

Package ‘ratecalib’

June 25, 2026

Type Package

Title Calibration Weighting to Multiple Subgroup Pass-Rate Targets

Version 0.3.0

Description Calibration weighting for binary-outcome pass rates against multiple overlapping subgroup targets. Adjusts initial positive weights so that the overall pass rate and subgroup pass rates approach (soft mode) or exactly match (exact mode) given targets, while preserving the initial weight structure and population margins. Provides a one-step interface, pre-solve data checks, target-table construction, effective sample size and design-effect diagnostics, and example data. The solver works on a bounded convex quadratic program over demographic-cell-by-outcome aggregates for efficiency on large samples. Methods follow the calibration approach of Deville and Saerndal (1992) <[doi:10.1080/01621459.1992.10475217](https://doi.org/10.1080/01621459.1992.10475217)>.

License MIT + file LICENSE

URL <https://github.com/makunxiang-cmd/ratecalib>

BugReports <https://github.com/makunxiang-cmd/ratecalib/issues>

Encoding UTF-8

Depends R (>= 4.1.0)

Imports Matrix, methods, osqp, stats

Suggests testthat (>= 3.0.0), openxlsx

Config/testthat/edition 3

RoxygenNote 8.0.0

NeedsCompilation no

Author Kunxiang Ma [aut, cre]

Maintainer Kunxiang Ma <mkx07080412@gmail.com>

Repository CRAN

Date/Publication 2026-06-25 15:10:02 UTC

Contents

calibrate_from_excel	2
calibrate_pass_rates	3
calibrate_rates	4
calibrate_replicate_weights	6
calibration_diagnostics	7
calibration_feasibility	8
check_calibration_data	9
example_rate_data	10
export_calibration_xlsx	11
make_rate_targets	11
pass_rate_calibration-methods	13
read_calibration_data	14
read_targets_xlsx	15
replicate_variance	15
Index	17

calibrate_from_excel *Calibrate directly from an Excel workbook*

Description

One-step convenience: read the sample data and the target table from an Excel workbook, infer the grouping variables from the targets, and solve.

Usage

```
calibrate_from_excel(
  path,
  outcome,
  weight,
  data_sheet = 1,
  targets_sheet = "targets",
  group_vars = NULL,
  ...
)
```

Arguments

path	Path to an .xlsx file.
outcome	Name of the binary outcome column.
weight	Name of the initial weight column.
data_sheet	Sheet holding the sample data (default 1).
targets_sheet	Sheet holding the target table (default "targets").
group_vars	Optional grouping-variable names; inferred from the target table when NULL.
...	Further arguments passed to <code>calibrate_pass_rates()</code> .

Value

An object of class `pass_rate_calibration`.

Examples

```
if (requireNamespace("openxlsx", quietly = TRUE)) {
  path <- tempfile(fileext = ".xlsx")
  d <- example_rate_data(300)
  tg <- make_rate_targets(groups = list(sex = c(M = 0.72, F = 0.68)))
  openxlsx::write.xlsx(list(data = d, targets = tg), path)
  fit <- calibrate_from_excel(path, "qualified", "initial_weight",
                             data_sheet = "data", targets_sheet = "targets")
  fit$target_check
}
```

`calibrate_pass_rates` *Calibrate weights to multiple pass-rate targets*

Description

Adjust initial positive weights so that an overall binary outcome rate and subgroup outcome rates approach or exactly match specified targets. The optimization is performed on demographic-cell-by-outcome aggregates.

Usage

```
calibrate_pass_rates(
  data,
  outcome,
  weight,
  group_vars,
  targets,
  lower = 0.25,
  upper = 4,
  mode = c("soft", "exact"),
  distance = c("chi2", "raking", "logit"),
  lambda = 10000,
  new_weight = "weight_calibrated",
  verbose = FALSE
)
```

Arguments

<code>data</code>	A data frame containing one row per sampled unit.
<code>outcome</code>	Name of a binary 0/1 outcome column.
<code>weight</code>	Name of the initial positive weight column.
<code>group_vars</code>	Character vector naming grouping variables.

