

Package ‘sbwadjust’

October 10, 2026

Title Stable Balancing Weights for Covariate Adjustment in Trials

Version 0.3.0

Description Covariate adjustment for randomized trials using stable balancing weights (SBW), as proposed by Irish, Zubizarreta, and Luedtke (2026) <[doi:10.48550/arXiv.2609.01638](https://doi.org/10.48550/arXiv.2609.01638)>. `sbw_weights()` fits exact-balance SBW for a two-arm study by a closed-form solve with a nonnegative quadratic-programming fallback; `sbw_estimate()` computes a closed menu of treatment-effect estimands (average treatment effect, relative risk, survival ratio, Mann-Whitney, quantile contrasts) from the fitted weights, with bootstrap confidence intervals.

License MIT + file LICENSE

URL <https://github.com/kaylairish/sbwadjust>

BugReports <https://github.com/kaylairish/sbwadjust/issues>

Encoding UTF-8

Language en-US

RoxygenNote 7.2.3

Imports graphics, quadprog, stats, survival

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Kayla Irish [aut, cre]

Maintainer Kayla Irish <kayla.a.irish@gmail.com>

Repository CRAN

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

Contents

<code>boot_km_ratio</code>	2
<code>get_sbws_for_study</code>	3
<code>get_weights_for_group_neg</code>	3

get_weights_for_group_nonneg	4
km_ratio_greenwood	4
plot.sbw_fit	5
sbw_estimate	6
sbw_weights	7
summary.sbw_fit	8

Index 10

boot_km_ratio	<i>Bootstrap CI for an SBW-weighted KM survival ratio</i>
---------------	---

Description

Bootstraps the standard error of `km_ratio_greenwood()`'s log-ratio, then builds either a Wald CI (log scale) or a percentile CI (log scale). If the point estimate or bootstrap SE come out non-finite, falls back to the unadjusted KM ratio and sets `MC_fail = TRUE`; if even the unadjusted ratio is undefined (a double-degenerate case with no valid fallback), throws an error rather than returning a nonsense finite estimate. If the SBW fit or weighted KM ratio fails on an individual bootstrap resample, the unadjusted KM ratio is used for that resample; `boot_fail_rate` reports the fraction of resamples where this happened.

Usage

```
boot_km_ratio(
  time,
  status,
  A,
  X_subset,
  t0,
  B = 1500,
  alpha = 0.05,
  ci_method = c("wald", "percentile"),
  seed = NULL
)
```

Arguments

<code>time</code>	Event/censoring time.
<code>status</code>	Event indicator (1 = event, 0 = censored).
<code>A</code>	Treatment indicator (0/1).
<code>X_subset</code>	Covariate data frame for the full study sample, used to fit the SBW weights.
<code>t0</code>	Time at which to evaluate the survival ratio.
<code>B</code>	Number of bootstrap replicates.
<code>alpha</code>	Significance level for the confidence interval (default 0.05).
<code>ci_method</code>	Either "wald" (default) or "percentile".
<code>seed</code>	Optional seed set at the start of the bootstrap.

Value

A list with MC_fail, log_est, est, se_log, ci_log, ci, boot_fail_rate, boot_n_finite_reps, and SBW clipping diagnostics (sbw_n_clipped_full, sbw_max_abs_clipped_full, sbw_n_clipped_boot_mean, sbw_n_clipped_boot_max, sbw_max_abs_clipped_boot_max).

get_sbws_for_study	<i>Study-level SBW weights: closed-form first, QP fallback if needed</i>
--------------------	--

Description

Tries the unconstrained closed-form solve first for each arm ([get_weights_for_group_neg\(\)](#)); if the result is nonnegative within floating-point tolerance, zeroes out negligible negative dust and returns it directly. Otherwise falls back to the nonnegative-QP solve ([get_weights_for_group_nonneg\(\)](#)).

Usage

```
get_sbws_for_study(X_subset, A)
```

Arguments

X_subset	Covariate data frame for the full study sample.
A	Treatment indicator (0/1) for the full study sample.

Value

A list with w (numeric vector of weights, one per row of X_subset), n_clipped, and max_abs_clipped (both 0 if the closed-form solve was used).

get_weights_for_group_neg	<i>Closed-form SBW group weights (negative weights allowed)</i>
---------------------------	---

Description

Solves for weights on one treatment-arm's covariate data that exactly balance its mean to the target X_n, minimizing sum of squared weights, via the closed-form solution (no nonnegativity constraint).

Usage

```
get_weights_for_group_neg(data, X_n)
```

Arguments

data	Covariate data frame for the group (one treatment arm).
X_n	Target covariate mean to balance to.

