

Package ‘WeightIt’

October 30, 2019

Type Package

Title Weighting for Covariate Balance in Observational Studies

Version 0.7.1

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

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

Description Generates weights to form equivalent groups in observational studies with point or longitudinal treatments by easing and extending the functionality of the R packages 'twang' for generalized boosted modeling (McCaffrey, Ridgeway & Morral, 2004) <[doi:10.1037/1082-989X.9.4.403](https://doi.org/10.1037/1082-989X.9.4.403)>, 'CBPS' for covariate balancing propensity score weighting (Imai & Ratkovic, 2014) <[doi:10.1111/rssb.12027](https://doi.org/10.1111/rssb.12027)>, 'ebal' for entropy balancing (Hainmueller, 2012) <[doi:10.1093/pan/mpr025](https://doi.org/10.1093/pan/mpr025)>, 'optweight' for optimization-based weights (Zu-bizarreta, 2015) <[doi:10.1080/01621459.2015.1023805](https://doi.org/10.1080/01621459.2015.1023805)>, 'ATE' for empirical balancing calibration weighting (Chan, Yam, & Zhang, 2016) <[doi:10.1111/rssb.12129](https://doi.org/10.1111/rssb.12129)>, and 'SuperLearner' for stacked machine learning-based propensity scores (Pirracchio, Petersen, & van der Laan, 2015) <[doi:10.1093/aje/kwu253](https://doi.org/10.1093/aje/kwu253)>. Also allows for assessment of weights and checking of covariate balance by interfacing directly with 'cobalt'.

Depends R (>= 3.3.0)

Imports ggplot2 (>= 3.0.0)

Suggests cobalt (>= 3.9.0), twang (>= 1.5), CBPS (>= 0.18), ebal (>= 0.1-6), ATE (>= 0.2.0), optweight (>= 0.2.4), SuperLearner (>= 2.0-25), class, gam, mlogit (>= 0.3-0), MNP (>= 3.1-0), brglm2 (>= 0.5.1), survey, jtools, boot, MASS, gbm (>= 2.1.3), knitr, rmarkdown

License GPL (>= 2)

Encoding UTF-8

LazyData true

URL <https://github.com/ngreifer/WeightIt>

BugReports <https://github.com/ngreifer/WeightIt/issues>

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2019-10-30 06:10:08 UTC

R topics documented:

as.weightit	2
get_w_from_ps	4
make_full_rank	6
method_cbps	7
method_ebal	9
method_ebcw	11
method_gbm	12
method_npcbps	15
method_optweight	17
method_ps	19
method_super	22
method_twang	25
method_user	27
ps.cont	30
sbps	34
summary.weightit	37
trim	38
weightit	40
weightitMSM	44

Index	48
--------------	-----------

as.weightit	<i>Create a weightit object manually</i>
-------------	--

Description

This function allows users to get the benefits of a weightit object when using weights not estimated with weightit() or weightitMSM(). These benefits include diagnostics, plots, and direct compatibility with **cobalt** for assessing balance.

Usage

```
as.weightit(...)

## Default S3 method:
as.weightit(weights,
             treat,
             covs = NULL,
             estimand = NULL,
             s.weights = NULL,
             ps = NULL,
             ...)

as.weightitMSM(...)
```

```
## Default S3 method:
as.weightitMSM(weights,
               treat.list,
               covs.list = NULL,
               estimand = NULL,
               s.weights = NULL,
               ps.list = NULL,
               ...)
```

Arguments

weights	required; a numeric vector of weights, one for each unit.
treat	required; a vector of treatment statuses, one for each unit.
covs	an optional data.frame of covariates. For using WeightIt functions, this is not necessary, but for use with cobalt it is.
estimand	an optional character of length 1 giving the estimand. The text is not checked.
s.weights	an optional numeric vector of sampling weights, one for each unit.
ps	an optional numeric vector of propensity scores, one for each unit.
treat.list	a list of treatment statuses at each time point..
covs.list	an optional list of data.frames of covariates of covariates at each time point. For using WeightIt functions, this is not necessary, but for use with cobalt it is.
ps.list	an optional list of numeric vectors of propensity scores at each time point.
...	additional arguments. These must be named. They will be included in the output object.

Value

An object of class `weightit` (for `as.weightit`) or `weightitMSM` (for `as.weightitMSM`).

Author(s)

Noah Greifer

Examples

```
treat <- rbinom(500, 1, .3)
weights <- rchisq(500, df = 2)
W <- as.weightit(weights= weights, treat = treat,
                estimand = "ATE")
summary(W)
```

get_w_from_ps	<i>Get weights from propensity scores corresponding to different estimands</i>
---------------	--

Description

Given a vector or matrix of propensity scores, output a vector of weights that target the provided estimand.

Usage

```
get_w_from_ps(ps,
              treat,
              estimand = "ATE",
              focal = NULL,
              treated = NULL)
```

Arguments

ps	A vector, matrix, or data frame of propensity scores. See Details.
treat	A vector of treatment statuses for each individual. See Details.
estimand	The desired estimand that the weights should target. Current options include "ATE" (average treatment effect), "ATT" (average treatment effect on the treated), "ATC" (average treatment effect on the control), "ATO" (average treatment effect in the overlap), and "ATM" (average treatment effect in the matched sample). See method_ps for additional information and references.
focal	When the estimand is the ATT or ATC, which group should be consider the (focal) "treated" or "control" group, respectively. If not NULL and estimand is not "ATT" or "ATC", estimand will automatically be set to "ATT".
treated	When treatment is binary, the value of treat that is considered the "treated" group (i.e., the group for which the propensity scores are the probability of being in). If NULL, get_w_from_ps will attempt to figure it out on its own using some heuristics. This really only matters when treat has values other than 0 and 1 and when ps is given as a vector or an unnamed single-column matrix or data frame.

