Package 'optweight'

May 9, 2026

Type Package

Title Optimization-Based Stable Balancing Weights

Version 2.0.1

Description Use optimization to estimate weights that balance covariates for binary, multi-category, continuous, and multivariate treatments in the spirit of Zubizarreta (2015) <doi:10.1080/01621459.2015.1023805>. The degree of balance can be specified for each covariate. In addition, sampling weights can be estimated that allow a sample to generalize to a population specified with given target moments of covariates, as in matching-adjusted indirect comparison (MAIC).

Depends R (>= 4.1.0)

Imports osqp (>= 0.6.3.3), chk (>= 0.10.0), rlang (>= 1.1.6), cli (>= 3.6.5), Matrix (>= 1.2-13), collapse (>= 2.1.5), ggplot2 (>= 4.0.0), graphics, stats, utils

Suggests cobalt (>= 4.6.0), scs (>= 3.2.7), clarabel (>= 0.10.1), highs (>= 1.10.0-3), lpSolve (>= 5.6.23), WeightIt, gbm, marginaleffects, sandwich, fwb, knitr, rmarkdown, testthat (>= 3.0.0)

License GPL

Encoding UTF-8

URL https://ngreifer.github.io/optweight/, https://github.com/ngreifer/optweight

BugReports https://github.com/ngreifer/optweight/issues

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Noah Greifer [aut, cre] (ORCID: <https://orcid.org/0000-0003-3067-7154>)

Maintainer Noah Greifer <noah.greifer@gmail.com>

Repository CRAN

Date/Publication 2026-03-19 07:00:10 UTC

Contents

`optweight` *Stable Balancing Weights*

Description

Estimates stable balancing weights for the supplied treatments and covariates. The degree of balance for each covariate is specified by `tols` and the target population can be specified with `targets` or `estimand`. See Zubizarreta (2015) and Wang & Zubizarreta (2020) for details of the properties of the weights and the methods used to fit them.

Usage

```
optweight(
  formula,
  data = NULL,
  tols = 0,
  estimand = "ATE",
  targets = NULL,
  target.tols = 0,
  s.weights = NULL,
  b.weights = NULL,
  focal = NULL,
  norm = "l2",
  min.w = 1e-08,
  verbose = FALSE,
  ...
)

optweight.fit(
  covs,
  treat,
  tols = 0,
  estimand = "ATE",
  targets = NULL,
  target.tols = 0,
  s.weights = NULL,
```

```
  b.weights = NULL,
  focal = NULL,
  norm = "l2",
  std.binary = FALSE,
  std.cont = TRUE,
  min.w = 1e-08,
  verbose = FALSE,
  solver = NULL,
  ...
)
```

Arguments

formula	a formula with a treatment variable on the left hand side and the covariates to be balanced on the right hand side, or a list thereof. Interactions and functions of covariates are allowed.
data	an optional data set in the form of a data frame that contains the variables in formula.
tols	a vector of balance tolerance values for each covariate. The resulting weighted balance statistics will be at least as small as these values. If only one value is supplied, it will be applied to all covariates. Can also be the output of a call to process_tols(). See Details. Default is 0 for all covariates.
estimand	a string containing the desired estimand, which determines the target population. For binary treatments, can be "ATE", "ATT", "ATC", or NULL. For multi-category treatments, can be "ATE", "ATT", or NULL. For continuous treatments, can be "ATE" or NULL. The default for both is "ATE". estimand is ignored when targets is non-NULL. If both estimand and targets are NULL, no targeting will take place. See Details.
targets	an optional vector of target population mean values for each covariate. The resulting weights ensure the midpoint between group means are within target.tols units of the target values for each covariate. If NULL or all NA, estimand will be used to determine targets. Otherwise, estimand is ignored. If any target values are NA, the corresponding variable will not be targeted and its weighted mean will be wherever the weights yield the smallest value of the objective function; this is only allowed for binary and multi-category treatments. Can also be the output of a call to process_targets(). See Details.
target.tols	a vector of target balance tolerance values for each covariate. For binary and multi-category treatments, the average of each pair of means will be at most as far from the target means as these values. Can also be the output of a call to process_tols(). See Details. Default is 0 for all covariates. Ignored with continuous treatments and when estimand is "ATT" or "ATC".
s.weights	a vector of sampling weights. For optweight(), can also be the name of a variable in data that contains sampling weights.
b.weights	a vector of base weights. If supplied, the desired norm of the distance between the estimated weights and the base weights is minimized. For optweight(), can also the name of a variable in data that contains base weights.

`focal` when multi-category treatments are used and `estimand = "ATT"`, which group to consider the "treated" or focal group. This group will not be weighted, and the other groups will be weighted to be more like the focal group. If specified, `estimand` will automatically be set to `"ATT"`.

`norm` `character`; a string containing the name of the norm corresponding to the objective function to minimize. Allowable options include `"l1"` for the L_1 norm, `"l2"` for the L_2 norm (the default), `"linf"` for the L_∞ norm, `"entropy"` for the relative entropy, and `"log"` for the sum of the negative logs. See Details.

`min.w` `numeric`; a single value less than 1 for the smallest allowable weight. Some analyses require nonzero weights for all units, so a small, nonzero minimum may be desirable. The default is `1e-8` (10^{-8}), which does not materially change the properties of the weights from a minimum of 0 but prevents warnings in some packages that use weights in model fitting. When `norm` is `"entropy"` or `"log"` and `min.w <= 0`, `min.w` will be set to the smallest nonzero value.

`verbose` `logical`; whether information on the optimization problem solution should be printed. Default is `FALSE`.

`...` for `optweight()`, additional arguments passed to `optweight.fit()`, including options that are passed to the settings function corresponding to `solver`.

`covs` a numeric matrix of covariates to be balanced.

`treat` a vector of treatment statuses. Non-numeric (i.e., factor or character) vectors are allowed.

