

Package: **paretoinfer** (via r-universe)

October 24, 2026

Title Sequential Identification of Approximately Pareto-Optimal Alternatives under Noise

Version 0.1.0

Description Sequential identification of epsilon-Pareto-optimal alternatives among a finite set with noisy, possibly correlated objectives. Simultaneous anytime-valid confidence sequences for every objective of every alternative (Waudby-Smith and Ramdas, 2024, <[doi:10.1093/jrsssb/qkad009](https://doi.org/10.1093/jrsssb/qkad009)>) certify, at any time, which alternatives are epsilon-dominated and which are not, report the estimated frontier, the plausible Pareto set and the certified-dominated set, propose the next alternative to evaluate, and record an auditable result contract.

License GPL (>= 3)

URL <https://castlaboratory.github.io/paretoinfer/>,
<https://github.com/castlaboratory/paretoinfer>

BugReports <https://github.com/castlaboratory/paretoinfer/issues>

Encoding UTF-8

RoxygenNote 8.0.0

Language en-GB

Depends R (>= 4.1)

Imports cli (>= 3.6.0), generics, ggplot2 (>= 3.5.0), rlang (>= 1.1.0), seqbench (>= 0.1.0), stats, tibble (>= 3.2.0), utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author André Leite [aut, cre] (ORCID: <<https://orcid.org/0000-0002-4718-9766>>), Raydonal Ospina [aut] (ORCID: <<https://orcid.org/0000-0002-9884-9090>>), Cristiano Ferraz [aut] (ORCID: <<https://orcid.org/0000-0002-6838-6734>>)

Maintainer André Leite <leite@castlab.org>
Repository https://cran.r-universe.dev
Date/Publication 2026-10-24 15:50:01 UTC
RemoteUrl https://github.com/cran/paretoinfer
RemoteRef HEAD
RemoteSha e4927cb93b0f0a48109ab5dd323984e96913f5db

Contents

autoplot.paretoinfer_state	2
choose_next_evaluation	3
glance.paretoinfer_state	4
initialize_pareto	4
pareto_design	5
pareto_report	6
pareto_run	7
plausible_pareto_set	8
tidy.paretoinfer_state	9
update_objectives	9

Index	11
--------------	-----------

autoplot.paretoinfer_state
<i>Plot the uncertain frontier</i>

Description

Draws, for two objectives, the simultaneous confidence box of every alternative (the product of its two running intersections), its point estimate, and the estimated frontier. The boxes correspond to the actual guarantee: with probability at least $1 - \alpha$ every true mean vector lies in its box at every time. Alternatives are coloured by status.

Usage

```
## S3 method for class 'paretoinfer_state'
autoplot(object, objectives = c(1L, 2L), ...)

plot_uncertain_frontier(state, objectives = c(1L, 2L))
```

Arguments

object	A paretoinfer_state.
objectives	The two objectives to draw (names or positions); default the first two.
...	Unused.
state	A paretoinfer_state.

Value

A ggplot.

Examples

```
design <- pareto_design(alternatives = 3, objectives = 2, bounds = c(0, 1), epsilon = 0.05)
set.seed(1)
ev <- data.frame(alternative = rep(c("A1", "A2", "A3"), each = 40),
                 y1 = runif(120, c(0.1, 0.5, 0.6), c(0.3, 0.7, 0.8)),
                 y2 = runif(120, c(0.6, 0.1, 0.6), c(0.8, 0.3, 0.8)))
autoplot(update_objectives(initialize_pareto(design), ev))
```

choose_next_evaluation

Choose the next alternative(s) to evaluate

Description

Scores every uncertain alternative by the width of its simultaneous confidence box, normalised by the objective ranges and divided by its evaluation cost, and returns the highest-scoring ones. Certified alternatives (dominated or optimal) are never proposed: more evaluations cannot change their certificate. The rule is a heuristic that targets the ambiguous comparisons; it does not affect the validity of the certificates, which holds for any adaptive choice.

Usage

```
choose_next_evaluation(state, n = 1L)
```

Arguments

state	A <code>paretoinfer_state</code> .
n	Number of alternatives to propose.

Value

A tibble with columns `alternative`, `score` (normalised width per unit cost), `width` (largest normalised width over objectives) and `n_evaluations`; zero rows when nothing is uncertain.

Examples

```
design <- pareto_design(alternatives = 3, objectives = 2, bounds = c(0, 1), epsilon = 0.05)
state <- initialize_pareto(design)
choose_next_evaluation(state, n = 2)
```

```
glance.paretoinfer_state
```

Glance at an identification

Description

Glance at an identification

Usage

```
## S3 method for class 'paretoinfer_state'
glance(x, ...)
```

Arguments

`x` A paretoinfer_state.
`...` Unused.

