

Package ‘seqbench’

October 8, 2026

Title Anytime-Valid Sequential Benchmarking of Algorithms

Version 0.1.0

Description Sequential, anytime-valid comparison of algorithms on paired losses with a practical-equivalence margin. Confidence sequences for the mean paired difference, superiority/equivalence/continue decisions, cost accounting and auditable trajectories.

License GPL (>= 3)

URL <https://github.com/castlaboratory/seqbench>

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

Encoding UTF-8

RoxygenNote 8.0.0

Depends R (>= 4.1)

Imports cli (>= 3.6.0), generics, ggplot2 (>= 3.5.0), rlang (>= 1.1.0), 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],
Raydonal Ospina [aut],
Cristiano Ferraz [aut]

Maintainer André Leite <leite@castlab.org>

Repository CRAN

Date/Publication 2026-10-08 11:20:02 UTC

Contents

autoplot.seqbench_comparison	2
comparison_design	3

comparison_report	5
glance.seqbench_comparison	5
initialize_comparison	6
planning_horizon	7
seqbench_boundaries	8
stopping_decision	8
tidy.seqbench_comparison	9
update_comparison	9
Index	11

autoplot.seqbench_comparison
<i>Plot the inferential trajectory of a comparison</i>

Description

Running-intersection confidence sequence for the mean paired difference against the number of instances, with the equivalence band $[-margin, margin]$ and the zero line.

Usage

```
autoplot.seqbench_comparison(object, ...)
```

Arguments

- object A seqbench_comparison with at least one instance.
- ... Unused.

Value

A ggplot object.

Examples

```
design <- comparison_design(margin = 0.05, bounds = c(0, 1), boundary = "hoeffding")
set.seed(2)
losses <- data.frame(instance = 1:40, loss_a = runif(40, 0, .5), loss_b = runif(40, .2, .7))
state <- update_comparison(initialize_comparison(design), losses)
ggplot2::autoplot(state)
```

comparison_design	<i>Design a sequential comparison of two algorithms</i>
-------------------	---

Description

Fixes, before any data is seen, everything the procedure needs: the error level, the margin of practical equivalence, the loss bounds, the boundary and the budget. The design is immutable; `initialize_comparison()` turns it into a state that `update_comparison()` advances.

Usage

```
comparison_design(
  alpha = 0.05,
  margin,
  bounds,
  boundary = "betting",
  paired = TRUE,
  cost_per_round = 1,
  budget = Inf,
  n_max = Inf,
  betting_c = 0.5,
  betting_theta = 0.5,
  betting_grid = 1001L,
  sd_max = NULL
)
```

Arguments

alpha	Miscoverage level of the confidence sequence, in (0, 1). The probability that the procedure ever issues a false declaration (of any of the three kinds) is at most alpha.
margin	Margin of practical equivalence $\delta > 0$, in loss units. Algorithm A is declared relevantly better when the whole confidence sequence for $\text{mean}(\text{loss}_a - \text{loss}_b)$ lies below $-\text{margin}$; B when it lies above margin ; the two are equivalent when it lies inside $[-\text{margin}, \text{margin}]$.
bounds	Known bounds $c(\text{lower}, \text{upper})$ of the losses of both algorithms. Paired differences then lie in $[\text{lower} - \text{upper}, \text{upper} - \text{lower}]$. A loss outside the bounds is an error, never clipped.
boundary	One of <code>seqbench_boundaries()</code> . "betting" (default) is the hedged capital confidence sequence of Waudby-Smith & Ramdas (2024); "empirical_bernstein" and "hoeffding" are their conservative predictable-plug-in references; "bernstein_declared" is a predictable-plug-in Bennett confidence sequence that uses a declared upper bound <code>sd_max</code> on the standard deviation of the paired difference and is valid only if that bound holds. Its gain over the betting boundary is modest (a logarithmic factor in the declared variance): for bounded observations the Bennett bound keeps a range term of order $\log(2/\alpha) / t$ whatever the variance, so the

boundary is provided for completeness and for the package's cost study rather than as a shortcut; "naive_fixed" is a fixed-sample t interval recomputed at every step, **not valid** under optional stopping, provided only as a negative control for experiments.