`std.binary, std.cont` `logical`; whether the tolerances are in standardized mean units (`TRUE`) or raw units (`FALSE`) for binary variables and continuous variables, respectively. The default is `FALSE` for `std.binary` because raw proportion differences make more sense than standardized mean difference for binary variables. These arguments are analogous to the `binary` and `continuous` arguments in `cobalt::bal.tab()`.

`solver` string; the name of the optimization solver to use. Allowable options depend on `norm`. Default is to use whichever eligible solver is installed, if any, or the default solver for the corresponding `norm`. See Details for information.

Details

`optweight()` is the primary user-facing function for estimating stable balancing weights. The optimization is performed by the lower-level function `optweight.fit()`, which transforms the inputs into the required inputs for the optimization functions and then supplies the outputs (the weights, dual variables, and convergence information) back to `optweight()`. Little processing of inputs is performed by `optweight.fit()`, as this is normally handled by `optweight()`.

For binary and multi-category treatments, weights are estimated so that the weighted mean differences of the covariates are within the given tolerance thresholds controlled by `tols` and `target.tols` (unless `std.binary` or `std.cont` are `TRUE`, in which case standardized mean differences are considered for binary and continuous variables, respectively). For a covariate x with specified balance tolerance δ and target tolerance ε, the weighted means of each each group will be within δ of each other, and the midpoint between the weighted group means will be with ε of the target means. More specifically, the constraints are specified as follows:

$$|\bar{x}_1^w - \bar{x}_0^w| \leq \delta \left| \frac{\bar{x}_1^w + \bar{x}_0^w}{2} - \bar{x}^* \right| \leq \varepsilon$$

where $\bar{x}_1^w$ and $\bar{x}_0^w$ are the weighted means of covariate x for treatment groups 1 and 0, respectively, and $\bar{x}^*$ is the target mean for that covariate. δ corresponds to `tols`, and ε corresponds to `target.tols`. Setting a covariate's value of `target.tols` to `Inf` or its `target` to `NA` both serve to remove the second constraint, as is done in Barnard et al. (2025).

If standardized tolerance values are requested, the standardization factor corresponds to the estimand requested: when the ATE is requested or a target population specified, the standardization factor is the square root of the average variance for that covariate across treatment groups, and when the ATT or ATC are requested, the standardization factor is the standard deviation of the covariate in the focal group. The standardization factor is computed accounting for `s.weights`.

Target and balance constraints are applied to the product of the estimated weights and the sampling weights. In addition, the sum of the product of the estimated weights and the sampling weights is constrained to be equal to the sum of the product of the base weights and sampling weights. For binary and multi-category treatments, these constraints apply within each treatment group.

Continuous treatments:

For continuous treatments, weights are estimated so that the weighted correlation between the treatment and each covariate is within the specified tolerance threshold. The means of the weighted covariates and treatment are restricted to be exactly equal to those of the target population to ensure generalizability to the desired target population, regardless of `tols` or `target.tols`. The weighted correlation is computed as the weighted covariance divided by the product of the *unweighted* standard deviations. The means used to center the variables in computing the covariance are those specified in the target population.

`norm`**:**

The objective function for the optimization problem is $f(\mathbf{w}, \mathbf{b}, \mathbf{s})$, where $\mathbf{w} = \{w_1, \ldots, w_n\}$ are the estimated weights, $\mathbf{s} = \{s_1, \ldots, s_n\}$ are sampling weights (supplied by `s.weights`), and $\mathbf{b} = \{b_1, \ldots, b_n\}$ are base weights (supplied by `b.weights`). The `norm` argument determines $f(.,.,.)$, as detailed below:

- when `norm` = `"l2"`, $f(\mathbf{w}, \mathbf{b}, \mathbf{s}) = \frac{1}{n}\sum_i s_i(w_i - b_i)^2$
- when `norm` = `"l1"`, $f(\mathbf{w}, \mathbf{b}, \mathbf{s}) = \frac{1}{n}\sum_i s_i|w_i - b_i|$
- when `norm` = `"linf"`, $f(\mathbf{w}, \mathbf{b}, \mathbf{s}) = \max_i |w_i - b_i|$
- when `norm` = `"entropy"`, $f(\mathbf{w}, \mathbf{b}, \mathbf{s}) = \frac{1}{n}\sum_i s_i w_i \log \frac{w_i}{b_i}$
- when `norm` = `"log"`, $f(\mathbf{w}, \mathbf{b}, \mathbf{s}) = \frac{1}{n}\sum_i -s_i \log \frac{w_i}{b_i}$

By default, `s.weights` and `b.weights` are set to 1 for all units unless supplied. `b.weights` must be positive when `norm` is `"entropy"` or `"log"`, and `norm` = `"linf"` cannot be used when `s.weights` are supplied.

When `norm` = `"l2"` and both `s.weights` and `b.weights` are `NULL`, weights are estimated to maximize the effective sample size. When `norm` = `"entropy"`, the estimated weights are equivalent to entropy balancing weights (Källberg & Waernbaum, 2023). When `norm` = `"log"`, `b.weights` are ignored in the optimization, as they do not affect the estimated weights.

Dual Variables:

Two types of constraints may be associated with each covariate: target constraints and balance constraints, controlled by `target.tols` and `tols`, respectively. In the `duals` component of the output, each covariate has a dual variable for each constraint placed on it. The dual variable for each constraint is the instantaneous rate of change of the objective function at the optimum corresponding to a change in the constraint. Because this relationship is not linear, large changes

in the constraint will not exactly map onto corresponding changes in the objective function at the optimum, but will be close for small changes in the constraint. For example, for a covariate with a balance constraint of .01 and a corresponding dual variable of 40, increasing (i.e., relaxing) the constraint to .025 will decrease the value of the objective function at the optimum by approximately $(.025 - .01) * 40 = .6$.

For factor variables, `optweight()` takes the sum of the absolute dual variables for the constraints for all levels and reports it as the the single dual variable for the variable itself. This summed dual variable works the same way as dual variables for continuous variables do.