Details

get_w_from_ps applies the formula for computing weights from propensity scores for the desired estimand. See the References section at [method_ps](#) for information on these estimands and the formulas.

ps can be entered a variety of ways. For binary treatments, when ps is entered as a vector or unnamed single-column matrix or data frame, get_w_from_ps has to know which value of treat corresponds to the "treated" group. For 0/1 variables, 1 will be considered treated. For other types of variables, get_w_from_ps will try to figure it out using heuristics, but it's safer to supply an argument to treated. When estimand is "ATT" or "ATC", supplying a value to focal is sufficient

(for ATT, focal is the treated group, and for ATC, focal is the control group). When entered as a matrix or data frame, the columns must be named with the levels of the treatment, and it is assumed that each column corresponds to the probability of being in that treatment group. This is the safest way to supply ps unless treat is a 0/1 variable.

For multinomial treatments, ps can be entered as a vector or a matrix or data frame. When entered as a vector, it is assumed the value corresponds to the probability of being in the treatment actually received; this is only possible when the estimand is "ATE". Otherwise, ps must be entered as a named matrix or data frame as described above for binary treatments. When the estimand is "ATT" or "ATC", a value for focal must be specified.

get_w_from_ps is not compatible with continuous treatments.

Value

A vector of weights.

Author(s)

Noah Greifer

See Also

[method_ps](#)

Examples

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

ps.fit <- glm(treat ~ age + educ + race + married +
             nodegree + re74 + re75, data = lalonge,
             family = binomial)
ps <- ps.fit$fitted.values

w1 <- get_w_from_ps(ps, treat = lalonge$treat,
                  estimand = "ATT")

treatAB <- factor(ifelse(lalonge$treat == 1, "A", "B"))
w2 <- get_w_from_ps(ps, treat = treatAB,
                  estimand = "ATT", focal = "A")
all.equal(w1, w2)
w3 <- get_w_from_ps(ps, treat = treatAB,
                  estimand = "ATT", treated = "A")
all.equal(w1, w3)

#A multinomial example using GBM predicted probabilities
library(gbm)
T3 <- factor(sample(c("A", "B", "C"), nrow(lalonge), replace = TRUE))

gbm.fit <- gbm(T3 ~ age + educ + race + married +
              nodegree + re74 + re75, data = lalonge,
              distribution = "multinomial", n.trees = 200,
```

```
interaction.depth = 3)
ps.multi <- drop(predict(gbm.fit, type = "response",
                        n.trees = 200))
w <- get_w_from_ps(ps.multi, T3, estimand = "ATE")
```

make_full_rank	<i>Make a matrix full rank</i>
----------------	--------------------------------

Description

When writing [user-defined methods](#) for use with `weightit`, it may be necessary to take the potentially non-full rank covs data frame and make it full rank for use in a downstream function. This function performs that operation.

Usage

```
make_full_rank(mat,
               with.intercept = TRUE)
```

Arguments

<code>mat</code>	a numeric matrix or data frame to be transformed. Typically this contains covariates. NAs are not allowed.
<code>with.intercept</code>	whether an intercept (i.e., a vector of 1s) should be added to <code>mat</code> before making it full rank. If TRUE, the intercept will be used in determining whether a column is linearly dependent on others. Regardless, no intercept will be included in the output.

Details

`make_full_rank` makes a matrix full rank by removing columns one at a time and determining whether the rank of the matrix changes. If it does not, that column is deleted. First, all columns that only contain one value are deleted. Then, if `with.intercept` is set to TRUE, an intercept column is added to the matrix. After determining which columns can be removed without changing the rank of the matrix, a matrix is returned with only those columns (and not the added intercept).

See example at [method_user](#).

Value

An object of the same type as `mat` containing only linearly independent columns.

Author(s)

Noah Greifer

See Also[method_user](#), [model.matrix](#)**Examples**

```
set.seed(1000)
c1 <- rbinom(10, 1, .4)
c2 <- 1-c1
c3 <- rnorm(10)
c4 <- 10*c3
mat <- data.frame(c1, c2, c3, c4)

