rho-resolved table to one row per parameter combination `(sigma_nu, sigma_eps, n, beta)`. For each, reports where the scenario-I risk peaks and its value, and the range of the information loss: the minimum (smallest rho of the grid) and the maximum (rho = 1), in both metrics – expectation E|Z| and upper CI bound. The loss is monotone increasing in rho, so these are read at the extreme rho rows of each group.

Usage

```
## S3 method for class 'pm_calib_dominance'
summary(object, ...)
```

Arguments

`object`	A `pm_calib_dominance` table.
`...`	Ignored.

Value

A `data.frame` of class `pm_calib_dominance_summary`, with columns `sigma_nu`, `sigma_eps`, `n`, `beta`, `rho_at_max_risk`, `risk_max`, `EZ_min`, `EZ_max`, `CI_min`, `CI_max` (losses in percent).

Examples

```
grid <- pm_calib_dominance(sigma_nu = c(0.3, 0.4), n = c(3, 6), beta = 0.2)
summary(grid)
```

Index

assess_loss_ci, 2
assess_loss_expectation, 3
assess_risk_diff, 4
assess_risk_empirical, 5
assess_risk_I, 6
assess_risk_II, 6
assess_utility_empirical, 7

dominoise-package, *3–8*, *14*, *17*, *22*

plot(), *10*
plot.pm_calib_diff, 8
plot.pm_calib_dominance, 9
plot.pm_calib_dominance(), *19*
pm_calib_diff, 10
pm_calib_diff(), *8*, *11*, *12*
pm_calib_dominance, 10
pm_calib_dominance(), *13*, *18*, *20*, *21*
pm_commit_diff, 12
pm_commit_dominance, 13
pm_commit_dominance(), *10*
pm_draws, 14
pm_draws(), *16*
pm_params, 15
pm_perturb, 16
pm_perturb(), *5*, *7*, *17*
pm_plot_impact, 17
pm_plot_risk_max, 18
pm_plot_risk_profile, 19
pm_plot_tradeoff, 20
pm_plot_tradeoff(), *13*
pm_sigma_eps, 21
pm_suggest_n, 22
pm_suggest_n(), *10*, *19*

summary(), *11*
summary.pm_calib_dominance, 23