

Package: leafblower (via r-universe)

October 17, 2026

Title High-Performance Bounded Survey Weight Calibration

Version 0.1.1

Author Dennis Alexis Valin Dittrich [aut, cre]

Maintainer Dennis Alexis Valin Dittrich <davdittrich@gmail.com>

Description Drop-in replacement for `autumn::harvest()` providing multiple bounded survey-weight calibration solvers: ORIS (box-constrained Sinkhorn block-coordinate descent), raking, Sinkhorn, Greenhorn, GREG and logit (Deville-Sarndal linear and logit duals), Chebyshev interior-point, and Newton-KL. Adds `min_weight` and `max_weight` bounds; shares a single C++17 core with the Python bindings.

License MIT + file LICENSE

Encoding UTF-8

NeedsCompilation yes

SystemRequirements C++17 compiler, OpenMP (optional, for SIMD exp vectorisation)

Imports stats

Suggests testthat (>= 3.0.0), bench, lhs, DiceKriging, ggplot2, rprojroot, survey, PracTools (>= 1.4.0), arrow, callr, jsonlite, withr

RoxygenNote 7.3.3

Config/testthat/edition 3

Repository <https://davdittrich.r-universe.dev>

Date/Publication 2026-08-18 01:22:36 UTC

RemoteUrl <https://github.com/davdittrich/leafblower>

RemoteRef HEAD

RemoteSha a610d1d924f2191f362febeb9711edfc02da10cb

Contents

anesrake	2
design_effect	3
diagnose_weights	4
effective_sample_size	5
get_current_miss	5
harvest	6
weighted_pct	12
Index	13

anesrake	<i>anesrake compatibility wrapper</i>
----------	---------------------------------------

Description

Maps anesrake parameter names to harvest(). Unknown params produce warnings.

Usage

```
anesrake(  
  inputter,  
  targets,  
  weightvec = NULL,  
  caseid = NULL,  
  pctlim = 0.05,  
  cap = 5,  
  choosemethod = "rake",  
  type = "pctlim",  
  nlim = 500L,  
  iterate = TRUE,  
  threads = 1L,  
  ...  
)
```

Arguments

inputter	Data frame (maps to data).
targets	Named list of target proportions.
weightvec	Starting weights or NULL.
caseid	Ignored (row label not used by calibration).
pctlim	Deprecated; maps to convergence["pct"].
cap	Upper weight cap; maps to max_weight.
choosemethod	Algorithm; "rake" and "nrake" are silently remapped to "oris" (the recommended method).

type	Ignored.
nlim	Max iterations; maps to max_iterations.
iterate	Ignored (always iterates).
threads	Ignored (single-threaded v1).
...	Unknown params produce a warning then are ignored.

Value

Same as harvest().

design_effect	<i>Kish (1965) design effect (1-argument) or Henry & Valliant (2015) calibration design effect (4-argument)</i>
---------------	---

Description

When called with only weights, computes the Kish (1965) weighting design effect $\text{deff}_K(w) = n * \text{sum}(w^2) / \text{sum}(w)^2$.

Usage

```
design_effect(weights, outcome = NULL, data = NULL, target = NULL)
```

Arguments

weights	Numeric vector of calibrated weights. Must be finite, no NA, sum > 0.
outcome	Numeric outcome vector (optional; 4-arg form only). Must match length(weights), no NA, all finite.
data	Data frame with the calibration covariate columns named in target. Must have $\text{nrow}(\text{data}) == \text{length}(\text{weights})$. Each column named in target must contain only levels present in $\text{names}(\text{target}[[\text{var}]])$ and no NAs.
target	Named list of target proportions per categorical calibration margin (one named entry per margin variable in data). May be list().