Value

A one-row tibble: K, m, alpha, n_evaluations, total_cost, n_frontier, n_plausible, n_certified_optimal, n_uncertain, n_dominated, identified, stopping_reason.

Examples

```
design <- pareto_design(alternatives = 2, objectives = 2, bounds = c(0, 1), epsilon = 0.1)
glance(initialize_pareto(design))
```

```
initialize_pareto      Initialise a Pareto identification
```

Description

Initialise a Pareto identification

Usage

```
initialize_pareto(design)
```

Arguments

`design` A `pareto_design()`.

Value

An object of class paretoinfer_state.

Examples

```
design <- pareto_design(alternatives = 4, objectives = 2, bounds = c(0, 1), epsilon = 0.05)
state <- initialize_pareto(design)
state
```

pareto_design

*Design a sequential Pareto identification problem***Description**

Fixes, before any evaluation is seen, the finite set of alternatives, the objectives (all minimised) with their known bounds, the tolerance epsilon that defines practical dominance, the error level, the confidence-sequence boundary and the budget.

Usage

```
pareto_design(
  alternatives,
  objectives = 2L,
  bounds,
  alpha = 0.05,
  epsilon = 0,
  boundary = "betting",
  cost = 1,
  budget = Inf,
  max_evaluations = Inf,
  c = 0.5
)
```

Arguments

alternatives	Character vector of alternative names, or a single integer K (names "A1", ..., "AK").
objectives	Character vector of objective names, or a single integer m (names "y1", ..., "ym"). All objectives are minimised.
bounds	Known bounds of each objective: a list of m vectors $c(a, b)$, or one vector $c(a, b)$ used for all. Evaluations outside the bounds are refused, never clipped.
alpha	Error level of the simultaneous guarantee.
epsilon	Tolerance of practical dominance, per objective (length m) or a single value, in objective units. Zero means exact dominance; positive values operationalise ties and are required for the procedure to stop on near-ties.
boundary	Confidence-sequence boundary from <code>seqbench::seqbench_boundaries()</code> ("betting" by default; "naive_fixed" is refused because it is not anytime-valid).
cost	Evaluation cost per alternative (length K or a single value).
budget	Total cost allowed; Inf for none.

max_evaluations

Maximum number of evaluations; Inf for none.

c

Truncation constant of the betting and empirical-Bernstein bets, passed to `seqbench::boundary_init()`.

Value

An object of class `pareto_design`.

Dominance and guarantee

Alternative j *epsilon-dominates* k when, for every objective i , $\mu[j, i] \leq \mu[k, i] + \text{epsilon}[i]$: j is at least as good as k up to the tolerance everywhere. The package maintains one confidence sequence per alternative and objective, each at level $\alpha / (K * m)$, so that with probability at least $1 - \alpha$ every true mean lies in its running intersection at every time. On that event every certificate the package issues is true: an alternative reported as *certified dominated* is truly epsilon-dominated by the alternative named, and one reported as *certified optimal* is truly not epsilon-dominated by any other plausible alternative. When two alternatives certifiably epsilon-dominate each other (an epsilon-tie), only the later one in design order is discarded, so every tie keeps one representative; the discarded one is still truly epsilon-dominated by the alternative named. The guarantee holds under optional stopping, adaptive choice of what to evaluate next and arbitrary dependence between the objectives of one evaluation.

References

Waudby-Smith, I. and Ramdas, A. (2024). Estimating means of bounded random variables by betting. *Journal of the Royal Statistical Society: Series B*, 86(1), 1–27. doi:10.1093/jrsssb/qkad009

Auer, P., Chiang, C.-K., Ortner, R. and Drugan, M. (2016). Pareto front identification from stochastic bandit feedback. *AISTATS 2016*, 939–947.

Examples

```
pareto_design(alternatives = 6, objectives = c("error", "time"), bounds = c(0, 1),
              alpha = 0.05, epsilon = 0.02)
```

pareto_report	<i>The full report of an identification</i>
---------------	---

Description

Everything needed to audit the result: the design (alternatives, objectives, bounds, tolerance, level and its split, boundary, budget), the evaluations and cost per alternative, the three sets with their certificates, the simultaneous confidence boxes, the stopping reason and the software versions.

Usage

```
pareto_report(state)
```

Arguments

state A `paretoinfer_state`.

Value

A list of class `pareto_report`.

Examples

```
design <- pareto_design(alternatives = 2, objectives = 2, bounds = c(0, 1), epsilon = 0.1)
state <- update_objectives(initialize_pareto(design),
  data.frame(alternative = c("A1", "A2"), y1 = c(0.2, 0.8), y2 = c(0.3, 0.9)))
pareto_report(state)
```

pareto_run

Run an identification to completion with a simulator

Description

Repeats "choose the next alternative, evaluate it, update" until every alternative is certified or the budget is exhausted.