targets	A data frame with columns variable, level, and target_rate; optional column priority controls soft-mode importance.
lower, upper	Scalar lower and upper bounds on the multiplier applied to each initial cell weight.
mode	Either "soft" or "exact".
distance	Distance function of the calibration family. "chi2" (the default) is the linear/quadratic distance solved as a bounded QP via OSQP and reproduces earlier behaviour. "raking" is the entropy distance $g \log g - g + 1$, whose solution $g = \exp(\eta)$ is strictly positive by construction; it is solved by a dual Newton iteration and currently supports mode = "exact" only. Raking is unbounded above, so lower and upper are not enforced for it but multipliers outside that range are reported in the diagnostics. "logit" is the bounded logit distance whose multipliers stay strictly inside (lower, upper) by construction (requires $\text{lower} < 1 < \text{upper}$); it is also dual-Newton solved and mode = "exact" only. Use "logit" when capping extreme weights is a hard requirement.
lambda	Positive soft-constraint penalty. Larger values emphasize target matching more strongly.
new_weight	Name of the calibrated weight column added to data.
verbose	Logical; passed to OSQP.

Value

An object of class `pass_rate_calibration`.

Examples

```
d <- example_rate_data(300)
targets <- make_rate_targets(groups = list(sex = c(M = 0.72, F = 0.68)))
fit <- calibrate_pass_rates(d, "qualified", "initial_weight",
                           group_vars = "sex", targets = targets)
fit$target_check
```

calibrate_rates

One-step pass-rate weight calibration

Description

A convenience interface for everyday use. The user only supplies the data, the outcome variable, the initial weights, an overall target and per-group targets; the function builds the target table, identifies grouping variables, runs the data checks and calls `calibrate_pass_rates()`.

Usage

```
calibrate_rates(
  data,
  outcome,
  weight,
  overall = NULL,
  groups = list(),
  priority = 5,
  group_priority = 1,
  lower = 0.25,
  upper = 4,
  mode = c("soft", "exact"),
  distance = c("chi2", "raking", "logit"),
  lambda = 10000,
  new_weight = "weight_calibrated",
  check = TRUE,
  verbose = FALSE
)
```

Arguments

data	A data frame with one row per sampled unit.
outcome	Name of the binary outcome column (1 = pass, 0 = fail).
weight	Name of the initial weight column.
overall	Optional scalar overall target rate; may be NULL.
groups	Named list. Each element name is a grouping variable and each element is a named numeric vector of target rates.
priority	Priority of the overall target. Defaults to 5.
group_priority	Priority of the group targets; a single number or a vector named by grouping variable.
lower, upper	Lower and upper bounds on the weight-adjustment multiplier.
mode	"soft" for soft constraints, "exact" for exact constraints.
distance	Calibration distance, passed to <code>calibrate_pass_rates()</code> . "chi2" (default), "raking" (entropy) or "logit" (bounded); the latter two are exact mode only.
lambda	Soft-constraint penalty strength.
new_weight	Name of the new calibrated weight column.
check	Whether to run the data checks before solving.
verbose	Whether to print OSQP solver information.

Value

An object of class `pass_rate_calibration`.

Examples

```
d <- example_rate_data(300)
fit <- calibrate_rates(d, "qualified", "initial_weight",
                      groups = list(sex = c(M = 0.72, F = 0.68)))
summary(fit)
```

```
calibrate_replicate_weights
```

```
Re-calibrate replicate weights
```

Description

Re-runs the calibration of a fitted `pass_rate_calibration` object for each set of replicate weights (e.g. bootstrap, jackknife or BRR weights produced elsewhere), holding the targets and solver settings fixed. The resulting calibrated replicate weights feed `replicate_variance()` for design-consistent variance estimation, mirroring the replicate-weights approach of the survey package.

Usage

```
calibrate_replicate_weights(
  fit,
  repweights,
  scale = 1,
  rscales = NULL,
  progress = FALSE
)

## S3 method for class 'replicate_calibration'
print(x, ...)
```

Arguments

<code>fit</code>	A fitted object of class <code>pass_rate_calibration</code> .
<code>repweights</code>	A numeric matrix or data frame of replicate weights with one row per observation (matching <code>fit</code>) and one column per replicate. All entries must be finite and positive.
<code>scale, rscales</code>	Replication variance constants, as in <code>survey::svrepdesign()</code> . $\text{Var} = \text{scale} * \text{sum}_r \text{rscales}_r * (\text{th})$ Set them to match your replication scheme (for example JK1: $\text{scale} = (R-1)/R$, $\text{rscales} = 1$; bootstrap: $\text{scale} = 1/R$). <code>rscales</code> defaults to a vector of ones.
<code>progress</code>	Logical; show a text progress bar over the replicates.
<code>x</code>	A <code>replicate_calibration</code> object (for the print method).
<code>...</code>	Ignored.