Details

When called with all four arguments, computes the Henry & Valliant (2015) calibration design effect (Eq 3.5, zero-correlation approximation):

$$\text{deff}_H \approx \text{deff}_K(w) \cdot \hat{\sigma}_u^2 / \hat{\sigma}_y^2$$

where $\hat{u}_i = y_i - x_i^T \hat{\beta}$ are the GREG residuals from a survey- weighted least-squares fit of outcome on the calibration covariate matrix X (built from data + target), and $\hat{\sigma}_u^2, \hat{\sigma}_y^2$ are survey-weighted population unit variances per Eq 2.7. X is a constant column followed by reference-coded dummies (first level dropped per calibration margin): the constant is what makes $\hat{u}_i$ a true GREG residual and bounds $\text{deff}_H \leq \text{deff}_K(w)$, since the constant-only model is then nested in the calibration model.

Falls back to `deff_K(w)` when `target` is empty, when $\hat{\sigma}_y^2$ is zero, or when calibration margins are rank-deficient (with warning).

All numerical computation is performed in C++17 via `.Call(C_rk_design_effect, ...)`; R + Python wrappers produce byte-identical output by construction.

Value

Numeric scalar design effect (`deff_K` for 1-arg, `deff_H` for 4-arg).

References

Kish, L. (1965). *Survey Sampling*. Wiley.

Henry, K.A. and Valliant, R. (2015). A design effect measure for calibration weighting in single-stage samples. *Survey Methodology*, 41(2), 315-331. Statistics Canada Catalogue No. 12-001-X. Equation 3.5.

diagnose_weights	<i>Diagnose calibration quality</i>
------------------	-------------------------------------

Description

Returns a data frame comparing original and weighted marginals to targets.

Usage

```
diagnose_weights(data, target, weights)
```

Arguments

<code>data</code>	Data frame used in calibration.
<code>target</code>	Named list of target proportions (same format as <code>harvest()</code>).
<code>weights</code>	Numeric vector of calibrated weights, length <code>nrow(data)</code> .

Details

An "NA" entry in `target` is treated as the missing-data bin: a row counts toward it if it is NA *or* its value is the literal string "NA" (conflation), matching how `harvest(add_na_proportion = TRUE)` encodes the injected NA bin. For a hand-built target that names a real category "NA" without `add_na_proportion`, genuinely-missing rows are also counted there (a documented collision; see `?harvest`).

Value

Data frame with columns: `variable`, `level`, `prop_original`, `prop_weighted`, `target`, `error_original`, `error_weighted`.

effective_sample_size *Effective sample size given calibrated weights*

Description

Computes $\text{length}(\text{weights}) / \text{design_effect}(\text{weights})$ (Kish 1965 form).

Usage

```
effective_sample_size(weights)
```

Arguments

weights Numeric vector of calibrated weights. Must be finite, no NA, sum > 0.

Value

Numeric scalar effective sample size.

References

Kish, L. (1965). *Survey Sampling*. Wiley.

get_current_miss *Current calibration miss (matching autumn's export name)*

Description

Current calibration miss (matching autumn's export name)

Usage

```
get_current_miss(data, target, weights)
```

Arguments

data Data frame.
target Named list of target proportions.
weights Numeric weight vector.

Details

An "NA" entry in target is treated as the missing-data bin: a row counts toward it if it is NA *or* its value is the literal string "NA" (conflation), matching how `harvest(add_na_proportion = TRUE)` encodes the injected NA bin. For a hand-built target that names a real category "NA" without `add_na_proportion`, genuinely-missing rows are also counted there (a documented collision; see `?harvest`).

Value

Named numeric vector of max absolute errors per variable.

harvest

Generate calibrated weights (drop-in for autumn::harvest)

Description

Generate calibrated weights (drop-in for autumn::harvest)

Usage

```
harvest(
  data,
  target,
  min_weight = 0,
  max_weight = 5,
  capacity_penalty = NULL,
  alm_penalty = NULL,
  method = "oris",
  verbose = 0,
  max_iterations = 500,
  start_weights = NULL,
  attach_weights = TRUE,
  weight_column = "weights",
  convergence = list(),
  sor = NULL,
  bounds_mode = "cell",
  homotopy_levels = 1L,
  homotopy_start_factor = 1,
  homotopy_end_factor = 1,
  homotopy_budget_p = 0.5,
  scheduler = c("round_robin", "greedy"),
  eta_schedule = c("fixed", "tang_dynamic"),
  eta_start = 1,
  eta_end = 1,
  eta_schedule_power = 0.5,
  accelerate = FALSE,
  add_na_proportion = FALSE,
  auto_collapse = FALSE,
  collapse_vars = NULL,
  target_map = NULL,
  design_weights = NULL,
  newton_tsvd_ratio = 1e-08,
  ridge_lambda = 0,
  ...
)
```