An additional dual variable is computed for the constraint on the range of the weights, controlled by `min.w`. A high dual variable for this constraint implies that decreasing `min.w` will decrease the value of the objective function at the optimum.

`solver`**:**

The `solver` argument controls which optimization solver is used. Different solvers are compatible with each `norm`. See the table below for allowable options, which package they require, which function does the solving, and which function controls the settings.

`solver`	`norm`	Package	Solver function	Settings function
`"osqp"`	`"l2"`, `"l1"`, `"linf"`	**osqp**	`osqp::solve_osqp()`	`osqp::osqpSettings()`
`"highs"`	`"l2"`, `"l1"`, `"linf"`	**highs**	`highs::highs_solve()`	`highs::highs_control()` / `highs::highs_av`
`"lpsolve"`	`"l1"`, `"linf"`	**lpSolve**	`lpSolve::lp()`	.
`"scs"`	`"entropy"`, `"log"`	**scs**	`scs::scs()`	`scs::scs_control()`
`"clarabel"`	`"entropy"`, `"log"`	**clarabel**	`clarabel::clarabel()`	`clarabel::clarabel_control()`

Note that `"lpsolve"` can only be used when `min.w` is nonnegative.

The default `solver` for each `norm` is as follows:

`norm`	Default `solver`
`"l2"`	`"osqp"`
`"l1"`	`"highs"`
`"linf"`	`"highs"`
`"entropy"`	`"scs"`
`"log"`	`"scs"`

If the package corresponding to a default `solver` is not installed but the package for a different eligible solver is, that will be used. Otherwise, you will be asked to install the required package. **osqp** is required for **optweight**, and so will be the default for the `"l1"` and `"linf"` norms if **highs** is not installed. The default package is the one has shown good performance for the given norm in informal testing; generally, all eligible solvers perform about equally well in terms of accuracy but differ in time taken.

Solving Convergence Failure:

Sometimes the optimization will fail to converge at a solution. There are a variety of reasons why this might happen, which include that the constraints are nearly impossible to satisfy or that the optimization surface is relatively flat. It can be hard to know the exact cause or how to solve it, but this section offers some solutions one might try. Typically, solutions can be found most

easily when using the "l2" norm; other norms, especially "linf" and "l1", are more likely to see problems.

Rarely is the problem too few iterations, though this is possible. Most problems can be solved in the default 200,000 iterations, but sometimes it can help to increase this number with the max_iter argument. Usually, though, this just ends up taking more time without a solution found.

If the problem is that the constraints are too tight, it can be helpful to loosen the constraints. Sometimes examining the dual variables of a solution that has failed to converge can reveal which constraints are causing the problem. An extreme value of a dual variable typically suggests that its corresponding constraint is one cause of the failure to converge.

Sometimes a suboptimal solution is possible; such a solution does not satisfy the constraints exactly but will come pretty close. To allow these solutions, the argument eps can be increased to larger values. This is more likely to occur when s.weights are supplied.

Sometimes using a different solver can improve performance. Using the default solver for each norm, as described above, can reduce the probability of convergence failures.

Value

For optweight(), an optweight object with the following elements:

weights	The estimated weights, one for each unit.
treat	The values of the treatment variable.
covs	The covariates used in the fitting. Only includes the raw covariates, which may have been altered in the fitting process.
s.weights	The provided sampling weights.
b.weights	The provided base weights.
estimand	The estimand requested.
focal	The focal variable if the ATT was requested with a multi-category treatment.
call	The function call.
tols	The balance tolerance values for each covariate.
target.tols	The target balance tolerance values for each covariate.
duals	A data.frame containing the dual variables for each covariate. See Details for interpretation of these values.
info	A list containing information about the performance of the optimization at termination.
norm	The norm used.
solver	The solver used.

For optweight.fit(), an optweight.fit object with the following elements:

w	The estimated weights, one for each unit.
duals	A data.frame containing the dual variables for each covariate.
info	A list containing information about the performance of the optimization at termination.
norm	The norm used.
solver	The solver used.

References

Barnard, M., Huling, J. D., & Wolfson, J. (2025). Partially Retargeted Balancing Weights for Causal Effect Estimation Under Positivity Violations (No. arXiv:2510.22072). arXiv. doi:10.48550/arXiv.2510.22072

Chattopadhyay, A., Cohn, E. R., & Zubizarreta, J. R. (2024). One-Step Weighting to Generalize and Transport Treatment Effect Estimates to a Target Population. *The American Statistician*, 78(3), 280–289. doi:10.1080/00031305.2023.2267598

de los Angeles Resa, M., & Zubizarreta, J. R. (2020). Direct and Stable Weight Adjustment in Non-Experimental Studies With Multivalued Treatments: Analysis of the Effect of an Earthquake on Post-Traumatic Stress. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 183(4), 1387–1410. doi:10.1111/rssa.12561

Källberg, D., & Waernbaum, I. (2023). Large Sample Properties of Entropy Balancing Estimators of Average Causal Effects. *Econometrics and Statistics*. doi:10.1016/j.ecosta.2023.11.004

Wang, Y., & Zubizarreta, J. R. (2020). Minimal dispersion approximately balancing weights: Asymptotic properties and practical considerations. *Biometrika*, 107(1), 93–105. doi:10.1093/biomet/asz050

Zubizarreta, J. R. (2015). Stable Weights that Balance Covariates for Estimation With Incomplete Outcome Data. *Journal of the American Statistical Association*, 110(511), 910–922. doi:10.1080/01621459.2015.1023805

See Also

`optweightMV()` for estimating stable balancing weights for multivariate (i.e., multiple) treatments simultaneously.

sbw, which was the inspiration for this package and provides some additional functionality for binary treatments.

WeightIt, which provides a simplified interface to `optweight()` and a more efficient implementation of entropy balancing.