Value

Numeric vector of weights, one per row of data.

get_weights_for_group_nonneg	<i>Nonnegative-QP SBW group weights, with clipping diagnostics</i>
------------------------------	--

Description

Solves the same balance objective as [get_weights_for_group_neg\(\)](#), but constrained to nonnegative weights via quadratic programming.

Usage

```
get_weights_for_group_nonneg(data, X_n)
```

Arguments

- data Covariate data frame for the group (one treatment arm).
- X_n Target covariate mean to balance to.

Value

A list with w (numeric vector of weights), n_clipped, and max_abs_clipped (see [.clip_weights\(\)](#)).

km_ratio_greenwood	<i>Weighted KM survival ratio $S1(t0)/S0(t0)$ with a Greenwood-based CI</i>
--------------------	--

Description

Wald CI on the log scale, with $se(\log S) = se(S) / S$ for each arm and $se(S)$ from Greenwood's formula.

Usage

```
km_ratio_greenwood(time, status, A, t0, alpha = 0.05, weights = NULL)
```

Arguments

time	Event/censoring time.
status	Event indicator (1 = event, 0 = censored).
A	Treatment indicator (0/1; numeric, logical, character, or factor).
t0	Time at which to evaluate the survival ratio.
alpha	Significance level for the confidence interval (default 0.05).
weights	Optional case weights (e.g. SBW), passed to <code>survival::survfit()</code> . The weights are treated as fixed and known, so with estimated weights such as SBW, <code>se_log</code> and the CI ignore the uncertainty from estimating them; use <code>boot_km_ratio()</code> for inference.

Value

A list with `S0`, `S1`, `ratio` ($= S1(t0)/S0(t0)$), `log_ratio`, `se_log`, `ci_ratio`, and `ci_log`.

plot.sbw_fit	<i>Plot covariate balance before and after SBW weighting</i>
--------------	--

Description

Standardized mean differences (treated vs. control) for each balance covariate, unweighted and SBW-weighted.

Usage

```
## S3 method for class 'sbw_fit'
plot(x, ...)
```

Arguments

x	An sbw_fit from <code>sbw_weights()</code> .
...	Passed on to the underlying <code>graphics::plot()</code> call; these override the defaults (e.g. <code>main = "My trial"</code> , <code>xlim = c(-1, 1)</code>).

Value

x, invisibly.

sbw_estimate

*Estimate a treatment effect from a fitted sbw_fit object***Description**

Estimate a treatment effect from a fitted sbw_fit object

Usage

```
sbw_estimate(object, outcome, estimand, ...)
```

```
## S3 method for class 'sbw_fit'
sbw_estimate(
  object,
  outcome,
  estimand,
  data = NULL,
  probs = 0.5,
  horizon = NULL,
  B = 1500,
  alpha = 0.05,
  ci_method = c("wald", "percentile"),
  seed = NULL,
  ...
)
```

Arguments

object	An sbw_fit from sbw_weights() .
outcome	A one-sided-response formula naming the outcome, e.g. $Y \sim 1$ (or <code>Surv(time, status) ~ 1</code> for <code>estimand = "survival_ratio"</code> ; <code>Surv()</code> resolves without attaching the survival package).
estimand	One of "ATE", "RR" (relative risk, i.e. ratio of weighted arm means), "survival_ratio", "mann_whitney", "quantile_diff", or "quantile_ratio". No user-supplied functional is accepted; an uncovered estimand means: take <code>weights(object)</code> and run (and bootstrap) your own estimator.
...	Passed to methods.
data	Optional data frame holding the outcome, for when the weights were fit on base-line data before outcomes were available. Its rows must correspond one-to-one, in the same order, to the data passed to sbw_weights() ; any column the two share is checked for agreement. Defaults to the data the weights were fit on.
probs	Vector of probabilities in (0, 1), used when <code>estimand</code> is "quantile_diff" or "quantile_ratio". Default 0.5 (the median).
horizon	Time t_0 at which to evaluate the survival ratio; required for <code>estimand = "survival_ratio"</code> .
B	Number of bootstrap replicates.

alpha	Significance level for the confidence interval.
ci_method	Either "wald" (default) or "percentile".
seed	Optional seed set at the start of the bootstrap.

Details

The bootstrap resamples participants independently (a nonparametric row bootstrap), which is valid under simple randomization. It does not account for stratified or covariate-adaptive randomization (permuted blocks, minimization, biased coin); under those designs the confidence intervals may be miscalibrated.