make_full_rank(mat) #leaves c2 and c4

make_full_rank(mat, with.intercept = FALSE) #leaves c1, c2, and c4
```

`method_cbps`*Covariate Balancing Propensity Score Weighting*

Description

This page explains the details of estimating weights from covariate balancing propensity scores by setting `method = "cbps"` in the call to `weightit` or `weightitMSM`. This method can be used with binary, multinomial, and continuous treatments.

In general, this method relies on estimating propensity scores using generalized method of moments and then converting those propensity scores into weights using a formula that depends on the desired estimand. This method relies on [CBPS](#) from the **CBPS** package.

Binary Treatments: For binary treatments, this method estimates the propensity scores and weights using [CBPS](#). The following estimands are allowed: ATE, ATT, and ATC. The weights are taken from the output of the CBPS fit object. When the estimand is the ATE, the return propensity score is the probability of being in the "second" treatment group, i.e., `levels(factor(treat))[2]`; when the estimand is the ATC, the returned propensity score is the probability of being in the control (i.e., non-focal) group. When `include.obj = TRUE`, the returned object is the CBPS fit.

Multinomial Treatments: For multinomial treatments with three or four categories and when the estimand is the ATE, this method estimates the propensity scores and weights using one call to [CBPS](#). For multinomial treatments with three or four categories or when the estimand is the ATT, this method estimates the propensity scores and weights using multiple calls to [CBPS](#). The following estimands are allowed: ATE and ATT. The weights are taken from the output of the CBPS fit objects. When `include.obj = TRUE`, the returned object is either a single CBPS fit object if only one is used or a list of the CBPS fits if multiple are used.

Continuous Treatments: For continuous treatments, the generalized propensity score and weights are estimated using [CBPS](#). When `include.obj = TRUE`, the returned object is the CBPS fit.

Longitudinal Treatments: For longitudinal treatments, the weights are the product of the weights estimated at each time point. This is not how `CBMSM` in the `CBPS` package estimates weights for longitudinal treatments.

Sampling Weights: Sampling weights are supported through `s.weights` in all scenarios. See Note about sampling weights.

Missing Data: Missing data is not compatible with the CBPS algorithm, so a few extra things happen when NAs are present in the covariates. First, for each variable with missingness, a new missingness indicator variable is created which takes the value 1 if the original covariate is NA and 0 otherwise. The missingness indicators are added to the model formula as main effects. The missing values in the covariates are then replaced with 0s (this value is arbitrary and does not affect estimation). The weight estimation then proceeds with this new formula and set of covariates. The covariates output in the resulting `weightit` object will be the original covariates with the NAs.

Additional Arguments

All arguments to CBPS can be passed through `weightit` or `weightitMSM`, with the following exceptions:

`method` in CBPS is replaced with the argument `over` in `weightit`. Setting `over = FALSE` in `weightit` is the equivalent of setting `method = "exact"` in CBPS.

`sample.weights` is ignored because sampling weights are passed using `s.weights`.

`standardize` is ignored.

All arguments take on the defaults of those in CBPS. It may be useful in many cases to set `over = FALSE`, especially with continuous treatments.

Note

When sampling weights are used with CBPS, the estimated weights already incorporate the sampling weights. When `weightit` is used with `method = "cbps"`, the estimated weights are separated from the sampling weights, as they are with all other methods.

References

Binary treatments

Imai, K., & Ratkovic, M. (2014). Covariate balancing propensity score. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(1), 243–263.

Multinomial Treatments

Imai, K., & Ratkovic, M. (2014). Covariate balancing propensity score. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(1), 243–263.

Continuous treatments

Fong, C., Hazlett, C., & Imai, K. (2018). Covariate balancing propensity score for a continuous treatment: Application to the efficacy of political advertisements. *The Annals of Applied Statistics*, 12(1), 156–177. doi: [10.1214/17AOAS1101](https://doi.org/10.1214/17AOAS1101)

See Also[weightit](#), [weightitMSM](#)**Examples**

```

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

#Balancing covariates between treatment groups (binary)
(W1 <- weightit(treat ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "cbps", estimand = "ATT"))
summary(W1)
bal.tab(W1)

## Not run:
#Balancing covariates with respect to race (multinomial)
(W2 <- weightit(race ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "cbps", estimand = "ATE"))
summary(W2)
bal.tab(W2)

## End(Not run)

#Balancing covariates with respect to re75 (continuous)
(W3 <- weightit(re75 ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "cbps", over = FALSE))
summary(W3)
bal.tab(W3)

```

method_ebal

*Entropy Balancing***Description**

This page explains the details of estimating weights using entropy balancing by setting `method = "ebal"` in the call to [weightit](#) or [weightitMSM](#). This method can be used with binary and multinomial treatments.

In general, this method relies on estimating weights by minimizing the entropy of the weights subject to exact moment balancing constraints. This method relies on [ebalance](#) from the **ebal** package.

Binary Treatments: For binary treatments, this method estimates the weights using [ebalance](#). The following estimands are allowed: ATE, ATT, and ATC. The weights are taken from the output of the `ebalance` fit object. When the ATE is requested, `ebalance` is run twice, once for each treatment group. When `include.obj = TRUE`, the returned object is the `ebal` fit (or a list of the two fits when the estimand is the ATE).

Multinomial Treatments: For multinomial treatments, this method estimates the weights using `ebalance`. The following estimands are allowed: ATE and ATT. The weights are taken from the output of the `ebalance` fit objects. When the ATE is requested, `ebalance` is run once for each treatment group. When the ATT is requested, `ebalance` is run once for each non-focal (i.e., control) group. When `include.obj = TRUE`, the returned object is the list of `ebalance` fits.

Continuous Treatments: Continuous treatments are not supported.

Longitudinal Treatments: For longitudinal treatments, the weights are the product of the weights estimated at each time point. This method is not guaranteed to yield exact balance at each time point. NOTE: the use of entropy balancing with longitudinal treatments has not been validated!

Sampling Weights: Sampling weights are supported through `s.weights` in all scenarios.

Missing Data: Missing data is not compatible with the entropy balancing algorithm, so a few extra things happen when NAs are present in the covariates. First, for each variable with missingness, a new missingness indicator variable is created which takes the value 1 if the original covariate is NA and 0 otherwise. The missingness indicators are added to the model formula as main effects. The missing values in the covariates are then replaced with 0s (this value is arbitrary and does not affect estimation). The weight estimation then proceeds with this new formula and set of covariates. The covariates output in the resulting `weightit` object will be the original covariates with the NAs.

Additional Arguments

All arguments to `ebalance` can be passed through `weightit` or `weightitMSM` and take on the defaults of those in `ebalance`.

For `base.weights`, a vector with length equal to the total number of units can be supplied, in contrast to `ebalance()`, which requires a vector with length equal to the number of controls.

When `standardize = TRUE` in the call to `weightit`, `ebalance.trim` is run on the resulting `ebalance` fit objects. Doing so can reduce the variability of the weights while maintaining covariate balance.

References

Hainmueller, J. (2012). Entropy Balancing for Causal Effects: A Multivariate Reweighting Method to Produce Balanced Samples in Observational Studies. *Political Analysis*, 20(1), 25–46. doi: [10.1093/pan/mpr025](https://doi.org/10.1093/pan/mpr025)

See Also

[weightit](#), [weightitMSM](#)

Examples

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

#Balancing covariates between treatment groups (binary)
(W1 <- weightit(treat ~ age + educ + married +
```

```

        nodegree + re74, data = lalonde,
method = "ebal", estimand = "ATT"))
summary(W1)
bal.tab(W1)

#Balancing covariates with respect to race (multinomial)
(W2 <- weightit(race ~ age + educ + married +
        nodegree + re74, data = lalonde,
method = "ebal", estimand = "ATE",
standardize = TRUE))
summary(W2)
bal.tab(W2)