paired	Must be TRUE. Unpaired designs are not implemented; the argument exists so that the limitation is explicit.
cost_per_round	Default cost of one (instance, seed) evaluation of both algorithms together, used when the data carry no cost column.
budget	Total cost after which the procedure stops with an inconclusive outcome. Inf for no cost limit.
n_max	Maximum number of instances. Inf for no limit.
betting_c, betting_theta, betting_grid	Tuning of the betting boundary: truncation constant of the bets (default 1/2, the value recommended by Waudby-Smith & Ramdas, who also suggest 3/4), hedging weight (default 1/2) and grid size on $[0, 1]$ (default 1001). betting_c is also the truncation constant of the empirical-Bernstein boundary. Larger values bet more aggressively and stop earlier: in the package's simulation study (2,000 replications per cell) betting_c = 0.9 needed about 40% fewer instances than 1/2 at low noise and 20% fewer at high noise, with the largest observed coverage failure rising from 0.6% to 1.4% at $\alpha = 0.05$; 0.99 gains little more. Coverage is guaranteed for any value in $(0, 1)$.
sd_max	Required for boundary = "bernstein_declared": declared upper bound on the standard deviation of the per-instance (seed-averaged) paired difference, in loss units. This is an additional assumption (A5) recorded in the result contract.

Value

An object of class seqbench_design (a list).

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

See Also

`initialize_comparison()`, `planning_horizon()`

Examples

```
design <- comparison_design(alpha = 0.05, margin = 0.02, bounds = c(0, 1))
design
```

comparison_report	<i>Full result contract of a comparison</i>
-------------------	---

Description

Returns every element the protocol promises: estimand, estimate, current confidence sequence, required assumptions, diagnostics, decision and stopping reason, resources consumed, seeds and versions. `summary()` is an alias.

Usage

```
comparison_report(state)

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

Arguments

state	A seqbench_comparison.
object	A seqbench_comparison.
...	Unused.

Value

A list of class seqbench_report.

Examples

```
state <- initialize_comparison(comparison_design(margin = 0.02, bounds = c(0, 1)))
comparison_report(state)
```

glance.seqbench_comparison	<i>Glance at a comparison</i>
----------------------------	-------------------------------

Description

Glance at a comparison

Usage

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

Arguments

x A seqbench_comparison.
 ... Unused.

Value

A one-row tibble: boundary, valid, alpha, margin, n_instances, n_evaluations, cost, estimate, lower, upper, decision, stopping_reason.

Examples

```
state <- initialize_comparison(comparison_design(margin = 0.02, bounds = c(0, 1)))
glance(state)
```

initialize_comparison *Start a sequential comparison*

Description

Creates the state object that [update_comparison\(\)](#) advances one instance at a time. Nothing is observed yet; the confidence sequence is [a, b], the full range of the paired difference.

Usage

```
initialize_comparison(design)
```

Arguments

design A [comparison_design\(\)](#).

Value

An object of class seqbench_comparison. Its main components are design, data (all rows seen so far), trajectory (one row per instance: estimate, confidence sequence, running intersection, cost and decision at that time), decision, stopping_reason, diagnostics and meta (versions, timestamps). Use [tidy\(\)](#) for the trajectory, [glance\(\)](#) for a one-row summary and [comparison_report\(\)](#) for the full contract.

Examples

```
design <- comparison_design(margin = 0.02, bounds = c(0, 1))
state <- initialize_comparison(design)
state
```

planning_horizon	<i>Planning horizon for a distance to the decision threshold</i>
------------------	--

Description

Number of instances after which the confidence sequence is expected to have declared the true hypothesis, when the true mean paired difference is at distance distance from the nearest threshold -margin or margin. Two modes:

Usage

```
planning_horizon(design, distance, sd = NULL, max_t = 1e+06)
```

Arguments

design	A <code>comparison_design()</code> .
distance	Positive distance $ \mu - \text{margin}$ (superiority) or $\text{margin} - \mu $ (equivalence), in loss units.
sd	Optional standard deviation of the paired difference, loss units.
max_t	Search limit.

Details

- `sd = NULL` (default): the **guaranteed** horizon $t^*(\Delta) = \min\{t : 2 w_t < |\Delta|\}$ of the predictable-plug-in Hoeffding boundary, whose half-width w_t is deterministic. On the coverage event (probability at least $1 - \alpha$) the Hoeffding comparison has stopped with the correct declaration by then (theory note, Prop. 2). It is distribution-free and, for small margins, very conservative: it can exceed what the default betting boundary needs by two or three orders of magnitude.
- `sd` given (standard deviation of the paired difference, loss units): an **approximation** that plugs `sd` into the empirical-Bernstein half-width in place of the running variance estimate. It is not a bound; it is the order of magnitude a variance-adaptive boundary needs, and the betting boundary is usually faster still. Use a pilot or a conservative guess for `sd`.

Both charge the full half-width twice (worst-case position of the centre); typical stopping times are about half of the returned value, as measured in the package's pilot study. Both are computable before any data are collected and serve to size budget and `n_max`.

Value

An integer number of instances, or Inf if not reached by `max_t`.

Examples