Value

An object of class `replicate_calibration` with the full-sample calibrated weights, the matrix of calibrated `replicate_weights`, and the `scale/rscales` constants.

See Also

[replicate_variance\(\)](#)

Examples

```
d <- example_rate_data(300)
fit <- calibrate_rates(d, "qualified", "initial_weight",
                      groups = list(sex = c(M = 0.72, F = 0.68)))
repw <- d$initial_weight * matrix(stats::runif(nrow(d) * 5, 0.8, 1.2), ncol = 5)
rc <- calibrate_replicate_weights(fit, repw)
replicate_variance(rc, d$qualified, statistic = "mean")
```

calibration_diagnostics

Extract calibration diagnostics

Description

Extract calibration diagnostics

Usage

```
calibration_diagnostics(x, sort_targets = TRUE)
```

Arguments

x A `pass_rate_calibration` object.

sort_targets Whether to sort targets by absolute error, descending.

Value

A list with target, margin and weight diagnostics.

Examples

```
d <- example_rate_data(300)
fit <- calibrate_rates(d, "qualified", "initial_weight",
                      groups = list(sex = c(M = 0.72, F = 0.68)))
calibration_diagnostics(fit)
```

 calibration_feasibility

Pre-solve target feasibility checks

Description

Runs two deterministic, closed-form feasibility checks before calibration. Both rely only on the initial weight marginals that the solver preserves, so they are cheap and exact within their stated scope.

Usage

```
calibration_feasibility(
  data,
  outcome,
  weight,
  group_vars,
  targets,
  lower = 0.25,
  upper = 4,
  tol = 1e-08
)

## S3 method for class 'ratecalib_feasibility'
print(x, ...)
```

Arguments

data	A data frame with one row per sampled unit.
outcome	Name of the binary outcome column (1 = pass, 0 = fail).
weight	Name of the initial weight column.
group_vars	Character vector of grouping-variable names.
targets	Target table with columns variable, level, target_rate.
lower, upper	Lower and upper bounds on the weight-adjustment multiplier.
tol	Numeric tolerance for the consistency comparison.
x	A ratecalib_feasibility object (for the print method).
...	Further arguments (ignored by the print method).

Details

1. **Overall-vs-group consistency.** Marginal totals are held fixed during calibration, so if every level of a grouping variable carries an exact target, the overall rate is uniquely pinned to $\sum_{\ell} W_{\ell} r_{\ell} / W$. Two such "complete" variables, or one plus an explicit overall target, can disagree; that disagreement is a guaranteed conflict under mode = "exact".

2. **Single-target marginal interval.** With the group total fixed and per-unit multipliers bounded by [lower, upper], a group's reachable weighted rate lies in a closed interval (two-block water-filling). A target outside that interval can never be met. This is a *necessary* condition only: passing every single-target check does not guarantee the targets are jointly feasible, because overlapping units couple the groups.

Value

A list of class `ratecalib_feasibility` with elements `consistency` (a list with `pins`, `consistent` and `detail`), `marginal` (a data frame of per-target achievable intervals), `necessary_ok` (logical) and `note`.

Examples

```
d <- example_rate_data(300)
targets <- make_rate_targets(overall = 0.62,
                             groups = list(sex = c(M = 0.66, F = 0.60)))
calibration_feasibility(d, "qualified", "initial_weight", "sex", targets)
```

check_calibration_data

Pre-calibration data checks

Description

Checks variables, weights, the binary outcome, group coverage and target supportability, and reports the current weighted pass rates.

Usage

```
check_calibration_data(
  data,
  outcome,
  weight,
  group_vars,
  targets = NULL,
  consistency_tol = 0.01
)

## S3 method for class 'ratecalib_check'
print(x, ...)
```

Arguments

<code>data</code>	A data frame.
<code>outcome</code>	Name of the binary outcome column.
<code>weight</code>	Name of the initial weight column.

group_vars	Character vector of grouping-variable names.
targets	Optional target table.
consistency_tol	Tolerance (on the rate scale) for the overall-vs-group consistency warning. Only inconsistencies larger than this are reported, so that sub-tolerance rounding (round-number targets that do not divide the weighted marginals exactly) does not trigger noise. For a precise, exact-mode feasibility analysis call <code>calibration_feasibility()</code> directly.
x	A <code>ratecalib_check</code> object (for the print method).
...	Further arguments (ignored by the print method).