Examples

```
library("cobalt")
data("lalonde", package = "cobalt")

# Balancing covariates between treatment groups (binary)
(ow1 <- optweight(treat ~ age + educ + married +
                    nodegree + re74,
                  data = lalonde,
                  tols = c(.01, .02, .03, .04, .05),
                  estimand = "ATE"))
bal.tab(ow1)

# Exactly balancing covariates with respect to
# race (multi-category)
(ow2 <- optweight(race ~ age + educ + married +
                    nodegree + re74,
                  data = lalonde,
                  tols = 0,
```

```
                  estimand = "ATT",
                  focal = "black"))
bal.tab(ow2)

# Balancing covariates between treatment groups (binary)
# and requesting a specified target population
targets <- process_targets(~ age + educ + married +
                             nodegree + re74,
                           data = lalonde,
                           targets = c(26, 12, .4, .5,
                                       1000))

(ow3a <- optweight(treat ~ age + educ + married +
                     nodegree + re74,
                   data = lalonde,
                   targets = targets,
                   estimand = NULL))

bal.tab(ow3a, disp.means = TRUE)

# Balancing covariates between treatment groups (binary)
# and requesting a specified target population, allowing
# for approximate target balance
(ow3b <- optweight(treat ~ age + educ + married +
                     nodegree + re74,
                   data = lalonde,
                   targets = targets,
                   estimand = NULL,
                   target.tols = .05))

bal.tab(ow3b, disp.means = TRUE)

# Balancing covariates between treatment groups (binary)
# and not requesting a target population
(ow3c <- optweight(treat ~ age + educ + married +
                     nodegree + re74,
                   data = lalonde,
                   targets = NULL,
                   estimand = NULL))

bal.tab(ow3c, disp.means = TRUE)

# Using a different norm
(ow1b <- optweight(treat ~ age + educ + married +
                     nodegree + re74,
                   data = lalonde,
                   tols = c(.01, .02, .03, .04, .05),
                   estimand = "ATE",
                   norm = "l1"))

summary(ow1b, weight.range = FALSE)
summary(ow1, weight.range = FALSE)
```

```
# Allowing for negative weights
ow4 <- optweight(treat ~ age + educ + married + race +
                   nodegree + re74 + re75,
                 data = lalonde,
                 estimand = "ATE",
                 min.w = -Inf)

summary(ow4)

# Using `optweight.fit()`
treat <- lalonde$treat
covs <- splitfactor(lalonde[2:8], drop.first = "if2")

ow.fit <- optweight.fit(covs,
                        treat = treat,
                        tols = .02,
                        estimand = "ATE",
                        norm = "l2")
```

---

optweight.svy *Stable Balancing Weights for Generalization*

Description

Estimates stable balancing weights to generalize a sample characterized by supplied covariates to a given target population. The target means are specified with `targets` and the maximum distance between each weighted covariate mean. See Jackson et al. (2021) for details of the properties of the weights and the methods used to fit them.

Usage

```
optweight.svy(
  formula,
  data = NULL,
  tols = 0,
  targets = NULL,
  s.weights = NULL,
  b.weights = NULL,
  norm = "l2",
  min.w = 1e-08,
  verbose = FALSE,
  ...
)

optweight.svy.fit(
  covs,
  targets,
```

```
  tols = 0,
  s.weights = NULL,
  b.weights = NULL,
  norm = "l2",
  std.binary = FALSE,
  std.cont = TRUE,
  min.w = 1e-08,
  verbose = FALSE,
  solver = NULL,
  ...
)
```

Arguments

formula	a formula with nothing on the left hand side and the covariates to be targeted on the right hand side. Interactions and functions of covariates are allowed. Can be omitted, in which case all variables in data are assumed targeted. If data is NULL and formula is a data.frame, data will be replaced with formula.
data	an optional data set in the form of a data frame that contains the variables in formula.
tols	a vector of target balance tolerance values for each covariate. The resulting weighted covariate means will be no further away from the targets than the specified values. If only one value is supplied, it will be applied to all covariates. Can also be the output of a call to process_tols(). Default is 0 for all covariates.
targets	a vector of target population mean values for each covariate. The resulting weights will yield sample means within tols units of the target values for each covariate. If any target values are NA, the corresponding variable will not be targeted and its weighted mean will be wherever the weights yield the smallest variance. To ensure the weighted mean for a covariate is equal to its unweighted mean (i.e., so that its original mean is its target mean), its original mean must be supplied as a target. For factor variables, a target value must be specified for each level of the factor, and these values must add up to 1. Can also be the output of a call to process_targets().
s.weights	a vector of sampling weights. For optweight(), can also be the name of a variable in data that contains sampling weights.
b.weights	a vector of base weights. If supplied, the desired norm of the distance between the estimated weights and the base weights is minimized. For optweight(), can also the name of a variable in data that contains base weights.
norm	character; a string containing the name of the norm corresponding to the objective function to minimize. Allowable options include "l1" for the L_1 norm, "l2" for the L_2 norm (the default), "linf" for the L_∞ norm, "entropy" for the relative entropy, and "log" for the sum of the negative logs. See Details.
min.w	numeric; a single value less than 1 for the smallest allowable weight. Some analyses require nonzero weights for all units, so a small, nonzero minimum may be desirable. The default is 1e-8 (10^{-8}), which does not materially change the properties of the weights from a minimum of 0 but prevents warnings in

some packages that use weights in model fitting. When `norm` is `"entropy"` or `"log"` and `min.w <= 0`, `min.w` will be set to the smallest nonzero value.

`verbose` `logical`; whether information on the optimization problem solution should be printed. Default is `FALSE`.

`...` for `optweight()`, additional arguments passed to `optweight.fit()`, including options that are passed to the settings function corresponding to `solver`.

`covs` a numeric matrix of covariates to be targeted.

`std.binary, std.cont`
`logical`; whether the tolerances are in standardized mean units (`TRUE`) or raw units (`FALSE`) for binary variables and continuous variables, respectively. The default is `FALSE` for `std.binary` because raw proportion differences make more sense than standardized mean difference for binary variables. These arguments are analogous to the `binary` and `continuous` arguments in `cobalt::bal.tab()`.

