

Package ‘optweight’

September 9, 2025

Type Package

Title Optimization-Based Stable Balancing Weights

Version 1.0.0

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](https://doi.org/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.

Depends R (>= 4.1.0)

Imports osqp (>= 0.6.3.3), chk (>= 0.10.0), rlang (>= 1.1.6), Matrix (>= 1.2-13), ggplot2 (>= 3.5.0), 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 7.3.2

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 2025-09-09 06:50:11 UTC

Contents

optweight	2
optweight.fit	8
optweight.svy	12
optweight.svy.fit	15
plot.optweight	18
process_targets	20
process_tols	22
summary.optweight	24

Index	27
--------------	-----------

optweight	<i>Estimate Stable Balancing Weights</i>
-----------	------------------------------------------

Description

Estimate stable balancing weights for treatments and covariates specified in formula. 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 (2019) 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,
  s.weights = NULL,
  b.weights = NULL,
  focal = NULL,
  norm = "l2",
  verbose = FALSE,
  ...
)

optweightMV(
  formula.list,
  data = NULL,
  tols.list = list(0),
  estimand = "ATE",
  targets = NULL,
  s.weights = NULL,
  b.weights = NULL,
  focal = NULL,
```

```

    norm = "l2",
    verbose = FALSE,
    ...
)

```

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. See glm() for more details. 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, or a list thereof. 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.
estimand	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". For optweightMV() , only "ATE" or NULL are supported. <code>estimand</code> is ignored when <code>targets</code> is non-NULL. If both <code>estimand</code> and <code>targets</code> are NULL, no targeting will take place. See Details.
targets	A vector of target population mean values for each baseline covariate. The resulting weights will yield sample means within <code>tols/2</code> units of the target values for each covariate. If NULL or all NA, <code>estimand</code> will be used to determine targets. Otherwise, <code>estimand</code> 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 variance. Can also be the output of a call to process_targets() . See Details.
s.weights	A vector of sampling weights or the name of a variable in data that contains sampling weights.
b.weights	A vector of base weights or the name of a variable in data that contains base weights. If supplied, the desired norm of the distance between the estimated weights and the base weights is minimized.
focal	When multi-category treatments are used and <code>estimand = "ATT"</code> , 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, <code>estimand</code> 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 L1 norm, "l2" for the L2 norm (the default), "linf" for the L_∞ norm, "entropy" for the negative entropy, and "log" for the sum of the logs. See optweight.fit() for details.
verbose	logical; whether information on the optimization problem solution should be printed. Default is FALSE.
...	Arguments passed on to optweight.fit , optweightMV.fit

	<p><code>std.binary, std.cont</code> 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 <code>std.binary</code> because raw proportion differences make more sense than standardized mean difference for binary variables. These arguments are analogous to the <code>binary</code> and <code>continuous</code> arguments in <code>bal.tab()</code> in cobalt.</p> <p><code>min.w</code> 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 to estimate treatment effects. When <code>norm</code> is "entropy" or "log" and <code>min.w</code> ≤ 0, <code>min.w</code> will be set to the smallest nonzero value.</p> <p><code>covs.list</code> a list containing one numeric matrix of covariates to be balanced for each treatment.</p> <p><code>treat.list</code> a list containing one vector of treatment statuses for each treatment.</p> <p><code>solver</code> string; the name of the optimization solver to use. Allowable options depend on <code>norm</code>. Default is to use whichever eligible solver is installed, if any, or the default solver for the corresponding <code>norm</code>. See Details for information.</p>
<code>formula.list</code>	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.
<code>tols.list</code>	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.

Details

The optimization is performed by the lower-level function `optweight.fit()` (for `optweight()`) or `optweightMV.fit()` (for `optweightMV()`).

For binary and multi-category treatments, weights are estimated so that the weighted mean differences of the covariates are within the given tolerance thresholds (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 tolerance δ , the weighted means of each group will be within δ of each other. Additionally, when the ATE is specified as the estimand or a target population is specified, the weighted means of each group will each be within $\delta/2$ of the target means; this ensures generalizability to the same population from which the original sample was drawn.

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`.

For continuous treatments, weights are estimated so that the weighted correlation between the treatment and each covariate is within the specified tolerance threshold. If the ATE is requested or a

target population is specified, the means of the weighted covariates and treatment are restricted to be equal to those of the target population to ensure generalizability to the desired target population. 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.

Dual Variables:

Two types of constraints may be associated with each covariate: target constraints and balance constraints. Target constraints require the mean of the covariate to be at (or near) a specific target value in each treatment group (or for the whole group when treatment is continuous). Balance constraints require the means of the covariate in pairs of treatments to be near each other. For binary and multi-category treatments, balance constraints are redundant if target constraints are provided for a variable. For continuous variables, balance constraints refer to the correlation between treatment and the covariate and are not redundant with target constraints. In the `duals` component of the output, each covariate has a dual variable for each nonredundant 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 single dual variable for the variable itself. This summed dual variable works the same way as dual variables for continuous variables do.

Value

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

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

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

<code>weights</code>	The estimated weights, one for each unit.
<code>treat.list</code>	A list of the values of the treatment variables.
<code>covs.list</code>	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.
<code>s.weights</code>	The provided sampling weights.
<code>b.weights</code>	The provided base weights.
<code>call</code>	The function call.
<code>tols</code>	A list of tolerance values for each covariate for each treatment.
<code>duals</code>	A list of data.frames containing the dual variables for each covariate for each treatment. See Details for interpretation of these values.
<code>info</code>	Information about the performance of the optimization at termination.

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](https://doi.org/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](https://doi.org/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](https://doi.org/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](https://doi.org/10.1080/01621459.2015.1023805)

See Also

[optweight.fit\(\)](#), the lower-level function that performs the fitting. Links on that page can help with diagnosing and fixing more subtle issues with the optimization.

[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("lalonge", package = "cobalt")

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

bal.tab(ow1)
```

```

# Exactly alancing 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 not requesting a target population
(ow3b <- optweight(treat ~ age + educ + married +
  nodegree + re74, data = lalonde,
  targets = NULL,
  estimand = NULL))

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

# Balancing two treatments
(ow4 <- optweightMV(list(treat ~ age + educ + race + re74,
  re75 ~ age + educ + race + re74),
  data = lalonde))

summary(ow4)

bal.tab(ow4)

# 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
ow5 <- optweight(treat ~ age + educ + married + race +