```

method_ebcw

*Empirical Balancing Calibration Weighting***Description**

This page explains the details of estimating weights using empirical balancing calibration weighting (EBCW) by setting `method = "ebcw"` in the call to `weightit`. This method can be used with binary and multinomial treatments.

In general, this method relies on estimating weights by minimizing a function of the weights subject to exact moment balancing constraints. This method relies on [ATE](#) from the [ATE](#) package.

Binary Treatments: For binary treatments, this method estimates the weights using [ATE](#) with `ATT = TRUE`. The following estimands are allowed: ATE, ATT, and ATC. The weights are taken from the output of the ATE fit object. When the ATE is requested, ATE is run twice, once for each treatment group. When `include.obj = TRUE`, the returned object is the `ebal` fit (or a list of the two fits when the estimand is the ATE).

Multinomial Treatments: For multinomial treatments, this method estimates the weights using [ATE](#) with `ATT = TRUE`. The following estimands are allowed: ATE and ATT. The weights are taken from the output of the ATE fit objects. When the ATE is requested, ATE is run once for each treatment group. When the ATT is requested, ATE is run once for each non-focal (i.e., control) group. When `include.obj = TRUE`, the returned object is the list of ATE fits.

Continuous Treatments: Continuous treatments are not supported.

Longitudinal Treatments: For longitudinal treatments, the weights are the product of the weights estimated at each time point. This method is not guaranteed to yield exact balance at each time point. NOTE: the use of EBCW with longitudinal treatments has not been validated!

Sampling Weights: Sampling weights are supported through `s.weights` in all scenarios.

Missing Data: Missing data is not compatible with the EBCW algorithm, so a few extra things happen when NAs are present in the covariates. First, for each variable with missingness, a new missingness indicator variable is created which takes the value 1 if the original covariate is NA and 0 otherwise. The missingness indicators are added to the model formula as main effects.

The missing values in the covariates are then replaced with 0s (this value is arbitrary and does not affect estimation). The weight estimation then proceeds with this new formula and set of covariates. The covariates output in the resulting `weightit` object will be the original covariates with the NAs.

Additional Arguments

All argument to ATE can be passed through `weightit`, with the following exceptions:

ATT is ignored because the estimand is passed using `estimand`.

All arguments take on the defaults of those in ATE.

References

Chan, K. C. G., Yam, S. C. P., & Zhang, Z. (2016). Globally efficient non-parametric inference of average treatment effects by empirical balancing calibration weighting. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(3), 673–700. doi: [10.1111/rssb.12129](https://doi.org/10.1111/rssb.12129)

See Also

[weightit](#)

Examples

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

#Balancing covariates between treatment groups (binary)
(W1 <- weightit(treat ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "ebcw", estimand = "ATT"))
summary(W1)
bal.tab(W1)