Usage

```
pareto_run(design, simulate, batch = 1L, initial = 2L, max_rounds = 1e+05)
```

Arguments

design A `pareto_design()`.

simulate A function `function(alternative, n)` returning a data frame (or matrix) with `n` rows and one column per objective, in the design's objective order or with the objective names.

batch Evaluations per call for the chosen alternative.

initial Evaluations of every alternative before the adaptive phase.

max_rounds Safety cap on the number of adaptive rounds.

Value

The final `paretoinfer_state`.

Examples

```

design <- pareto_design(alternatives = 4, objectives = 2, bounds = c(0, 1), epsilon = 0.05,
                      max_evaluations = 400)
means <- rbind(c(0.2, 0.7), c(0.5, 0.4), c(0.7, 0.2), c(0.6, 0.6))
sim <- function(alternative, n) {
  k <- match(alternative, design$alternatives)
  data.frame(y1 = runif(n, means[k, 1] - 0.1, means[k, 1] + 0.1),
            y2 = runif(n, means[k, 2] - 0.1, means[k, 2] + 0.1))
}
set.seed(1)
state <- pareto_run(design, sim)
state

```

plausible_pareto_set *The three sets of a Pareto identification*

Description

The three sets of a Pareto identification

Usage

```
plausible_pareto_set(state)
```

Arguments

state A paretoinfer_state.

Value

A list of class `pareto_sets` with `frontier` (alternatives whose point estimates are not dominated by the estimate of any other plausible alternative), `plausible` (not certified dominated), `certified_optimal` (plausible and certified not epsilon-dominated by any other plausible alternative), `uncertain` (plausible but not certified), `dominated` (certified epsilon-dominated), `identified` (no uncertain alternative left) and `table`, a tibble with one row per alternative: `status`, `on_estimated_frontier`, `dominated_by`, `n_evaluations`, `cost`, `cs_empty`.

Examples

```

design <- pareto_design(alternatives = 3, objectives = 2, bounds = c(0, 1), epsilon = 0.05)
state <- initialize_pareto(design)
set.seed(1)
ev <- data.frame(alternative = rep(c("A1", "A2", "A3"), each = 40),
                y1 = runif(120, c(0.1, 0.5, 0.6), c(0.3, 0.7, 0.8)),
                y2 = runif(120, c(0.6, 0.1, 0.6), c(0.8, 0.3, 0.8)))
plausible_pareto_set(update_objectives(state, ev))

```

tidy.paretoinfer_state

Tidy the confidence boxes

Description

One row per alternative and objective with the point estimate, the running intersection of the confidence sequence, the number of evaluations and the status of the alternative.

Usage

```
## S3 method for class 'paretoinfer_state'
tidy(x, ...)
```

Arguments

x	A paretoinfer_state.
...	Unused.

Value

A tibble with columns alternative, objective, estimate, lower, upper, n_evaluations, status.

Examples

```
design <- pareto_design(alternatives = 2, objectives = 2, bounds = c(0, 1), epsilon = 0.1)
state <- update_objectives(initialize_pareto(design),
  data.frame(alternative = c("A1", "A2"), y1 = c(0.2, 0.8), y2 = c(0.3, 0.9)))
tidy(state)
```

update_objectives

Update the identification with new evaluations

Description

Feeds one or more evaluations, each a vector of objective values for one alternative, to the confidence sequences. The running intersections are updated and the three sets are recomputed.

Usage

```
update_objectives(state, evaluations)
```

Arguments

state	A <code>paretoinfer_state</code> .
evaluations	Data frame with a column <code>alternative</code> (name as in the design) and one numeric column per objective; an optional cost column overrides the design cost for those rows.

Value

The updated state.

Examples

```
design <- pareto_design(alternatives = c("a", "b"), objectives = c("y1", "y2"),
                      bounds = c(0, 1), epsilon = 0.05)
state <- initialize_pareto(design)
state <- update_objectives(state, data.frame(alternative = c("a", "b", "a"),
                                           y1 = c(0.2, 0.5, 0.25), y2 = c(0.6, 0.3, 0.55)))
state
```

Index

`autoplot.paretoinfer_state`, [2](#)
`choose_next_evaluation`, [3](#)
`glance.paretoinfer_state`, [4](#)
`initialize_pareto`, [4](#)
`pareto_design`, [5](#)
`pareto_design()`, [4](#), [7](#)
`pareto_report`, [6](#)
`pareto_run`, [7](#)
`plausible_pareto_set`, [8](#)
`plot_uncertain_frontier`
 (`autoplot.paretoinfer_state`), [2](#)
`seqbench::boundary_init()`, [6](#)
`seqbench::seqbench_boundaries()`, [5](#)
`tidy.paretoinfer_state`, [9](#)
`update_objectives`, [9](#)