```

```

        nodegree + re74 + re75,
data = lalonde,
estimand = "ATE",
min.w = -Inf)

summary(ow5)

```

optweight.fit

Fitting Function for Stable Balancing Weights

Description

optweight.fit() and optweightMV.fit() perform the optimization for [optweight\(\)](#) and [optweightMV\(\)](#) and should, in most cases, not be used directly. Little processing of inputs is performed, so they must be given exactly as described below.

Usage

```

optweight.fit(
  covs,
  treat,
  tols = 0,
  estimand = "ATE",
  targets = NULL,
  s.weights = NULL,
  b.weights = NULL,
  focal = NULL,
  norm = "l2",
  std.binary = FALSE,
  std.cont = TRUE,
  min.w = 1e-08,
  verbose = FALSE,
  solver = NULL,
  ...
)

```

```

optweightMV.fit(
  covs.list,
  treat.list,
  tols.list = list(0),
  estimand = "ATE",
  targets = NULL,
  s.weights = NULL,
  b.weights = NULL,
  norm = "l2",
  std.binary = FALSE,

```



```

    std.cont = TRUE,
    min.w = 1e-08,
    verbose = FALSE,
    solver = NULL,
    ...
)

```

Arguments

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.
tols	a vector of balance tolerance values for each covariate. Default is 0.
estimand	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". For <code>optweightMV.fit()</code> , only "ATE" or NULL are supported. <code>estimand</code> is ignored when <code>targets</code> is non-NULL. If both <code>estimand</code> and <code>targets</code> are NULL, no targeting will take place.
targets	an optional vector of target population mean values for each baseline covariate. The resulting weights will yield sample means within <code>tols/2</code> units of the target values for each covariate. If NULL or all NA, <code>estimand</code> will be used to determine targets. Otherwise, <code>estimand</code> 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 variance.
s.weights	an optional vector of sampling weights. Default is a vector of 1s.
b.weights	an optional vector of base weights. Default is a vector of 1s.
focal	when multi-categorical treatments are used and the <code>estimand = "ATT"</code> , which group to consider the "treated" or focal group. This group will not be weighted, and the other groups will be weighted to resemble the focal group.
norm	character; a string containing the name of the norm corresponding to the objective function to minimize. Allowable options include "l1" for the L1 norm, "l2" for the L2 norm (the default), "linf" for the L_∞ norm, "entropy" for the negative entropy, and "log" for the sum of the negative logs. See Details.
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 <code>std.binary</code> because raw proportion differences make more sense than standardized mean difference for binary variables. These arguments are analogous to the <code>binary</code> and <code>continuous</code> arguments in <code>bal.tab()</code> in cobalt .
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 to estimate treatment effects. When <code>norm</code> is "entropy" or "log" and <code>min.w <= 0</code> , <code>min.w</code> will be set to the smallest nonzero value.