#Balancing covariates with respect to race (multinomial)
(W2 <- weightit(race ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "ebcw", estimand = "ATE"))
summary(W2)
bal.tab(W2)
```

Description

This page explains the details of estimating weights from generalized boosted model-based propensity scores by setting `method = "gbm"` in the call to `weightit` or `weightitMSM`. This method can be used with binary, multinomial, and continuous treatments.

In general, this method relies on estimating propensity scores using generalized boosted modeling and then converting those propensity scores into weights using a formula that depends on the desired estimand. The algorithm involves choosing a balance criterion to optimize so that balance, rather than prediction, is prioritized.

Binary Treatments: For binary treatments, this method estimates the propensity scores using `gbm.fit` and then optimizes balance using `col_w_smd` for standardized mean differences and `col_w_ks` for Kolmogorov-Smirnoff statistics, both from `cobalt`. The following estimands are allowed: ATE, ATT, ATC, ATO, and ATM. The weights are computed from the estimated propensity scores using `get_w_from_ps`, which implements the standard formulas. When `include.obj = TRUE`, the returned object is the `gbm` fit.

Multinomial Treatments: For multinomial treatments, this method estimates the propensity scores using `gbm.fit` with `distribution = "multinomial"` and then optimizes balance using `col_w_smd` for standardized mean differences and `col_w_ks` for Kolmogorov-Smirnoff statistics, both from `cobalt`. The following estimands are allowed: ATE, ATT, ATC, ATO, and ATM. The weights are computed from the estimated propensity scores using `get_w_from_ps`, which implements the standard formulas. The balance that is optimized is that between each non-focal treatment and the focal treatment for the ATT and ATC and between each treatment and the overall unweighted sample for other estimands. When `include.obj = TRUE`, the returned object is the `gbm` fit.

Continuous Treatments: For continuous treatments, the generalized propensity score is estimated using `gbm.fit`. When `include.obj = TRUE`, the returned object is the `gbm` fit.

Longitudinal Treatments: For longitudinal treatments, the weights are the product of the weights estimated at each time point.

Sampling Weights: Sampling weights are supported through `s.weights` in all scenarios.

Missing Data: Missing data is compatible with generalized boosted modeling. NA values are simply considered their own plausible value for the covariate. When balance statistics are computed to choose the optimal tree, they are computed using only the non-missing values for the variable in question.

Additional Arguments

The following additional arguments can be specified:

`stop.method` A string describing the criterion used to select the best weights. This has two parts: a statistics and a summarizing function, which should be combined as "`{stat},{summary}`". For binary and multinomial treatments, the available stats are "es" for absolute standardized mean differences and "ks" for Kolmogorov-Smirnoff statistics; for continuous treatments, the available stats are "p" for Pearson correlations between each covariate and the treatment and "s" for Spearman correlations. For all treatment types, the available summaries are "mean"

for the mean of the statistics, "max" for the maximum of the statistics, and "rms" for the root mean square of the statistics (i.e., the L2 norm). The default for binary and multinomial treatments is "es.mean" and the default for continuous treatments is "p.mean".

`trim.at` A number supplied to `at` in `trim` which trims the weights from all the trees before choosing the best tree. This can be valuable when some weights are extreme, which occur especially with continuous treatment. The default is 0 (i.e., no trimming).

`distribution` A string with the distribution used in the loss function of the boosted model. This is supplied to the `distribution` argument in `gbm.fit`. For binary treatments, "bernoulli" and "adaboost" are available, with "bernoulli" the default. For multinomial treatments, only "multinomial" is allowed. For continuous treatments "gaussian", "laplace", and "tdist" are available, with "gaussian" the default.

`n.trees` The maximum number of trees used. This is passed onto the `n.trees` argument in `gbm.fit`. The default is 10000 for binary and multinomial treatments and 20000 for continuous treatments.

`interaction.depth` The depth of the trees. This is passed onto the `interaction.depth` argument in `gbm.fit`. Higher values indicate better ability to capture nonlinear and nonadditive relationships. The default is 3 for binary and multinomial treatments and 4 for continuous treatments.

`shrinkage` The shrinkage parameter applied to the trees. This is passed onto the `shrinkage` argument in `gbm.fit`. The default is .01 for binary and multinomial treatments and .0005 for continuous treatments.

`bag.fraction` The fraction of the units randomly selected to propose the next tree in the expansion. This is passed onto the `bag.fraction` argument in `gbm.fit`. The default is 1, but smaller values should be tried.

All other arguments take on the defaults of those in `gbm.fit`, and some are not used at all.

The `w` argument in `gbm.fit` is ignored because sampling weights are passed using `s.weights`.

Note

This method used to implement what is now implemented by `method = "twang"`. Using `method = "gbm"` is *much* faster than using `method = "twang"` or `twang:ps` because of efficiency improvements that are not present in **twang**, but the results will differ slightly between the two methods. Notably, for multinomial treatments, the methods use different strategies: `method = "gbm"` estimates multinomial propensity scores (one for each individual for each treatment level) and chooses the tree that optimizes balance between each treatment and the target population, while `method = "twang"` estimates a set of binary propensity scores for each treatment, each optimizing balance for the treatment and the target population, and then combines the weights. For continuous treatments, the results will be identical.

References

Binary treatments

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](https://doi.org/10.1037/1082989X.9.4.403)

Multinomial Treatments

McCaffrey, D. F., Griffin, B. A., Almirall, D., Slaughter, M. E., Ramchand, R., & Burgette, L. F. (2013). A Tutorial on Propensity Score Estimation for Multiple Treatments Using Generalized Boosted Models. *Statistics in Medicine*, 32(19), 3388–3414. doi: [10.1002/sim.5753](https://doi.org/10.1002/sim.5753)

Continuous treatments

Zhu, Y., Coffman, D. L., & Ghosh, D. (2015). A Boosting Algorithm for Estimating Generalized Propensity Scores with Continuous Treatments. *Journal of Causal Inference*, 3(1).

See Also

[weightit](#), [weightitMSM](#), [method_twang](#)

Examples

```
# Examples take a long time to run
## Not run:
library("cobalt")
data("lalonde", package = "cobalt")

#Balancing covariates between treatment groups (binary)
(W1 <- weightit(treat ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "gbm", estimand = "ATE",
               stop.method = "es.max"))

summary(W1)
bal.tab(W1)

#Balancing covariates with respect to race (multinomial)
(W2 <- weightit(race ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "gbm", estimand = "ATT",
               focal = "hispan", stop.method = "ks.mean"))

summary(W2)
bal.tab(W2)

#Balancing covariates with respect to re75 (continuous)
(W3 <- weightit(re75 ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "gbm", use.kernel = TRUE,
               stop.method = "p.rms", trim.at = .97))

summary(W3)
bal.tab(W3)

