Package: regrake (via r-universe)

May 14, 2026

Type Package

Title Regularized Survey Raking

Version 1.0.0

Description Calibrates survey weights to known population targets using regularized raking. Constraints are specified with a formula interface (for example, rr_exact(), rr_l2(), rr_range(), rr_mean(), rr_var(), and rr_quantile()). Supports common target formats including autumn-style proportions tables, raw or weighted population microdata, named-list targets (as in 'anesrake'), and 'survey' package design objects. Optimization follows Barratt et al. (2021) <https://web.stanford.edu/~boyd/papers/pdf/optimal_representative_sampling.pdf> and returns calibrated weights with balance and convergence diagnostics.

License Apache License (== 2.0) | file LICENSE

Language en-US

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports Matrix, rlang (>= 1.0.0), stats, lamW, tibble, digest

Suggests dplyr, survey, testthat (>= 3.0.0), knitr, rmarkdown, CVXR, RhpcBLASctl

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/andytimm/regrake

BugReports https://github.com/andytimm/regrake/issues

Config/pak/sysreqs make

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

Date/Publication 2026-03-15 17:30:05 UTC

RemoteUrl https://github.com/andytimm/regrake

RemoteRef HEAD

RemoteSha ba8be4d3546e09c87bf75626e2ef5ed2b578a76c

Contents

`print.raking_formula` *Print method for raking_formula objects*

Description

Displays a human-readable representation of a raking formula. This function shows each term in the formula in a structured format, making it easy to understand complex formulas with multiple constraints.

Usage

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

Arguments

`x`	A raking_formula object
`...`	Additional arguments passed to other methods

Value

Invisibly returns the object

print.regrake *Print method for regrake objects*

Description

Print method for regrake objects

Usage

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

Arguments

x	A regrake object
...	Additional arguments (ignored)

Value

Invisibly returns the object

prox_boolean_reg *Proximal operator for boolean regularizer*

Description

Selects top-k weights and assigns them equal weight 1/k. All other weights are set to zero. This is used for representative sample selection where exactly k samples should be selected.

Usage

```
prox_boolean_reg(w, lam, k)
```

Arguments

w	Input vector of weights
lam	Regularization parameter (unused, kept for interface consistency)
k	Number of samples to select

Value

Vector with k non-zero entries, each equal to 1/k

prox_equality *Proximal operator for equality constraints*

Description

Proximal operator for equality constraints

Usage

```
prox_equality(x, target, rho)
```

Arguments

x	Input vector
target	Target vector
rho	Proximal parameter (unused for equality constraints)

Value

Projected vector equal to target

prox_equality_reg *Proximal operator for equality regularizer*

Description

Proximal operator for equality regularizer

Usage

```
prox_equality_reg(w, lam)
```

Arguments

w	Input vector
lam	Regularization parameter

Value

Original vector (identity operation)

prox_inequality *Proximal operator for inequality constraints*

Description

Proximal operator for inequality constraints

Usage

```
prox_inequality(x, target, rho, lower, upper)
```

Arguments

x	Input vector
target	Target vector (used for offset)
rho	Proximal parameter (unused for inequality constraints)
lower	Lower bound
upper	Upper bound

Value

Clipped vector within bounds relative to target

prox_kl *Proximal operator for KL divergence loss*

Description

Proximal operator for KL divergence loss

Usage

```
prox_kl(x, target, rho, scale = 0.5)
```

Arguments

x	Input vector
target	Target vector
rho	Proximal parameter
scale	Scale factor for KL divergence (default 0.5, matching the Python reference)

Value

Updated vector minimizing KL divergence plus proximal term

prox_kl_reg *Proximal operator for KL regularizer*

Description

Proximal operator for KL regularizer

Usage

```
prox_kl_reg(w, lam, prior = NULL, limit = NULL)
```

Arguments

w	Input vector
lam	Regularization parameter
prior	Prior weights (default uniform)
limit	Optional upper bound on weight magnitudes

Value

Updated vector minimizing KL divergence plus proximal term

prox_least_squares *Proximal operator for least squares loss*

Description

Proximal operator for least squares loss

Usage

```
prox_least_squares(x, target, tau, diag_weight = 1)
```

Arguments

x	Input vector
target	Target vector
tau	Proximal parameter (1/rho)
diag_weight	Numeric scalar or vector of weights for each element (default 1)

Value

Updated vector minimizing weighted quadratic plus proximal term

prox_sum_squares_reg *Proximal operator for sum squares regularizer*

Description

Proximal operator for sum squares regularizer

Usage

```
prox_sum_squares_reg(w, lam)
```

Arguments

w	Input vector
lam	Regularization parameter

Value

Updated vector minimizing sum squares plus proximal term

regrake *Optimal representative sample weighting*

Description

Optimal representative sample weighting

Usage

```
regrake(
  data,
  formula,
  population_data,
  pop_type = c("raw", "weighted", "proportions", "anesrake", "survey", "survey_design"),
  pop_weights = NULL,
  regularizer = "entropy",
  lambda = 1,
  prior = NULL,
  k = NULL,
  bounds = c(0.1, 10),
  bounds_method = c("soft", "hard"),
  exact_tol = NULL,
  normalize = TRUE,
  control = list(),
  verbose = FALSE,
  ...
)
```