verbose	logical; whether information on the optimization problem solution should be printed. Default is FALSE.
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.
...	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.
tols.list	a list of balance tolerance vectors, one for each treatment, each with a value for each covariate.

Details

optweight.fit() and optweightMV.fit() transform the inputs into the required inputs for the optimization functions, which are (sparse) matrices and vectors, and then supplies the outputs (the weights, dual variables, and convergence information) back to optweight() or optweightMV(). Little processing of inputs is performed, as this is normally handled by optweight() or optweightMV().

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.

norm:

The objective function for the optimization problem is $f(w_i, b_i, s_i)$, where w_i is the estimated weight for unit i , s_i is the sampling weight for unit i (supplied by s.weights) and b_i is the base weight for unit i (supplied by b.weights). The norm argument determines $f(., ., .)$, as detailed below:

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

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.

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; 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.

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.

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

An `optweight.fit` or `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 (for <code>optweight.fit()</code>), or a list thereof (for <code>optweightMV.fit()</code>). See <code>vignette("optweight")</code> for interpretation of these values.
info	A list containing information about the performance of the optimization at termination.

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()` and `optweightMV()` which you should use for estimating the balancing weights, unless you know better.

Examples

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

treat <- lalonge$treat
covs <- splitfactor(lalonge[2:8], drop.first = "if2")

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

Description

Estimate targeting weights for covariates specified in `formula`. The target means are specified with `targets` and the maximum distance between each weighted covariate mean and the corresponding target mean is specified by `tols`. See Zubizarreta (2015) 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",
  verbose = FALSE,
  ...
)
```

Arguments

<code>formula</code>	a formula with nothing on the left hand side and the covariates to be targeted on the right hand side. See <code>glm()</code> for more details. Interactions and functions of covariates are allowed.
<code>data</code>	An optional data set in the form of a data frame that contains the variables in <code>formula</code> .
<code>tols</code>	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 <code>process_tols()</code> .
<code>targets</code>	A vector of target population mean values for each baseline covariate. The resulting weights will yield sample means within <code>tols/2</code> units of the target values for each covariate. If <code>NULL</code> or all <code>NA</code> , <code>estimand</code> will be used to determine targets. Otherwise, <code>estimand</code> is ignored. If any target values are <code>NA</code> , the corresponding variable will not be targeted and its weighted mean will be wherever the weights yield the smallest variance. Can also be the output of a call to <code>process_targets()</code> . See Details.
<code>s.weights</code>	A vector of sampling weights or the name of a variable in <code>data</code> that contains sampling weights.
<code>b.weights</code>	A vector of base weights or the name of a variable in <code>data</code> that contains base weights. If supplied, the desired norm of the distance between the estimated weights and the base weights is minimized.
<code>norm</code>	character; a string containing the name of the norm corresponding to the objective function to minimize. Allowable options include "l1" for the L1 norm, "l2" for the L2 norm (the default), "l1nf" for the L_∞ norm, "entropy" for

	the negative entropy, and "log" for the sum of the logs. See optweight.fit() for details.
verbose	logical; whether information on the optimization problem solution should be printed. Default is FALSE.
...	Arguments passed on to optweight.svy.fit
	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 <code>bal.tab()</code> in cobalt .
	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 to estimate treatment effects. When norm is "entropy" or "log" and <code>min.w <= 0</code> , min.w will be set to the smallest nonzero value.

Details

The optimization is performed by the lower-level function [optweight.svy.fit\(\)](#).

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.

See the [optweight\(\)](#) help page for information on interpreting dual variables and solving convergence failure.

Value

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	Information about the performance of the optimization at termination.

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](https://doi.org/10.1080/01621459.2015.1023805)

See Also

[optweight.svy.fit\(\)](#), the lower-level function that performs the fitting.

[optweight.fit\(\)](#) for more details about the optimization options.

[optweight\(\)](#) for estimating weights that balance treatment groups.

Examples

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

cov.formula <- ~ age + educ + race + married + nodegree

targets <- process_targets(cov.formula, data = lalonge,
                           targets = c(23, 9, .3, .3, .4,
                                       .2, .5))

ows <- optweight.svy(cov.formula,
                     data = lalonge,
                     tols = 0,
                     targets = targets)

ows

#Unweighted means
col_w_mean(ows$covs)

#Weighted means; same as targets
col_w_mean(ows$covs, w = ows$weights)
```