## End(Not run)
```

Description

This page explains the details of estimating weights from nonparametric covariate balancing propensity scores by setting `method = "npcbps"` in the call to `weightit` or `weightitMSM`. This method can be used with binary, multinomial, and continuous treatments.

In general, this method relies on estimating weights by maximizing the empirical likelihood of the data subject to balance constraints. This method relies on `npCBPS` from the `CBPS` package.

Binary Treatments: For binary treatments, this method estimates the weights using `npCBPS`. The ATE is the only estimand allowed. The weights are taken from the output of the `npCBPS` fit object. When `include.obj = TRUE`, the returned object is the `npCBPS` fit.

Multinomial Treatments: For multinomial treatments, this method estimates the weights using `npCBPS`. The ATE is the only estimand allowed. The weights are taken from the output of the `npCBPS` fit object. When `include.obj = TRUE`, the returned object is the `npCBPS` fit.

Continuous Treatments: For continuous treatments, this method estimates the weights using `npCBPS`. The weights are taken from the output of the `npCBPS` fit object. When `include.obj = TRUE`, the returned object is the `npCBPS` fit.

Longitudinal Treatments: For longitudinal treatments, the weights are the product of the weights estimated at each time point. This is not how `CBMSM` in the `CBPS` package estimates weights for longitudinal treatments.

Sampling Weights: Sampling weights are supported through `s.weights` in all scenarios.

Missing Data: Missing data is not compatible with the nonparametric CBPS algorithm, so a few extra things happen when NAs are present in the covariates. First, for each variable with missingness, a new missingness indicator variable is created which takes the value 1 if the original covariate is NA and 0 otherwise. The missingness indicators are added to the model formula as main effects. The missing values in the covariates are then replaced with 0s (this value is arbitrary and does not affect estimation). The weight estimation then proceeds with this new formula and set of covariates. The covariates output in the resulting `weightit` object will be the original covariates with the NAs.

Additional Arguments

All argument to `npCBPS` can be passed through `weightit` or `weightitMSM`.

All arguments take on the defaults of those in `npCBPS`.

References

Fong, C., Hazlett, C., & Imai, K. (2018). Covariate balancing propensity score for a continuous treatment: Application to the efficacy of political advertisements. *The Annals of Applied Statistics*, 12(1), 156–177. doi: [10.1214/17AOAS1101](https://doi.org/10.1214/17AOAS1101)

See Also

[weightit](#), [weightitMSM](#)

Examples

```
# Examples take a long time to run
## Not run:
library("cobalt")
data("lalonde", package = "cobalt")

#Balancing covariates between treatment groups (binary)
(W1 <- weightit(treat ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "npcbps", estimand = "ATE"))
summary(W1)
bal.tab(W1)

#Balancing covariates with respect to race (multinomial)
(W2 <- weightit(race ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "npcbps", estimand = "ATE"))
summary(W2)
bal.tab(W2)

## End(Not run)
```

 method_optweight

Optimization-Based Weighting

Description

This page explains the details of estimating optimization-based weights by setting `method = "optweight"` in the call to `weightit` or `weightitMSM`. This method can be used with binary, multinomial, and continuous treatments.

In general, this method relies on estimating weights by solving a quadratic programming problem subject to approximate or exact balance constraints. This method relies on `optweight` from the **optweight** package.

Because `optweight` offers finer control and uses the same syntax as `weightit`, it is recommended that `optweight` be used instead of `weightit` with `method = "optweight"`.

Binary Treatments: For binary treatments, this method estimates the weights using `optweight`. The following estimands are allowed: ATE, ATT, and ATC. The weights are taken from the output of the `optweight` fit object. When `include.obj = TRUE`, the returned object is the `optweight` fit.

Multinomial Treatments: For multinomial treatments, this method estimates the weights using `optweight`. The following estimands are allowed: ATE and ATT. The weights are taken from the output of the `optweight` fit object. When `include.obj = TRUE`, the returned object is the `optweight` fit.

Continuous Treatments: For binary treatments, this method estimates the weights using `optweight`. The weights are taken from the output of the `optweight` fit object. When `include.obj = TRUE`, the returned object is the `optweight` fit.

Longitudinal Treatments: For longitudinal treatments, `optweight` estimates weights that simultaneously satisfy balance constraints at all time points, so only one model is fit to obtain the weights. Using `method = "optweight"` in `weightitMSM` cause `is.MSM.method` to be set to `TRUE` by default. Setting it to `FALSE` will run one model for each time point and multiply the weights together, a method that is not recommended. NOTE: neither use of optimization-based weights with longitudinal treatments has been validated!

Sampling Weights: Sampling weights are supported through `s.weights` in all scenarios.

Missing Data: Missing data is not compatible with the optimization-based weighting algorithm, so a few extra things happen when NAs are present in the covariates. First, for each variable with missingness, a new missingness indicator variable is created which takes the value 1 if the original covariate is NA and 0 otherwise. The missingness indicators are added to the model formula as main effects. The missing values in the covariates are then replaced with 0s (this value is arbitrary and does not affect estimation). The weight estimation then proceeds with this new formula and set of covariates. The covariates output in the resulting `weightit` object will be the original covariates with the NAs.

Additional Arguments

All arguments to `optweight` can be passed through `weightit` or `weightitMSM`, with the following exception:

`targets` cannot be used and is ignored.

All arguments take on the defaults of those in `optweight`.

Note

The specification of `tols` differs between `weightit` and `optweight`. In `weightit`, one tolerance value should be included per level of each factor variable, whereas in `optweight`, all levels of a factor are given the same tolerance, and only one value needs to be supplied for a factor variable. Because of the potential for confusion and ambiguity, it is recommended to only supply one value for `tols` in `weightit` that applies to all variables. For finer control, use `optweight` directly.

With point treatments, the returned fit object when `include.obj = TRUE` is an `optweight` obj, but with longitudinal treatments, the returned object is the output of a call to `optweight.fit`.

Seriously, just use `optweight`.

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

[weightit](#), [weightitMSM](#)

Examples

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

#Balancing covariates between treatment groups (binary)
(W1 <- weightit(treat ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "optweight", estimand = "ATT",
               tols = 0))

summary(W1)
bal.tab(W1)

#Balancing covariates with respect to race (multinomial)
(W2 <- weightit(race ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "optweight", estimand = "ATE",
               tols = .01))

summary(W2)
bal.tab(W2)

#Balancing covariates with respect to re75 (continuous)
(W3 <- weightit(re75 ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "optweight", tols = .05))

summary(W3)
bal.tab(W3)
```

 method_ps

Propensity Score Weighting Using Generalized Linear Models

Description

This page explains the details of estimating weights from generalized linear model-based propensity scores by setting `method = "ps"` in the call to `weightit` or `weightitMSM`. This method can be used with binary, multinomial, and continuous treatments.