Value

A list of class `ratecalib_check` with `ok`, `errors`, `warnings`, `overview`, `group_summary` and `target_support`.

Examples

```
d <- example_rate_data(300)
check_calibration_data(d, "qualified", "initial_weight", group_vars = "sex")
```

example_rate_data	<i>Generate example data</i>
-------------------	------------------------------

Description

Creates a simulated data set with sex, residence, a 5-level education variable, a 5-level age variable, a pass indicator and initial weights.

Usage

```
example_rate_data(n = 5000L, seed = 2026L)
```

Arguments

n	Sample size.
seed	Random seed.

Value

A data frame.

Examples

```
d <- example_rate_data(200)
head(d)
```

`export_calibration_xlsx`*Export a calibration result to an Excel workbook*

Description

Writes a multi-sheet workbook: data (with the calibrated weight column), target_check, margin_check, diagnostics and settings.

Usage

```
export_calibration_xlsx(fit, path, overwrite = TRUE)
```

Arguments

<code>fit</code>	An object of class <code>pass_rate_calibration</code> .
<code>path</code>	Output .xlsx path.
<code>overwrite</code>	Whether to overwrite an existing file.

Value

path, invisibly.

Examples

```
if (requireNamespace("openxlsx", quietly = TRUE)) {  
  d <- example_rate_data(300)  
  fit <- calibrate_rates(d, "qualified", "initial_weight",  
                        groups = list(sex = c(M = 0.72, F = 0.68)))  
  export_calibration_xlsx(fit, tempfile(fileext = ".xlsx"))  
}
```

`make_rate_targets`*Build a pass-rate target table*

Description

Build a pass-rate target table

Usage

```
make_rate_targets(
  overall = NULL,
  groups = list(),
  interactions = list(),
  overall_priority = 5,
  group_priority = 1,
  interaction_priority = 1,
  means = NULL,
  totals = NULL,
  proportions = NULL
)
```

Arguments

overall	Optional scalar overall target rate.
groups	Named list. Each element is a named numeric vector containing target rates for one grouping variable.
interactions	Named list of cross-classification (interaction) targets. Each element name is a colon-joined set of grouping variables (e.g. "sex:residence") and each element is a named numeric vector whose names are the matching colon-joined level combinations (e.g. c("M:Urban" = 0.7)).
overall_priority	Positive priority for the overall target.
group_priority	Either one positive scalar or a named positive vector indexed by group-variable name.
interaction_priority	Either one positive scalar or a named positive vector indexed by interaction key.
means, totals	Optional data frames of mean/total targets for a numeric variable, each with columns variable, level, value_var and target (plus optional priority). When supplied, the result gains statistic, value_var and value columns; proportion rows are tagged statistic = "proportion".
proportions	Optional data frame of proportion targets for a value of a categorical variable, with columns variable, level, value_var, value and target (plus optional priority).

Value

A data frame suitable for `calibrate_pass_rates()`.

Examples

```
make_rate_targets(overall = 0.70, groups = list(sex = c(M = 0.72, F = 0.68)))
```

 pass_rate_calibration-methods

Methods for pass_rate_calibration objects

Description

S3 methods that print, summarize and plot calibration results.

Usage

```
## S3 method for class 'pass_rate_calibration'
weights(object, ...)

## S3 method for class 'pass_rate_calibration'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'pass_rate_calibration'
print(x, digits = 4, ...)

## S3 method for class 'pass_rate_calibration'
summary(object, top = 10L, ...)

## S3 method for class 'summary_pass_rate_calibration'
print(x, digits = 4, ...)

## S3 method for class 'pass_rate_calibration'
plot(x, type = c("target_error", "multipliers"), top = 20L, ...)
```

Arguments

...	Further arguments passed to the underlying print or graphics functions.
x, object	A <code>pass_rate_calibration</code> object (or, for the summary printer, a <code>summary_pass_rate_calibration</code> object).
row.names, optional	Standard <code>as.data.frame()</code> arguments (ignored).
digits	Number of decimal places to display.
top	Number of largest-error targets to display.
type	Plot type: "target_error" for target errors or "multipliers" for the weight-multiplier histogram.

Value

`print` and `plot` return their input invisibly; `summary` returns a `summary_pass_rate_calibration` object; `weights` returns the calibrated weight vector; `as.data.frame` returns the data with the calibrated weight column.