optweight.svy.fit

Fitting Function for Optweight for Survey Weights

Description

`optweight.svy.fit()` performs the optimization for [optweight.svy\(\)](#) and should, in most cases, not be used directly. Little processing of inputs is performed, so they must be given exactly as described below.

Usage

```
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

covs	a numeric matrix of covariates to be targeted.
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.
tol	a vector of target balance tolerance values. Default is 0.
s.weights	an optional vector of sampling weights. Default is a vector of 1s.
b.weights	an optional vector of base weights. Default is a vector of 1s.
norm	character; a string containing the name of the norm corresponding to the objective function to minimize. Allowable options include "l1" for the L1 norm, "l2" for the L2 norm (the default), "linf" for the L_∞ norm, "entropy" for the negative entropy, and "log" for the sum of the negative logs. See Details at optweight.fit() for more information.
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 <code>bal.tab()</code> in cobalt .
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 to estimate treatment effects. When norm is "entropy" or "log" and <code>min.w <= 0</code> , 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.
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 at <code>optweight.fit()</code> for information.
...	Options that are passed to the settings function corresponding to solver.

Details

`optweight.svy.fit()` transforms the inputs into the required inputs for the optimization functions, which are (sparse) matrices and vectors, and then supplies the outputs (the weights, dual variables, and convergence information) back to `optweight.svy()`. Little processing of inputs is performed, as this is normally handled by `optweight.svy()`.

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.

Value

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. See Zubizarreta (2015) for interpretation of these values.
info	A list containing information about the performance of the optimization at termination.

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.svy()` which you should use for estimating the balancing weights, unless you know better.

`optweight.fit()` for more details about the allowed norms and optimization.

Examples

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

covs <- splitfactor(lalonge[c("age", "educ", "race",
                             "married", "nodegree")],
                   drop.first = FALSE)
```

```

targets <- c(23, 9, .3, .3, .4, .2, .5)

ows.fit <- optweight.svy.fit(covs,
                           targets = targets,
                           norm = "l2")

#Unweighted means
col_w_mean(covs)

#Weighted means; same as targets
col_w_mean(covs, w = ows.fit$w)

```

plot.optweight

Plot Dual Variables for Assessing Balance Constraints

Description

Plots the dual variables resulting from `optweight()` in a way similar to figure 2 of Zubizarreta (2015), which explained how to interpret these values. These represent the cost of changing the constraint on the variance of the resulting weights. For covariates with large values of the dual variable, tightening the constraint will increase the variability of the weights, and loosening 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.

Usage

```

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

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

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

```

Arguments

<code>x</code>	an <code>optweight</code> , <code>optweightMV</code> , or <code>optweight.svy</code> object; the output of a call to <code>optweight()</code> , <code>optweightMV()</code> , or <code>optweight.svy()</code> .
<code>...</code>	Ignored.
<code>which.treat</code>	For <code>optweightMV</code> objects, which treatment to display. Only one may be displayed at a time.

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("lalonge", package = "cobalt")

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

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

summary(ow1) # Note the RMSE Dev and effective
#           sample size (ESS)

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

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

summary(ow2) # Notice that tightening the constraint
#           on age had a negligible effect on the
#           variability of the weights and ESS