In general, this method relies on estimating propensity scores with a parametric generalized linear model and then converting those propensity scores into weights using a formula that depends on the desired estimand. For binary and multinomial treatments, a binomial or multinomial regression model is used to estimate the propensity scores as the predicted probability of being in each treatment given the covariates. For continuous treatments, a generalized linear model is used to estimate generalized propensity scores as the conditional density of treatment given the covariates.

Binary Treatments: For binary treatments, this method estimates the propensity scores using `glm`. An additional argument is `link`, which uses the same options as `link` in `family`. The default link is "logit", but others, including "probit", are allowed. The following estimands are allowed: ATE, ATT, ATC, ATO, and ATM. The weights for the ATE, ATT, and ATC are computed from the estimated propensity scores using the standard formulas, the weights for the ATO are computed as in Li & Li (2018), and the weights for the ATM (i.e., average treatment effect in the

equivalent sample "pair-matched" with calipers) are computed as in Yoshida et al (2017). When `include.obj = TRUE`, the returned object is the `glm` fit.

Multinomial Treatments: For multinomial treatments, the propensity scores are estimated using multinomial regression from one of a few functions depending on the requested link: for logit ("`logit`") and probit ("`probit`") links, `mlogit` from the **mlogit** package is used; for the Bayesian probit ("`bayes.probit`") link, `mnp` from the **MNP** package is used; and for the biased-reduced multinomial logistic regression ("`br.logit`"), `brmultinom` from the **brglm2** package is used. If the treatment variable is an ordered factor, `polr` from the **MASS** package is used to fit ordinal regression. Any of the methods allowed in the `method` argument of `polr` can be supplied to `link`. The following estimands are allowed: ATE, ATT, ATC, ATO, and ATM. The weights for each estimand are computed using the standard formulas or those mentioned above. When `include.obj = TRUE`, the returned object is the fit object from the fitting function used.

Continuous Treatments: For continuous treatments, the generalized propensity score is estimated using linear regression. In addition, kernel density estimation can be used instead of assuming a normal density for the numerator and denominator of the generalized propensity score by setting `use.kernel = TRUE`. Other arguments to `density` can be specified to refine the density estimation parameters. `plot = TRUE` can be specified to plot the density for the numerator and denominator, which can be helpful in diagnosing extreme weights. When `include.obj = TRUE`, the returned object is the `glm` fit from denominator model.

Longitudinal Treatments: For longitudinal treatments, the weights are the product of the weights estimated at each time point.

Sampling Weights: Sampling weights are supported through `s.weights` in all scenarios except for multinomial treatments with `link = "bayes.probit"`. Warning messages may appear otherwise about non-integer successes, and these can be ignored.

Missing Data: Missing data is not directly compatible with estimating propensity scores, so a few extra things happen when NAs are present in the covariates. First, for each variable with missingness, a new missingness indicator variable is created which takes the value 1 if the original covariate is NA and 0 otherwise. The missingness indicators are added to the model formula as main effects. The missing values in the covariates are then replaced with 0s (this value is arbitrary and does not affect estimation). The weight estimation then proceeds with this new formula and set of covariates. The covariates output in the resulting `weightit` object will be the original covariates with the NAs.

Additional Arguments

The following additional arguments can be specified:

`link` The link used in the generalized linear model for the propensity scores. For binary treatments, `link` can be any of those allowed by `binomial`. A `br.` prefix can be added (e.g., "`br.logit`"); this changes the fitting method to the [bias-corrected generalized linear models](#) implemented in the **brglm2** package. For multinomial treatments, `link` can be "`logit`", "`probit`", "`bayes.probit`", or "`br.logit`". For ordered treatments, `link` can be any of those allowed by the `method` argument of `polr`. For continuous treatments, `link` can be any of those allowed by `gaussian`.

- `use.kernel` If TRUE, uses kernel density estimation through `density` to estimate the numerator and denominator densities for the weights with continuous treatments. If FALSE, assumes a normal distribution.
- `bw`, `adjust`, `kernel`, `n` If `use.kernel = TRUE` with continuous treatments, the arguments to `density`. The defaults are the same as those in `density` except that `n` is 10 times the number of units in the sample.
- `plot` If `use.kernel = TRUE` with continuous treatments, whether to plot the estimated density. For binary treatments, additional arguments to `glm` can be specified as well. The `method` argument in `glm` is renamed to `glm.method`. This can be used to supply alternative fitting functions, such as those implemented in the **glm2** package. Other arguments to `weightit` are massed to ... in `glm`.

References

Binary treatments

- `estimand = "ATO"`

Li, F., Morgan, K. L., & Zaslavsky, A. M. (2018). Balancing covariates via propensity score weighting. *Journal of the American Statistical Association*, 113(521), 390–400. doi: [10.1080/01621459.2016.1260466](https://doi.org/10.1080/01621459.2016.1260466)

- `estimand = "ATM"`

Li, L., & Greene, T. (2013). A Weighting Analogue to Pair Matching in Propensity Score Analysis. *The International Journal of Biostatistics*, 9(2). doi: [10.1515/ijb20120030](https://doi.org/10.1515/ijb20120030)

- Other estimands

Austin, P. C. (2011). An Introduction to Propensity Score Methods for Reducing the Effects of Confounding in Observational Studies. *Multivariate Behavioral Research*, 46(3), 399–424. doi: [10.1080/00273171.2011.568786](https://doi.org/10.1080/00273171.2011.568786)

Multinomial Treatments

- `estimand = "ATO"`

Li, F., & Li, F. (2018). Propensity Score Weighting for Causal Inference with Multi-valued Treatments. *ArXiv:1808.05339 [Stat]*. Retrieved from <http://arxiv.org/abs/1808.05339>