`solver` string; the name of the optimization solver to use. Allowable options depend on `norm`. Default is to use whichever eligible solver is installed, if any, or the default solver for the corresponding `norm`. See Details for information.

Details

`optweight.svy()` is the primary user-facing function for estimating stable balancing weights for generalization to a target population. The optimization is performed by the lower-level function `optweight.svy.fit()`, which transforms the inputs into the required inputs for the optimization functions and then supplies the outputs (the weights, dual variables, and convergence information) back to `optweight.svy()`. Little processing of inputs is performed by `optweight.svy.fit()`, as this is normally handled by `optweight.svy()`.

Weights are estimated so that the standardized differences between the weighted covariate means and the corresponding targets are within the given tolerance thresholds (unless `std.binary` or `std.cont` are `FALSE`, in which case unstandardized mean differences are considered for binary and continuous variables, respectively). For a covariate x with specified tolerance δ, the weighted mean will be within δ of the target. If standardized tolerance values are requested, the standardization factor is the standard deviation of the covariate in the whole sample. The standardization factor is always unweighted.

Target constraints are applied to the product of the estimated weights and the sampling weights. In addition, sum of the product of the estimated weights and the sampling weights is constrained to be equal to the sum of the product of the base weights and sampling weights.

See `optweight()` for information on `norm`, `solver`, and convergence failure.

Value

For `optweight.svy()`, an `optweight.svy` object with the following elements:

`weights` The estimated weights, one for each unit.

`covs` The covariates used in the fitting. Only includes the raw covariates, which may have been altered in the fitting process.

`s.weights` The provided sampling weights.

`call` The function call.

`tols`	The tolerance values for each covariate.
`duals`	A data.frame containing the dual variables for each covariate. See `optweight()` for interpretation of these values.
`info`	A list containing information about the performance of the optimization at termination.
`norm`	The `norm` used.
`solver`	The `solver` used.

For `optweight.svy.fit()`, an `optweight.svy.fit` object with the following elements:

`w`	The estimated weights, one for each unit.
`duals`	A data.frame containing the dual variables for each covariate.
`info`	A list containing information about the performance of the optimization at termination.
`norm`	The `norm` used.
`solver`	The `solver` used.

References

Jackson, D., Rhodes, K., & Ouwens, M. (2021). Alternative weighting schemes when performing matching-adjusted indirect comparisons. *Research Synthesis Methods*, 12(3), 333–346. doi:10.1002/jrsm.1466

See Also

`optweight()` for estimating weights that balance treatment groups.

`process_targets()` for specifying the covariate target means supplied to `targets`.

Examples

```
library("cobalt")
data("lalonde", package = "cobalt")

cov.names <- c("age", "educ", "race",
               "married", "nodegree")

targets <- c(age = 23,
             educ = 9,
             race_black = .3,
             race_hispan = .3,
             race_white = .4,
             married = .2,
             nodegree = .5)

ows <- optweight.svy(lalonde[cov.names],
                     targets = targets)
ows

# Unweighted means
```

```
col_w_mean(lalonde[cov.names])

# Weighted means; same as targets
col_w_mean(lalonde[cov.names],
           w = ows$weights)
```

---

optweightMV *Stable Balancing Weights for Multivariate Treatments*

Description

Estimates stable balancing weights for the supplied multivariate (i.e., multiple) treatments and covariates. The degree of balance for each covariate is specified by `tols.list`. See Zubizarreta (2015) and Wang & Zubizarreta (2020) for details of the properties of the weights and the methods used to fit them.

Usage

```
optweightMV(
  formula.list,
  data = NULL,
  tols.list = list(0),
  estimand = "ATE",
  targets = NULL,
  target.tols.list = list(0),
  s.weights = NULL,
  b.weights = NULL,
  norm = "l2",
  min.w = 1e-08,
  verbose = FALSE,
  ...
)

optweightMV.fit(
  covs.list,
  treat.list,
  tols.list = list(0),
  estimand = "ATE",
  targets = NULL,
  target.tols.list = list(0),
  s.weights = NULL,
  b.weights = NULL,
  norm = "l2",
  std.binary = FALSE,
  std.cont = TRUE,
  min.w = 1e-08,
```

```
  verbose = FALSE,
  solver = NULL,
  ...
)
```

Arguments

formula.list	a list of formulas, each with a treatment variable on the left hand side and the covariates to be balanced on the right hand side.
data	an optional data set in the form of a data frame that contains the variables in formula.list.
tols.list	a list of vectors of balance tolerance values for each covariate for each treatment. The resulting weighted balance statistics will be at least as small as these values. If only one value is supplied, it will be applied to all covariates. See Details. Default is 0 for all covariates.
estimand	the desired estimand, which determines the target population. Only "ATE" or NULL are supported. estimand is ignored when targets is non-NULL. If both estimand and targets are NULL, no targeting will take place.
targets	an optional vector of target population mean values for each covariate. The resulting weights ensure the midpoint between group means are within target.tols units of the target values for each covariate. If NULL or all NA, estimand will be used to determine targets. Otherwise, estimand is ignored. If any target values are NA, the corresponding variable will not be targeted and its weighted mean will be wherever the weights yield the smallest value of the objective function; this is only allowed if all treatments are binary or multi-category. Can also be the output of a call to process_targets(). See Details.
target.tols.list	a list of vectors of target balance tolerance values for each covariate for each treatment. For binary and multi-category treatments, the average of each pair of means will be at most as far from the target means as these values. Can also be the output of a call to process_tols(). See Details. Default is 0 for all covariates. Ignored with continuous treatments.
s.weights	a vector of sampling weights. For optweightMV(), can also be the name of a variable in data that contains sampling weights.
b.weights	a vector of base weights. If supplied, the desired norm of the distance between the estimated weights and the base weights is minimized. For optweightMV(), can also the name of a variable in data that contains base weights.
norm	character; a string containing the name of the norm corresponding to the objective function to minimize. Allowable options include "l1" for the L_1 norm, "l2" for the L_2 norm (the default), "linf" for the L_∞ norm, "entropy" for the relative entropy, and "log" for the sum of the negative logs. See Details.
min.w	numeric; a single value less than 1 for the smallest allowable weight. Some analyses require nonzero weights for all units, so a small, nonzero minimum may be desirable. The default is 1e-8 (10^{-8}), which does not materially change the properties of the weights from a minimum of 0 but prevents warnings in some packages that use weights in model fitting. When norm is "entropy" or "log" and min.w <= 0, min.w will be set to the smallest nonzero value.