Arguments

<code>data</code>	A data frame containing columns to calibrate.
<code>target</code>	A named list of named numeric vectors (variable -> proportions).
<code>min_weight</code>	Lower bound on weights. Default 0 (no lower bound).
<code>max_weight</code>	Upper bound on weights. Default 5.
<code>capacity_penalty</code>	Numeric, controls how strongly capacity bounds are enforced during ALM optimization in <code>method="oris_soft"</code> . Use NULL (default) for auto-computed value (<code>estimate\M_cell() / n</code> , capped at 1.0 for $K > 8$) which gives a balanced blend between unconstrained KL minimization and hard-clamp projection. Larger values force tighter constraint adherence; smaller values allow more temporary bound violation before the final projection enforces exact bounds. Tuning: if <code>attr(result, "result")\$alm_capacity_mu_final / capacity_penalty >= 1000</code> , the adaptive schedule hit its ceiling – increase <code>capacity_penalty</code> by 10x. Ignored for methods other than <code>"oris_soft"</code> .
<code>alm_penalty</code>	Numeric or NULL (default, disabled). When supplied it must be a positive finite scalar; values above $1e15$ are rejected and values below $1e-15$ warn that the objective penalty may be ineffective. Sets the augmented-Lagrangian penalty weight used by the ALM path.
<code>method</code>	Calibration method (ORIS: Over-Relaxed Iterative Scaling). One of <code>"auto"</code> (default: ORIS or raking based on <code>M_cell/n</code> ratio), <code>"oris"</code> (paper-faithful ORIS), <code>"oris_soft"</code> (ORIS with augmented Lagrangian soft capacity enforcement; better than <code>"oris"</code> on tight-bounds problems where cells hit <code>max_weight</code>), <code>"raking"</code> (IPF + water-filling box projection (KL projection, Csiszar-Tusnady 1984)), <code>"sinkhorn"</code> (KL Bregman Dykstra), <code>"greg"</code> (Newton QP, Deville-Sarndal 1992), <code>"chebyshev"</code> (L-infinity LP via IPM), <code>"greenkhorn"</code> (greedy coordinate-descent IPF – Altschuler-Weed-Rigollet 2017; picks the single hardest margin per step; supports <code>accelerate=TRUE</code> for SQUAREM round-level acceleration. Distinct from <code>scheduler="greedy"</code> on raking, which sorts all K margins but still sweeps all per round), <code>"logit"</code> (Deville-Sarndal 1992 logit-distance Newton calibration; bounds enforced analytically – no clamping; typically converges in 10-20 Newton steps; equivalent to <code>autumn::calibrate()</code>).
<code>verbose</code>	Integer verbosity: 0=silent, 1=progress, 2=debug.
<code>max_iterations</code>	Maximum inner BCD iterations per outer step. Default 500. Under <code>accelerate = TRUE</code> (SRAA) the budget is denominated consistently in FUNCTION EVALUATIONS across raking and greenkhorn (CR-C19): each accelerated step spends $\sim 1-2$ <code>f_evals</code> , one <code>f_eval</code> is a full margin sweep, and the reported <code>iterations</code> field counts <code>f_evals</code> for both. The pure (non-accelerated) paths report their native unit: greenkhorn counts single greedy margin steps, other solvers count full BCD sweeps.
<code>start_weights</code>	Starting weights vector or NULL (uniform).
<code>attach_weights</code>	If TRUE, return data frame with weights column. Default TRUE.
<code>weight_column</code>	Name of weight column. Default <code>"weights"</code> .
<code>convergence</code>	Named list controlling the stopping criterion. Accepted keys:

- **metric:** quality metric to monitor. One of "max_err" (default), "mean_err", "kl", "chi2", "grake_norm", "l1_weight", "marginal_kl".
- **rule:** stopping rule. One of "improvement" (default) – stop when metric improves by less than tol relative; "threshold" – stop when metric falls below tol absolutely; "plateau" – stop when metric changes less than tol over a window.
- **tol:** tolerance value (default 0.001).
- **stop_when:** "any" (default) or "all".