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

summary(ow3) # In contrast, tightening the constraint
#           on married had a large effect on the
#           variability of the weights, shrinking
#           the ESS
```

process_targets	<i>Construct and Check Targets Input</i>
-----------------	------------------------------------------

Description

Checks whether proposed target population means values for targets are suitable in number and order for submission to `optweight()` and `optweight.svy()`. Users should include one value per variable in formula. For factor variables, one value per level of the variable is required. The output of `process_targets()` can also be used as an input to targets in `optweight()` and `optweight.svy()`.

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 the covariates to be targeted on the right-hand side. See <code>glm()</code> for more details. Interactions and functions of covariates are allowed. Can be omitted, in which case all variables in data are assumed targeted.
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 <code>NULL</code> , the current sample means will be produced (weighted by <code>s.weights</code>). If <code>NA</code> , an <code>NA</code> vector named with the covariate names will be produced.
s.weights	an optional vector of sampling weights. Default is a vector of 1s.
...	ignored.
x	an <code>optweight.targets</code> object; the output of a call to <code>process_targets()</code> .
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()` 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 user inputs to `optweight()` 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()`. Users should include one value per item in formula. The output can also be used as an input to `tols` in `optweight()`.

Usage

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

check.tols(...)

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

Arguments

<code>formula</code>	a formula with the covariates to be balanced on the right-hand side. See <code>glm()</code> for more details. Interactions and functions of covariates are allowed. Lists of formulas are not allowed; multiple formulas must be checked one at a time.
<code>data</code>	an optional data set in the form of a data frame that contains the variables in formula.
<code>tols</code>	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.
<code>...</code>	ignored.
<code>x</code>	an <code>optweight.tols</code> object; the output of a call to <code>process_tols()</code> .
<code>internal</code>	logical; whether to print the tolerance values that are to be used internally by <code>optweight()</code> . See Value section.
<code>digits</code>	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.

`process_tols()` was designed to be used primarily for its message printing and `print()` method, but you can also assign its output to an object for use as an input to `tols` in `optweight()`.

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 user inputs to `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("lalonge", package = "cobalt")

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

tols

tols <- process_tols(treat ~ age + educ + married +
  nodegree + re74,
  data = lalonge,
  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	<i>Summarize, print, and plot information about estimated weights</i>
-------------------	-----------------------------------------------------------------------

Description

These functions summarize the weights resulting from a call to `optweight()` 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 (computing using the formula in McCaffrey, Rudgeway, & Morral, 2004). `plot()` creates a histogram of the weights.

Usage

```

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

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

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

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

```

Arguments

<code>object</code>	An <code>optweight</code> , <code>optweightMV</code> , or <code>optweight.svy</code> object; the output of a call to <code>optweight()</code> or <code>optweight.svy()</code> .
<code>top</code>	How many of the largest and smallest weights to display. Default is 5.
<code>ignore.s.weights</code>	Whether or not to ignore sampling weights when computing the weight summary. If <code>FALSE</code> , the default, the estimated weights will be multiplied by the sampling weights (if any) before values are computed.

weight.range	logical; whether 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 <code>graphics::hist()</code> to determine the number of bins, though <code>ggplot2::geom_histogram()</code> 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 L2 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 L1 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 (entropy norm), 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, a list of the above elements 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

check.targets (process_targets), 20
check.tols (process_tols), 22
clarabel::clarabel(), 11
clarabel::clarabel_control(), 11

ggplot2::geom_histogram(), 25
glm(), 3, 13, 20, 22
graphics::hist(), 25

highs::highs_available_solver_options(), 11
highs::highs_control(), 11
highs::highs_solve(), 11

lpSolve::lp(), 11

optweight, 2
optweight(), 8, 10, 12, 14, 15, 18–24
optweight.fit, 3, 8
optweight.fit(), 3, 4, 6, 14–17
optweight.svy, 12
optweight.svy(), 15, 17–21, 24
optweight.svy.fit, 14, 15
optweight.svy.fit(), 14, 15
optweightMV (optweight), 2
optweightMV(), 8, 10, 12, 18, 19
optweightMV.fit, 3
optweightMV.fit (optweight.fit), 8
optweightMV.fit(), 4
osqp::osqpSettings(), 11
osqp::solve_osqp(), 11

plot.optweight, 18
plot.optweight(), 25
plot.optweightMV (plot.optweight), 18
plot.summary.optweight
 (summary.optweight), 24
plot.summary.optweight(), 19
print.optweight.targets
 (process_targets), 20
print.optweight.tols (process_tols), 22

process_targets, 20
process_targets(), 3, 13, 23
process_tols, 22
process_tols(), 3, 13, 21

scs::scs(), 11
scs::scs_control(), 11
summary.optweight, 24
summary.optweightMV
 (summary.optweight), 24