verbose	logical; whether information on the optimization problem solution should be printed. Default is FALSE.
...	for optweightMV(), additional arguments passed to optweightMV.fit(), including options that are passed to the settings function corresponding to solver.
covs.list	a list containing one numeric matrix of covariates to be balanced for each treatment.
treat.list	a list containing one vector of treatment statuses for each treatment.
std.binary, std.cont	logical; whether the tolerances are in standardized mean units (TRUE) or raw units (FALSE) for binary variables and continuous variables, respectively. The default is FALSE for std.binary because raw proportion differences make more sense than standardized mean difference for binary variables. These arguments are analogous to the binary and continuous arguments in cobalt::bal.tab().
solver	string; the name of the optimization solver to use. Allowable options depend on norm. Default is to use whichever eligible solver is installed, if any, or the default solver for the corresponding norm. See Details for information.

Details

optweightMV() is the primary user-facing function for estimating stable balancing weights for multivariate treatments. The optimization is performed by the lower-level function optweightMV.fit(), which transforms the inputs into the required inputs for the optimization functions and then supplies the outputs (the weights, dual variables, and convergence information) back to optweightMV(). Little processing of inputs is performed by optweightMV.fit(), as this is normally handled by optweightMV().

See optweight() for more information about balance tolerances (i.e., those specified in tols.list), targets, norm, solver, and convergence failure.

Value

For optweightMV(), an optweightMV object with the following elements:

weights	The estimated weights, one for each unit.
treat.list	A list of the values of the treatment variables.
covs.list	A list of the covariates for each treatment used in the fitting. Only includes the raw covariates, which may have been altered in the fitting process.
s.weights	The provided sampling weights.
b.weights	The provided base weights.
call	The function call.
tols	A list of tolerance values for each covariate for each treatment.
duals	A list of data.frames containing the dual variables for each covariate for each treatment. See optweight() for interpretation of these values.
info	A list containing information about the performance of the optimization at termination.
norm	The norm used.

`solver`	The `solver` used.

For `optweightMV.fit()`, an `optweightMV.fit` object with the following elements:

`w`	The estimated weights, one for each unit.
`duals`	A data.frame containing the dual variables for each covariate.
`info`	A list containing information about the performance of the optimization at termination.
`norm`	The `norm` used.
`solver`	The `solver` used.

References

Chattopadhyay, A., Cohn, E. R., & Zubizarreta, J. R. (2024). One-Step Weighting to Generalize and Transport Treatment Effect Estimates to a Target Population. *The American Statistician*, 78(3), 280–289. doi:10.1080/00031305.2023.2267598

Källberg, D., & Waernbaum, I. (2023). Large Sample Properties of Entropy Balancing Estimators of Average Causal Effects. *Econometrics and Statistics*. doi:10.1016/j.ecosta.2023.11.004

Wang, Y., & Zubizarreta, J. R. (2020). Minimal dispersion approximately balancing weights: Asymptotic properties and practical considerations. *Biometrika*, 107(1), 93–105. doi:10.1093/biomet/asz050

Zubizarreta, J. R. (2015). Stable Weights that Balance Covariates for Estimation With Incomplete Outcome Data. *Journal of the American Statistical Association*, 110(511), 910–922. doi:10.1080/01621459.2015.1023805

See Also

`optweight()` for more information on the optimization, specifications, and options.

Examples

```
library("cobalt")
data("lalonde", package = "cobalt")

# Balancing two treatments
(ow1 <- optweightMV(list(treat ~ age + educ + race + re74,
                         re75 ~ age + educ + race + re74),
                    data = lalonde))

summary(ow1)

bal.tab(ow1)
```

plot.optweight *Plot Dual Variables for Covariate Constraints*

Description

Plots the dual variables resulting from optweight(), optweightMV(), or optweight.svy() in a way similar to figure 2 of Zubizarreta (2015), which explains how to interpret these values.

Usage

```
## S3 method for class 'optweight'
plot(x, type = "variables", ...)

## S3 method for class 'optweightMV'
plot(x, which.treat = 1L, type = "variables", ...)

## S3 method for class 'optweight.svy'
plot(x, type = "variables", ...)
```

Arguments

x	an optweight, optweightMV, or optweight.svy object; the output of a call to optweight(), optweightMV(), or optweight.svy().
type	the type of plot to display; allowable options include "variables" (the default), which produces a row for each covariate, and "constraints", which produces a row for each type of constraint (computed as the sum of the absolute dual variables for each constraint type).
...	ignored.
which.treat	for optweightMV objects, an integer corresponding to which treatment to display. Only one may be displayed at a time.

Details

Dual variables represent the cost of changing the constraint on the objective function minimized to estimate the weights. For covariates with large values of the dual variable, tightening the constraint will increase the variability of the weights, and relaxing the constraint will decrease the variability of the weights, both to a greater extent than would doing the same for covariate with small values of the dual variable. See optweight() and vignette("optweight") for more information on interpreting dual variables.

Value

A ggplot object that can be used with other **ggplot2** functions.

References

Zubizarreta, J. R. (2015). Stable Weights that Balance Covariates for Estimation With Incomplete Outcome Data. *Journal of the American Statistical Association*, 110(511), 910–922. doi:10.1080/01621459.2015.1023805

See Also

`optweight()`, `optweightMV()`, or `optweight.svy()` to estimate the weights and the dual variables.