Shorthand keys (each sets metric+rule+tol simultaneously):

- **pct:** sets metric="l1_weight", rule="plateau", tol=pct. Example: `list(pct = 0.001)`.
- **absolute:** sets metric="max_err", rule="threshold", tol=absolute. Example: `list(absolute = 1e-4)`.
- **improvement:** sets metric="max_err", rule="improvement", tol=improvement. Example: `list(improvement = 0.001)`.

Shorthands and explicit keys may be combined; explicit keys override shorthand defaults. `convergence = list()` selects a per-method default metric equal to that solver's natural objective (applied only when none of metric, criterion, improvement, pct, or absolute is set; rule="improvement", tol = 0.001).

Defaults by method:

- **oris, oris_soft, auto:** marginal_kl (calibration quality: $\sum_k \sum_j t_{kj} \log(t_{kj}/\text{achieved}_{kj})$).
- **raking, greenkhorn, sinkhorn, newton_kl:** kl.
- **greg:** chi2.
- **chebyshev, logit:** max_err.

chi2 cross-solver note: chi2 is not directly comparable across methods. ORIS uses unnormalized cell mass as W_{total} ; raking uses n . Use chi2 as a convergence criterion within one method; do not compare values across methods.

chebyshev note: method="chebyshev" ignores rule (and metric/stop_when). The interior-point solver stops on its own complementarity-gap criterion, falling back to the pct/absolute tolerance on the max marginal error. Other convergence keys are no-ops for this method.

sor

Named list for SOR adaptive under-relaxation (ORIS and raking). Default NULL disables SOR. Keys:

- **auto:** logical, default TRUE.
- **omega_min:** lower bound on relaxation factor, default 0.3.
- **omega_max:** upper bound (recovery ceiling) on relaxation factor, default 1.5. Values in (1, 2) enable over-relaxation, which reduces iteration count; global convergence is guaranteed for all omega in (0, 2) (Thibault 2021). ORIS only; ignored for greedy scheduler.
- **omega:** fixed relaxation factor (disables auto-adapt).
- **burnin:** iterations before adaptation starts, default 20.
- **omega_mode_id:** omega adaptation strategy. 0 = heuristic (0.7 damp / 1.05 grow), 1 = fixed (jump to omega_max), 2 = iterate-change (free-coordinate

	$\ \Delta X_{\text{free}}\ ^2$ estimator, default ; feasibility-agnostic; e18t.9 SHIP – 240 vs 350 iters on T2 unconstrained, 50 vs 140 on bounded stepstone). String aliases "heuristic", "fixed", "spectral" are accepted.
bounds_mode	One of "cell" (default) or "unit". Controls per-observation vs cell-aggregate bound enforcement.
homotopy_levels	Number of homotopy levels (default 1 = disabled). Values > 1 progressively tighten max_weight from homotopy_start_factor to homotopy_end_factor across n levels.
homotopy_start_factor	Starting max_weight multiplier (default 1.0 = no change).
homotopy_end_factor	Ending max_weight multiplier (default 1.0 = no change).
homotopy_budget_p	Budget split fraction across homotopy levels (default 0.5).
scheduler	Margin sweep scheduler: "round_robin" (default) or "greedy" (Greenkhorn priority).
eta_schedule	ALM penalty schedule: "fixed" (default) or "tang_dynamic".
eta_start	Starting ALM penalty multiplier (default 1.0).
eta_end	Ending ALM penalty multiplier (default 1.0).
eta_schedule_power	Power for Tang-eta schedule interpolation (default 0.5).
accelerate	Logical. Enable Safeguarded Regularized Anderson Acceleration (SRAA-m, window m=5) for method="raking", "greenkhorn", "oris", and "oris_soft". Default FALSE. For oris and oris_soft, SRAA-m operates in log-factor space (dimension n_cats_total ~ 50-500, not M_cell), so history matrices are small. Expected benefit: >=30% fewer outer iterations on tight-bounds problems (K>=6, max_weight<3, or skewed margins). Behavioral changes when accelerate=TRUE for oris/oris_soft: • SOR disabled: adaptive under-relaxation (SOR) is turned off; omega stays fixed at omega_init (default 1.0). Use accelerate=FALSE if adaptive SOR is required. • Greedy scheduler downgraded: if scheduler="greedy", it is silently changed to "round_robin" (logged at verbose >= 1). AA requires a deterministic sweep map. • res\$iterations: reports F-evals consumed, not BCD outer iterations (AA-accepted step = 2 F-evals; plain step = 1). • res\$aa_accepted_count: number of AA steps accepted this solve (0 when accelerate=FALSE). Access via attr(result, "result")\$aa_accepted_count. • SRAA history is reset at each homotopy level boundary and on linear-to-log-path fallback.
add_na_proportion	Logical (default FALSE). When TRUE, NA observations in each margin column are encoded as an explicit "NA" bin: existing targets are renormalized by (1