For `estimand = "survival_ratio"`, if the SBW point estimate or its bootstrap standard error is non-finite, the unadjusted Kaplan-Meier ratio is returned instead, with a warning and `mc_fail = TRUE`.

Value

An object of class `sbw_estimate`: a list with `estimand`; `estimate` (named by `estimand`, or `q<p>` per quantile); `se`, the bootstrap standard error (on the log scale when `scale = "log"`); `ci`, a matrix with one row per estimate (lower, upper) on the estimate's own scale; `scale` ("log" for "RR", "quantile_ratio", and "survival_ratio", otherwise "identity"); `alpha`; and `B`. Most estimands also return `boot_fail_rate`, the fraction of bootstrap resamples that failed. "survival_ratio" instead returns `horizon`, `mc_fail`, and `detail` (the full `boot_km_ratio()` result, which includes its own `boot_fail_rate`).

Examples

```
set.seed(1)
n = 100
trial_baseline = data.frame(
  age = rnorm(n, 50, 10),
  region = sample(c("N", "S"), n, replace = TRUE),
  arm = rbinom(n, 1, 0.5)
)
# Design stage: fit weights before any outcomes exist.
sbw = sbw_weights(~ age + region, data = trial_baseline, treatment = arm)

# Analysis stage: outcomes arrive later, one row per participant, same order.
trial_outcomes = data.frame(Y = rbinom(n, 1, plogis(-1 + 0.02 * trial_baseline$age)))
sbw_estimate(sbw, Y ~ 1, estimand = "RR", data = trial_outcomes, B = 200, seed = 1)
```

sbw_weights

Fit stable balancing weights for a two-arm study (design stage)

Description

Formula-based front end to `get_sbws_for_study()`: balances the covariates named in `balance` (factors dummy-coded automatically) to the pooled-arm mean, exactly (imbalance tolerance fixed at zero), matching the paper's SBW specification. Intended to be called before unblinding, using only baseline covariates and the randomization assignment.

Usage

```
sbw_weights(balance, data, treatment)
```

Arguments

balance	One-sided formula naming the covariates to balance, e.g. <code>~ age + sex + bmi + region</code> .
data	Data frame containing the balance covariates and treatment.
treatment	Treatment assignment: either a column name in data, unquoted (<code>treatment = arm</code>) or quoted (<code>treatment = "arm"</code>), or a vector. Accepts numeric 0/1, logical, or a two-level factor/character vector. For a factor, the second level is treated (coded 1); for a character vector, the second in alphabetical order. Use a factor with explicit levels to control this.

Value

An object of class `sbw_fit` with elements `weights`, `n_clipped`, `max_abs_clipped` (see [get_sbws_for_study\(\)](#)), `balance` (the formula), `treatment_name`, `treatment_levels`, `treatment` (recoded 0/1), `X` (the balance design matrix), `data`, `n`, and `call`.

Examples

```
set.seed(1)
n = 100
trial_baseline = data.frame(
  age = rnorm(n, 50, 10),
  region = sample(c("N", "S"), n, replace = TRUE),
  arm = rbinom(n, 1, 0.5)
)
sbw = sbw_weights(~ age + region, data = trial_baseline, treatment = arm)
sbw
summary(sbw)
```

summary.sbw_fit	<i>Summarize an sbw_fit: balance table and weight diagnostics</i>
-----------------	---

Description

Summarize an `sbw_fit`: balance table and weight diagnostics

Usage

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

Arguments

<code>object</code>	An <code>sbw_fit</code> from <code>sbw_weights()</code> .
<code>...</code>	Currently unused.

Value

An object of class `summary.sbw_fit`, printed by `print.summary.sbw_fit()`, with elements `balance` (unweighted and SBW-weighted arm means of each balance column), `n_treated`, `n_control`, and `n_zero` (units with weight 0, by arm).

Index

`.clip_weights()`, [4](#)

`boot_km_ratio`, [2](#)
`boot_km_ratio()`, [5](#), [7](#)

`get_sbws_for_study`, [3](#)
`get_sbws_for_study()`, [7](#), [8](#)
`get_weights_for_group_neg`, [3](#)
`get_weights_for_group_neg()`, [3](#), [4](#)
`get_weights_for_group_nonneg`, [4](#)
`get_weights_for_group_nonneg()`, [3](#)

`km_ratio_greenwood`, [4](#)
`km_ratio_greenwood()`, [2](#)

`plot.sbw_fit`, [5](#)

`sbw_estimate`, [6](#)
`sbw_weights`, [7](#)
`sbw_weights()`, [5](#), [6](#), [9](#)
`summary.sbw_fit`, [8](#)