Arguments

`data`	A data.frame or tibble containing the sample data
`formula`	A formula specifying the raking constraints (e.g., `~ rr_exact(sex) + rr_l2(age)`)
`population_data`	Population data: a data.frame, list, or survey.design object (see `pop_type`)
`pop_type`	How population data is specified: • "raw": Raw population data (one row per unit) • "weighted": Population data with weights column • "proportions": Direct specification of target proportions (variable, level, target columns) • "anesrake": List of named numeric vectors (anesrake package format) • "survey": Data frame with margin, category, value columns • "survey_design": survey package design object
`pop_weights`	Column name in population_data containing weights (if pop_type = "weighted")
`regularizer`	Regularization method ("entropy", "zero", "kl", or "boolean")
`lambda`	Regularization strength (default = 1)
`prior`	Optional prior weights used when `regularizer = "kl"`. Must be a positive numeric vector of length `nrow(data)`. If it does not sum to 1, it is normalized internally.
`k`	Number of samples to select (required for regularizer = "boolean")
`bounds`	Numeric vector of length 2 specifying (min, max) allowed weight values. Weights returned sum to n (sample size), so `bounds = c(0.3, 3)` means each weight is between 0.3 and 3 times the "average" weight of 1. Default is `c(0.1, 10)`.
`bounds_method`	How to enforce bounds: **"soft"** (Default) Uses regularizer clipping. Fast but bounds may be slightly violated when targets conflict with bounds. Asymmetric bounds are approximated as symmetric. **"hard"** Uses bounded simplex projection. Bounds are strictly enforced but optimization may be slower and targets may be less closely matched when bounds are binding.
`exact_tol`	Optional tolerance for exact constraints. When non-NULL, all `rr_exact()` (and `rr_mean()`) constraints are converted to `rr_range()` with this margin. For example, `exact_tol = 0.02` means categorical proportions must be within +/- 2 percentage points of targets, and continuous means within +/- 0.02 of targets. Use `rr_range()` directly in the formula for per-variable control. Default is NULL (exact constraints enforced strictly).
`normalize`	Logical. If TRUE (default), continuous variables are automatically scaled by their target value for numerical stability. The achieved values are reported in original units. Set to FALSE to disable this behavior.
`control`	List of control parameters for the ADMM solver:

margin_tol Margin-based convergence tolerance (default 1e-4). The ADMM solver scales this by problem size to achieve approximately this level of margin accuracy regardless of sample size or constraint count. For example, `margin_tol = 0.001` targets ~0.1% max margin error. Internally computes `eps = margin_tol / sqrt(m + 2*n)`. Set to NULL to use raw eps_abs/eps_rel instead.

eps_abs Absolute convergence tolerance for ADMM residuals. Only used when margin_tol is NULL. Note: the effective tolerance scales with `sqrt(m + 2*n)`, so the same value behaves differently at different problem sizes. Default 1e-5.

eps_rel Relative convergence tolerance for ADMM residuals. Only used when margin_tol is NULL. Default 1e-5.

rho ADMM penalty parameter (default 50).

maxiter Maximum ADMM iterations (default 5000).

`verbose` Whether to print progress information

`...` Additional arguments passed to methods

Value

An object of class `"regrake"` containing:

`weights`	The optimal weights (sum to n)
`balance`	Data frame comparing achieved vs target values with columns: constraint (e.g., "exact_sex"), type ("exact" or "l2"), variable, level, achieved, target, residual
`solution`	Full solution details from solver
`diagnostics`	Weight, convergence, and margin matching diagnostics

Examples

```
set.seed(42)
sample_data <- data.frame(
  sex = sample(c("M", "F"), 200, replace = TRUE, prob = c(0.6, 0.4)),
  age = sample(c("young", "old"), 200, replace = TRUE, prob = c(0.7, 0.3))
)
pop_targets <- data.frame(
  variable = c("sex", "sex", "age", "age"),
  level = c("M", "F", "young", "old"),
  target = c(0.49, 0.51, 0.45, 0.55)
)
result <- regrake(
  data = sample_data,
  formula = ~ rr_exact(sex) + rr_exact(age),
  population_data = pop_targets,
  pop_type = "proportions"
)
result
result$balance
```

`rr_constraints` *Raking Constraint Functions*

Description

These functions specify constraint types for raking formulas. They are used within formula specifications passed to `regrake()`.

Usage

```
rr_l2(x)

rr_kl(x)

rr_exact(x)

rr_mean(x)

rr_var(x)

rr_quantile(x, p)

rr_range(x, ...)

rr_between(x, ...)
```

Arguments

`x`	Variable name (unquoted) to apply the constraint to
`p`	For `rr_quantile`, the quantile probability (0 to 1, e.g., 0.5 for median)
`...`	For `rr_range`/`rr_between`: either a single margin value (or named vector for level-specific margins), or separate lower and upper bounds. Examples: `rr_range(x, 0.02)`, `rr_range(x, c(A=0.01, B=0.02))`, `rr_range(x, 40, 45)`, `rr_range(x, lower=40, upper=45)`

Details

- `rr_exact()`: Exact equality constraint (weighted sum equals target exactly)
- `rr_l2()`: Soft L2/least squares constraint (penalizes deviation from target)
- `rr_kl()`: KL divergence constraint
- `rr_mean()`: Match the mean of a continuous variable (alias for rr_exact on continuous)
- `rr_var()`: Match the variance of a continuous variable
- `rr_quantile()`: Match a specific quantile of a continuous variable

Value

The input variable (these functions are markers for the formula parser)

Examples

```
# Match sex proportions exactly, age proportions with soft constraint
formula <- ~ rr_exact(sex) + rr_l2(age)

# Match categorical variable and continuous mean
formula <- ~ rr_exact(region) + rr_mean(income)

# Match mean and variance of a continuous variable
formula <- ~ rr_mean(income) + rr_var(income)

# Match median income
formula <- ~ rr_quantile(income, 0.5)
```

---

summary.regrake *Summary method for regrake objects*

Description

Summary method for regrake objects

Usage

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

Arguments

object	A regrake object
...	Additional arguments (ignored)

Value

Invisibly returns the object

Index