```
design <- comparison_design(margin = 0.02, bounds = c(0, 1))
planning_horizon(design, distance = 0.05)           # guaranteed, Hoeffding
planning_horizon(design, distance = 0.05, sd = 0.1) # variance-based approximation
```

seqbench_boundaries	<i>Names of the available boundaries</i>
---------------------	--

Description

Names of the available boundaries

Usage

```
seqbench_boundaries()
```

Value

A character vector. "betting" is the default in `comparison_design()`; "hoeffding" and "empirical_bernstein" are conservative references; "bernstein_declared" requires a declared upper bound on the standard deviation of the paired difference and is valid only if that bound holds; "naive_fixed" is an invalid negative control for experiments.

Examples

```
seqbench_boundaries()
```

stopping_decision	<i>Current decision of a comparison</i>
-------------------	---

Description

Current decision of a comparison

Usage

```
stopping_decision(state)
```

Arguments

state A seqbench_comparison.

Value

A character scalar: "A" (A relevantly better), "B", "equivalent", "continue" or "inconclusive", with attribute reason ("declaration", "budget", "n_max", "cs_empty" or NA).

Examples

```
state <- initialize_comparison(comparison_design(margin = 0.02, bounds = c(0, 1)))
stopping_decision(state)
```

tidy.seqbench_comparison

Tidy the trajectory of a comparison

Description

One row per instance with the estimate, the confidence sequence at that time, its running intersection, cumulative cost and the decision the procedure would have taken there.

Usage

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

Arguments

x	A seqbench_comparison.
...	Unused.

Value

A tibble with columns t, instance, n_seeds, x, estimate, lower, upper, lower_running, upper_running, cost, cost_cum, decision.

Examples

```
design <- comparison_design(margin = 0.05, bounds = c(0, 1), boundary = "hoeffding")
state <- update_comparison(initialize_comparison(design),
  data.frame(instance = 1:5, loss_a = c(.1, .2, .1, .3, .2), loss_b = c(.5, .6, .4, .7, .5)))
tidy(state)
```

update_comparison

Update a comparison with new paired losses

Description

Absorbs the losses of one or more new instances, in the order given, and advances the confidence sequence and the decision after each instance. The sequential unit is the instance: all rows of an instance (its seeds) are averaged into one observation. Updating stops at the first instance that triggers a terminal decision or exhausts the budget; rows after it are not consumed and a warning says so.

Usage

```
update_comparison(state, losses)
```

Arguments

state	A seqbench_comparison from <code>initialize_comparison()</code> .
losses	A data frame with columns instance, loss_a, loss_b, and optionally seed (default 1) and cost (default design\$cost_per_round per row). Losses must lie within the design bounds; instances must not have been seen before.

Value

The updated seqbench_comparison.

Examples

```
design <- comparison_design(margin = 0.05, bounds = c(0, 1), boundary = "hoeffding")
state <- initialize_comparison(design)
set.seed(1)
losses <- data.frame(instance = 1:50, loss_a = runif(50, 0, 0.5),
                     loss_b = runif(50, 0.3, 0.8))
state <- update_comparison(state, losses)
stopping_decision(state)
```

Index

`autoplot.seqbench_comparison`, 2

`comparison_design`, 3
`comparison_design()`, 6–8
`comparison_report`, 5
`comparison_report()`, 6

`glance()`, 6
`glance.seqbench_comparison`, 5

`initialize_comparison`, 6
`initialize_comparison()`, 3, 4, 10

`planning_horizon`, 7
`planning_horizon()`, 4

`seqbench_boundaries`, 8
`seqbench_boundaries()`, 3
`stopping_decision`, 8
`summary.seqbench_comparison`
 (`comparison_report`), 5

`tidy()`, 6
`tidy.seqbench_comparison`, 9

`update_comparison`, 9
`update_comparison()`, 3, 6