Examples

```
d <- example_rate_data(300)
fit <- calibrate_rates(d, "qualified", "initial_weight",
                      groups = list(sex = c(M = 0.72, F = 0.68)))

print(fit)
summary(fit)
head(weights(fit))
head(as.data.frame(fit))
plot(fit, type = "target_error")
```

read_calibration_data *Read calibration sample data from an Excel workbook*

Description

Read calibration sample data from an Excel workbook

Usage

```
read_calibration_data(path, sheet = 1)
```

Arguments

path	Path to an .xlsx file.
sheet	Sheet name or index (default 1).

Value

A data frame with one row per sampled unit.

Examples

```
if (requireNamespace("openxlsx", quietly = TRUE)) {
  path <- tempfile(fileext = ".xlsx")
  openxlsx::write.xlsx(example_rate_data(50), path)
  head(read_calibration_data(path))
}
```

read_targets_xlsx	<i>Read a pass-rate target table from an Excel workbook</i>
-------------------	---

Description

Reads a worksheet of targets and maps its headers (English or Chinese, in any letter case) onto the canonical columns variable, level, target_rate and the optional priority.

Usage

```
read_targets_xlsx(path, sheet = 1)
```

Arguments

path	Path to an .xlsx file.
sheet	Sheet name or index (default 1).

Value

A data frame suitable for [calibrate_pass_rates\(\)](#).

Examples

```
if (requireNamespace("openxlsx", quietly = TRUE)) {
  path <- tempfile(fileext = ".xlsx")
  openxlsx::write.xlsx(
    make_rate_targets(groups = list(sex = c(M = 0.72, F = 0.68))), path)
  read_targets_xlsx(path)
}
```

replicate_variance	<i>Replicate-weight variance of a calibrated estimate</i>
--------------------	---

Description

Computes the point estimate, variance and standard error of a weighted total or mean of a study variable, using calibrated replicate weights.

Usage

```
replicate_variance(object, x, statistic = c("total", "mean"))
```

Arguments

object	A replicate_calibration object from calibrate_replicate_weights() .
x	Numeric study variable, one value per observation.
statistic	Either "total" (weighted sum) or "mean" (weighted mean).

Value

A list with estimate, variance and se.

Examples

```
d <- example_rate_data(300)
fit <- calibrate_rates(d, "qualified", "initial_weight",
                      groups = list(sex = c(M = 0.72, F = 0.68)))
repw <- d$initial_weight * matrix(stats::runif(nrow(d) * 5, 0.8, 1.2), ncol = 5)
rc <- calibrate_replicate_weights(fit, repw)
replicate_variance(rc, d$qualified, statistic = "mean")
```

Index

`as.data.frame()`, [13](#)
`as.data.frame.pass_rate_calibration`
 (`pass_rate_calibration-methods`),
 [13](#)

`calibrate_from_excel`, [2](#)
`calibrate_pass_rates`, [3](#)
`calibrate_pass_rates()`, [2](#), [4](#), [5](#), [15](#)
`calibrate_rates`, [4](#)
`calibrate_replicate_weights`, [6](#)
`calibrate_replicate_weights()`, [15](#)
`calibration_diagnostics`, [7](#)
`calibration_feasibility`, [8](#)
`calibration_feasibility()`, [10](#)
`check_calibration_data`, [9](#)

`example_rate_data`, [10](#)
`export_calibration_xlsx`, [11](#)

`make_rate_targets`, [11](#)

`pass_rate_calibration-methods`, [13](#)
`plot.pass_rate_calibration`
 (`pass_rate_calibration-methods`),
 [13](#)

`print.pass_rate_calibration`
 (`pass_rate_calibration-methods`),
 [13](#)

`print.ratecalib_check`
 (`check_calibration_data`), [9](#)
`print.ratecalib_feasibility`
 (`calibration_feasibility`), [8](#)
`print.replicate_calibration`
 (`calibrate_replicate_weights`),
 [6](#)

`print.summary_pass_rate_calibration`
 (`pass_rate_calibration-methods`),
 [13](#)

`read_calibration_data`, [14](#)
`read_targets_xlsx`, [15](#)

`replicate_variance`, [15](#)
`replicate_variance()`, [6](#), [7](#)

`summary.pass_rate_calibration`
 (`pass_rate_calibration-methods`),
 [13](#)

`weights.pass_rate_calibration`
 (`pass_rate_calibration-methods`),
 [13](#)