	- <code>na_frac</code>) and an "NA" target equal to <code>na_frac</code> is injected, so that the NA observations receive positive calibration weights rather than being excluded. An error is raised if a margin is entirely NA (<code>na_frac == 1</code>). Known limitation: a column that simultaneously contains real NA values <i>and</i> a literal factor level or character value named "NA" will collide – both will be mapped to the injected NA bin.
<code>auto_collapse</code>	When TRUE, automatically merge rare categories (target proportion < 0.01 or fewer than 30 observations) into an "__other__" bin across all target variables. Default FALSE.
<code>collapse_vars</code>	Optional character vector of target variable names to collapse. When supplied it <i>always</i> takes effect (rare categories in those variables are merged into "__other__"), independent of <code>auto_collapse</code> . NULL (default) collapses nothing unless <code>auto_collapse = TRUE</code> , in which case all target variables are scanned.
<code>target_map</code>	Passed through for data-frame target format handling.
<code>design_weights</code>	Optional design weights vector. When non-NULL and <code>start_weights</code> is NULL, used as starting weights (normalized to mean=1). Length must equal <code>nrow(data)</code> .
<code>newton_tsvd_ratio</code>	Positive finite scalar (default 1e-8); the truncated-SVD singular-value cutoff ratio for the <code>method="newton_kl"</code> Newton solve – singular values below this fraction of the largest are dropped for pseudo-inverse regularization. Ignored by all other methods.
<code>ridge_lambda</code>	Tikhonov ridge penalty on the dual lambda (default 0.0 = off). Adds <code>ridge_lambda</code> to the diagonal of the Hessian (<code>newton_kl</code>) or normal-equations matrix (<code>greg</code>) before factorization. Hardens near-singular systems from cell-level sparseness. Ignored by all other solvers.
...	Additional arguments ignored.

Details

When to use `oris_soft` vs `oris`: Use `method="oris_soft"` when `oris` gives `max_error > 1e-3` and many observations are near `max_weight`. `oris_soft` is roughly 10-30% slower than `oris` but finds a better constrained optimum by temporarily relaxing bounds during optimization, then projecting to exact feasibility at exit.

Greenkhorn: `method="greenkhorn"` implements the Altschuler-Weed-Rigollet (2017) greedy coordinate-descent Sinkhorn variant. At each step greenkhorn selects the single margin with the largest violation and updates only that row/column – unlike standard raking which sweeps all K margins every round. Pass `accelerate=TRUE` to enable SRAA-m outer-loop acceleration on top of greenkhorn.

Logit calibration: `method="logit"` implements Deville-Sarndal (1992) logit-distance Newton calibration. Bounds are enforced analytically through the logit link – no post-hoc weight clamping is required. The logit method typically converges in 10-20 Newton steps and is equivalent to `autumn::calibrate()` with a logit distance function.