`plot.summary.optweight()` for plots of the distribution of weights.

Examples

```
library("cobalt")
data("lalonde", package = "cobalt")

tols <- process_tols(treat ~ age + educ + married +
                       nodegree + re74, data = lalonde,
                     tols = .1)

#Balancing covariates between treatment groups (binary)
ow1 <- optweight(treat ~ age + educ + married +
                   nodegree + re74, data = lalonde,
                 tols = tols,
                 estimand = "ATT")

# Note the L2 divergence and effective sample
# size (ESS)
summary(ow1, weight.range = FALSE)

# age has a low value, married is high
plot(ow1)

tols["age"] <- 0
ow2 <- optweight(treat ~ age + educ + married +
                   nodegree + re74, data = lalonde,
                 tols = tols,
                 estimand = "ATT")

# Notice that tightening the constraint on age has
# a negligible effect on the variability of the
# weights and ESS
summary(ow2, weight.range = FALSE)

tols["age"] <- .1
tols["married"] <- 0
ow3 <- optweight(treat ~ age + educ + married +
                   nodegree + re74, data = lalonde,
                 tols = tols,
                 estimand = "ATT")
```

```
# In contrast, tightening the constraint on married
# has a large effect on the variability of the
# weights, shrinking the ESS
summary(ow3, weight.range = FALSE)

# More duals are displayed when targeting other
# estimands:
ow4 <- optweight(treat ~ age + educ + married +
                   nodegree + re74, data = lalonde,
                 estimand = "ATE")

plot(ow4)

# Display duals by constraint type
plot(ow4, type = "constraints")
```

---

process_targets *Construct and Check Targets Input*

Description

Checks whether proposed target population means values for `targets` are suitable in number and order for submission to `optweight()`, `optweightMV()`, and `optweight.svy()`, and returns an object that can supplied to the `targets` argument of these functions.

Usage

```
process_targets(formula, data = NULL, targets = NULL, s.weights = NULL)

check.targets(...)

## S3 method for class 'optweight.targets'
print(x, digits = 5, ...)
```

Arguments

`formula`	a formula with nothing on the left hand side and the covariates to be targeted on the right hand side. Interactions and functions of covariates are allowed. Can be omitted, in which case all variables in `data` are assumed targeted. If `data` is `NULL` and `formula` is a data.frame, `data` will be replaced with `formula`.
`data`	an optional data set in the form of a data frame that contains the variables in `formula`.
`targets`	a vector of target population mean values for each covariate. These should be in the order corresponding to the order of the corresponding variable in `formula`, except for interactions, which will appear after all lower-order terms. For factor variables, a target value must be specified for each level of the factor, and these values must add up to 1. If `NULL`, the current sample means will be produced

	(weighted by `s.weights`). If `NA`, an `NA` vector named with the covariate names will be produced.
`s.weights`	a vector of sampling weights. For `optweight()`, can also be the name of a variable in `data` that contains sampling weights.
`...`	for `optweight()`, additional arguments passed to `optweight.fit()`, including options that are passed to the settings function corresponding to `solver`.
`x`	an `optweight.targets` object; the output of a call to `process_targets()`.
`digits`	how many digits to print.

Details

The purpose of `process_targets()` is to allow users to ensure that their proposed input to `targets` in `optweight()`, `optweightMV()`, and `optweight.svy()` is correct both in the number of entries and their order. This is especially important when factor variables and interactions are included in the formula because factor variables are split into several dummies and interactions are moved to the end of the variable list, both of which can cause some confusion and potential error when entering `targets` values.

Factor variables are internally split into a dummy variable for each level, so the user must specify a target population mean value for each level of the factor. These must add up to 1, and an error will be displayed if they do not. These values represent the proportion of units in the target population with each factor level.

Interactions (e.g., `a:b` or `a*b` in the `formula` input) are always sent to the end of the variable list even if they are specified elsewhere in the `formula`. It is important to run `process_targets()` to ensure the order of the proposed `targets` corresponds to the represented order of covariates used in the formula. You can run `process_targets(., targets = NA)` to see the order of covariates that is required without specifying any targets.

Value

An `optweight.targets` object, which is a named vector of target population mean values, one for each (expanded) covariate specified in `formula`. This should be used as an input to the `targets` argument of `optweight()`, `optweightMV()`, and `optweight.svy()`.

See Also

`process_tols()`

Examples

```
library("cobalt")
data("lalonde", package = "cobalt")

# Generating targets; means by default
targets <- process_targets(~ age + race + married +
                             nodegree + re74,
                           data = lalonde)

# Notice race is split into three values
```

```
targets

# Generating targets; NA by default
targets <- process_targets(~ age + race + married +
                             nodegree + re74,
                           data = lalonde,
                           targets = NA)
targets

# Can also supply just a dataset
covs <- lalonde |>
  subset(select = c(age, race, married,
                    nodegree, re74))

targets <- process_targets(covs)

targets
```

---

process_tols *Construct and Check Tolerance Input*

Description

Checks whether proposed tolerance values for `tols` are suitable in number and order for submission to `optweight()` and `optweight.svy()`, and returns an object that can supplied to the `tols` argument of these functions.

Usage

```
process_tols(formula, data = NULL, tols = 0)

check.tols(...)

## S3 method for class 'optweight.tols'
print(x, internal = FALSE, digits = 5, ...)
```

Arguments

`formula`	a formula with the covariates to be balanced on the right-hand side. Interactions and functions of covariates are allowed. Lists of formulas are not allowed; multiple formulas must be checked one at a time.
`data`	an optional data set in the form of a data frame that contains the variables in `formula`.
`tols`	a vector of balance tolerance values in standardized mean difference units for each covariate. These should be in the order corresponding to the order of the corresponding variable in `formula`, except for interactions, which will appear after all lower-order terms. If only one value is supplied, it will be applied to all covariates.