Value

data frame with weights column if `attach_weights=TRUE`, else a numeric vector of length n. In both cases the object carries attributes:

result Named list of solver diagnostics. Key fields:

- status: integer status code. 0=converged (RK_OK); 1=did not converge (RK_ERR_NOCONV); 2=infeasible (RK_ERR_INFEAS); 3=bad argument (RK_ERR_BADARG); 4=budget exhausted – loss still decreasing, increase max_iterations (RK_ERR_BUDGET); 5=loss plateau at constrained optimum, weights are valid (RK_ERR_STALL).
- iterations: number of outer iterations completed.
- max_error: maximum marginal error at the returned weights.
- l1_weight_change: L1-normalized weight change sumldw/n . Measured over the last convergence-check interval: $\text{sumldXl}/W_{\text{input}}$ (total input weight) for oris, and sumldw/n for raking and sinkhorn. Computed start-to-final (calibrated minus input, sumldw/n) for logit. Reported as 0 for greg, chebyshev, newton_kl, and greenkhorn (not tracked by those solvers).
- grake_norm: survey-grake normalized residual $\text{max_klmisfit}/(1+|\text{pop}|)$.
- convergence_used: Named list with metric, rule, tol, and fired_at_iter documenting which criterion fired.
- best_error: the convergence-metric value (the convergence_used\$metric family) associated with the returned result. For the acceleration / ORIS finalize path this is RE-COMPUTED on the finalized, bound-clamped, n-normalized weights that are actually returned (y2ks.13) – it reflects the returned solution and is no longer a minimum-across-iterates value, so it need not equal the metric at best_weights. Surfaced verbatim in the budget/stall warning.
- best_weights: numeric vector (length n, sum normalized to n) at the best iterate the solver tracked, finalized through the same $\text{sum}(w)=n + \text{bounds contract}$. It is a DIFFERENT solution from the returned weights and from the best_error reference above. It is all-zero when the solver recorded no best iterate (exited before the first convergence check); guard with `if (sum(attr(r, "result")$best_weights) > 0)` before use.
- convergence_used\$convergence_reason: Character. Why the solver exited: "criterion" (improvement criterion satisfied), "budget" (budget exhausted – increase max_iterations), "stall_kl" (weight KL plateau – at constrained KL minimum), "stall_wchange" (SRAA-m weight-change plateau – at constrained optimum), "infeasible", "error", or "legacy".
- alm_capacity_mu_final: final ALM penalty after adaptive scaling (0 if not oris_soft).
- alm_n_growth_events: adaptive penalty growth fire count.
- alm_max_dual_norm: max absolute Lagrange dual at solver exit.
- alm_sum_drift: $|\text{sum}(\text{weights}) - n|$ after final projection (bounded by $1e-6 * n$).
- sraa_demoted: logical; TRUE iff SRAA-m acceleration was requested with scheduler="greedy" and the greedy scheduler was demoted to round-robin (greedy is incompatible with SRAA's fixed-point geometry). FALSE for non-oris / non-raking solvers and whenever no demotion occurred.

algorithm Character name of the solver used.

iterations Convenience alias for result\$iterations.

Examples

```
## Not run:
df <- data.frame(sex = factor(sample(c("M", "F"), 500, TRUE)))
tgt <- list(sex = c(M = 0.5, F = 0.5))
```

```
# Greenkhorn (greedy coordinate-descent IPF)
r_grk <- harvest(df, tgt, method = "greenkhorn")

# Logit-distance Newton calibration (Deville-Sarndal 1992)
r_logit <- harvest(df, tgt, method = "logit")

## End(Not run)
```

weighted_pct

Weighted proportions

Description

Weighted proportions

Usage

```
weighted_pct(x, weights)
```

Arguments

x	Factor or character vector.
weights	Numeric weights, same length as x.

Value

Named numeric vector of weighted proportions summing to 1.

Index

anesrake, [2](#)

design_effect, [3](#)

diagnose_weights, [4](#)

effective_sample_size, [5](#)

get_current_miss, [5](#)

harvest, [6](#)

weighted_pct, [12](#)