...	ignored.
x	an optweight.tols object; the output of a call to process_tols().
internal	logical; whether to print the tolerance values that are to be used internally by optweight(). See Value section.
digits	how many digits to print.

Details

The purpose of process_tols() is to allow users to ensure that their proposed input to tols in optweight() is correct both in the number of entries and their order. This is especially important when factor variables and interactions are included in the formula because factor variables are split into several dummies and interactions are moved to the end of the variable list, both of which can cause some confusion and potential error when entering tols values.

Factor variables are internally split into a dummy variable for each level, but the user only needs to specify one tolerance value per original variable; process_tols() automatically expands the tols input to match the newly created variables.

Interactions (e.g., a:b or a*b in the formula input) are always sent to the end of the variable list even if they are specified elsewhere in the formula. It is important to run process_tols() to ensure the order of the proposed tols corresponds to the represented order of covariates used in optweight(). You can run process_tols() with no tols input to see the order of covariates that is required.

Note that only one formula and vector of tolerance values can be assessed at a time; for multiple treatments, each formula and tolerance vector must be entered separately.

Value

An optweight.tols object, which is a named vector of tolerance values, one for each variable specified in formula. This should be used as an input to the tols argument of optweight(). The "internal.tols" attribute contains the tolerance values to be used internally by optweight(). These will differ from the vector values when there are factor variables that are split up; the user only needs to submit one tolerance per factor variable, but separate tolerance values are produced for each new dummy created.

See Also

process_targets()

Examples

```
library("cobalt")
data("lalonde", package = "cobalt")

# Generating tols; 0 by default
tols <- process_tols(treat ~ age + educ + married +
                       nodegree + re74,
                     data = lalonde)

tols
```

```
tols <- process_tols(treat ~ age + educ + married +
                       nodegree + re74,
                     data = lalonde,
                     tols = .05)

tols

# Checking the order of interactions; notice they go
# at the end even if specified at the beginning.
tols <- process_tols(treat ~ age:educ + married*race +
                       nodegree + re74,
                     data = lalonde,
                     tols = .05)

tols

# Internal tolerances for expanded covariates
print(tols, internal = TRUE)
```

---

summary.optweight *Summarize, Print, and Plot Information about Estimated Weights*

Description

These functions summarize the weights resulting from a call to optweight(), optweightMV(), or optweight.svy(). summary() produces summary statistics on the distribution of weights, including their range and variability, and the effective sample size of the weighted sample (computed using the formula in McCaffrey, et al., 2004). plot() creates a histogram of the weights.

Usage

```
## S3 method for class 'optweight'
summary(object, top = 5L, ignore.s.weights = FALSE, weight.range = TRUE, ...)

## S3 method for class 'optweightMV'
summary(object, top = 5L, ignore.s.weights = FALSE, weight.range = TRUE, ...)

## S3 method for class 'optweight.svy'
summary(object, top = 5L, ignore.s.weights = FALSE, weight.range = TRUE, ...)

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

Arguments

object	an optweight, optweightMV, or optweight.svy object; the output of a call to optweight(), optweightMV(), or optweight.svy().

top — integer; how many of the largest and smallest weights to display. Default is 5. Ignored when weight.range = FALSE.

ignore.s.weights — logical; whether to ignore sampling weights when computing the weight summary. Default is

weight.range — logical; whether to display statistics about the range of weights and the highest and lowest weights for each group. Default is TRUE.

... — Additional arguments. For plot(), additional arguments passed to graphics::hist() to determine the number of bins, though ggplot2::geom_histogram() from **ggplot2** is actually used to create the plot.

x — a summary.optweight, summary.optweightMV, or summary.optweight.svy object; the output of a call to summary.optweight(), summary.optweightMV(), or ()summary.optweight.svy.

Value

For point treatments (i.e., optweight objects), summary() returns a summary.optweight object with the following elements:

weight.range — The range (minimum and maximum) weight for each treatment group.

weight.top — The units with the greatest weights in each treatment group; how many are included is determined by top.

l2 — The square root of the L_2 norm of the estimated weights from the base weights, weighted by the sampling weights (if any): $\sqrt{\frac{1}{n}\sum_i s_i(w_i - b_i)^2}$

l1 — The L_1 norm of the estimated weights from the base weights, weighted by the sampling weights (if any): $\frac{1}{n}\sum_i s_i|w_i - b_i|$

linf — The L_∞ norm (maximum absolute deviation) of the estimated weights from the base weights: $\max_i |w_i - b_i|$

rel.ent — The relative entropy between the estimated weights and the base weights, weighted by the sampling weights (if any): $\frac{1}{n}\sum_i s_i w_i \log\left(\frac{w_i}{b_i}\right)$. Only computed if all weights are positive.

num.zeros — The number of units with a weight equal to 0.

effective.sample.size — The effective sample size for each treatment group before and after weighting.

For multivariate treatments (i.e., optweightMV objects), a list of the above elements for each treatment.

For optweight.svy objects, the above object but with no treatment group divisions.

plot() returns a ggplot object with a histogram displaying the distribution of the estimated weights. If the estimand is the ATT or ATC, only the weights for the non-focal group(s) will be displayed (since the weights for the focal group are all 1). A dotted line is displayed at the mean of the weights (the mean of the base weights, or 1 if not supplied).

References

McCaffrey, D. F., Ridgeway, G., & Morral, A. R. (2004). Propensity Score Estimation With Boosted Regression for Evaluating Causal Effects in Observational Studies. *Psychological Methods*, 9(4), 403–425. doi:10.1037/1082989X.9.4.403

See Also

plot.optweight() for plotting the values of the dual variables.

Examples

```
library("cobalt")
data("lalonde", package = "cobalt")

#Balancing covariates between treatment groups (binary)
(ow1 <- optweight(treat ~ age + educ + married +
                    nodegree + re74, data = lalonde,
                  tols = .001,
                  estimand = "ATT"))

(s <- summary(ow1))

plot(s, breaks = 12)
```

Index