- `estimand = "ATM"`

Yoshida, K., Hernández-Díaz, S., Solomon, D. H., Jackson, J. W., Gagne, J. J., Glynn, R. J., & Franklin, J. M. (2017). Matching weights to simultaneously compare three treatment groups: Comparison to three-way matching. *Epidemiology (Cambridge, Mass.)*, 28(3), 387–395. doi: [10.1097/EDE.0000000000000627](https://doi.org/10.1097/EDE.0000000000000627)

- Other estimands

McCaffrey, D. F., Griffin, B. A., Almirall, D., Slaughter, M. E., Ramchand, R., & Burgette, L. F. (2013). A Tutorial on Propensity Score Estimation for Multiple Treatments Using Generalized Boosted Models. *Statistics in Medicine*, 32(19), 3388–3414. doi: [10.1002/sim.5753](https://doi.org/10.1002/sim.5753)

Continuous treatments

`method = "ps"`

Robins, J. M., Hernán, M. Á., & Brumback, B. (2000). Marginal Structural Models and Causal Inference in Epidemiology. *Epidemiology*, 11(5), 550–560.

See Also

[weightit](#), [weightitMSM](#)

Examples

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

#Balancing covariates between treatment groups (binary)
(W1 <- weightit(treat ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "ps", estimand = "ATT",
               link = "probit"))

summary(W1)
bal.tab(W1)

#Balancing covariates with respect to race (multinomial)
(W2 <- weightit(race ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "ps", estimand = "ATE"))

summary(W2)
bal.tab(W2)

#Balancing covariates with respect to re75 (continuous)
(W3 <- weightit(re75 ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "ps", use.kernel = TRUE))

summary(W3)
bal.tab(W3)
```

method_super

Propensity Score Weighting Using SuperLearner

Description

This page explains the details of estimating weights from SuperLearner-based propensity scores by setting `method = "super"` in the call to [weightit](#) or [weightitMSM](#). This method can be used with binary, multinomial, and continuous treatments.

In general, this method relies on estimating propensity scores using the SuperLearner algorithm for stacking predictions and then converting those propensity scores into weights using a formula that depends on the desired estimand. For binary and multinomial treatments, one or more binary classification algorithms are used to estimate the propensity scores as the predicted probability of being in each treatment given the covariates. For continuous treatments, a regression algorithm is used to estimate generalized propensity scores as the conditional density of treatment given the covariates.

Binary Treatments: For binary treatments, this method estimates the propensity scores using [SuperLearner](#) in the **SuperLearner** package. The following estimands are allowed: ATE, ATT,

ATE, ATO, and ATM. The weights for the ATE, ATT, and ATC are computed from the estimated propensity scores using the standard formulas, the weights for the ATO are computed as in Li & Li (2018), and the weights for the ATM (i.e., average treatment effect in the equivalent sample "pair-matched" with calipers) are computed as in Yoshida et al (2017). When `include.obj = TRUE`, the returned object is the SuperLearner fit.

Multinomial Treatments: For multinomial treatments, the propensity scores are estimated using several calls to `SuperLearner`, one for each treatment group, and the treatment probabilities are normalized to sum to 1. The following estimands are allowed: ATE, ATT, ATO, and ATM. The weights for each estimand are computed using the standard formulas or those mentioned above. When `include.obj = TRUE`, the returned object is the list of fit object from the SuperLearner calls.

Continuous Treatments: For continuous treatments, the generalized propensity score is estimated using `SuperLearner`. In addition, kernel density estimation can be used instead of assuming a normal density for the numerator and denominator of the generalized propensity score by setting `use.kernel = TRUE`. Other arguments to `density` can be specified to refine the density estimation parameters. `plot = TRUE` can be specified to plot the density for the numerator and denominator, which can be helpful in diagnosing extreme weights. When `include.obj = TRUE`, the returned object is the `SuperLearner` fit from denominator model.

Longitudinal Treatments: For longitudinal treatments, the weights are the product of the weights estimated at each time point.

Sampling Weights: Sampling weights are supported through `s.weights` in all scenarios.

Missing Data: Missing data is not compatible with SuperLearner, so a few extra things happen when NAs are present in the covariates. First, for each variable with missingness, a new missingness indicator variable is created which takes the value 1 if the original covariate is NA and 0 otherwise. The missingness indicators are added to the model formula as main effects. The missing values in the covariates are then replaced with 0s (this value is arbitrary and does not affect estimation). The weight estimation then proceeds with this new formula and set of covariates. The covariates output in the resulting `weightit` object will be the original covariates with the NAs.

Additional Arguments

An argument to `SL.library` must be supplied. To see a list of available entries, use `listWrappers`.

All arguments to SuperLearner can be passed through `weightit` or `weightitMSM`, with the following exceptions:

`method` in SuperLearner is replaced with the argument `SL.method` in `weightit`.

`obsWeights` is ignored because sampling weights are passed using `s.weights`.

The following additional arguments can be specified:

`use.kernel` If `TRUE`, uses kernel density estimation through `density` to estimate the numerator and denominator densities for the weights with continuous treatments. If `FALSE`, assumes a normal distribution.

`bw`, `adjust`, `kernel`, `n` If `use.kernel = TRUE` with continuous treatments, the arguments to `density`. The defaults are the same as those in `density` except that `n` is 10 times the number of units in the sample.

`plot` If `use.kernel = TRUE` with continuous treatments, whether to plot the estimated density.

Note

Some methods formerly available in **SuperLearner** are now in SuperLearnerExtra, which can be found on GitHub at <https://github.com/ecpolley/SuperLearnerExtra>.

References

Pirracchio, R., Petersen, M. L., & van der Laan, M. (2015). Improving Propensity Score Estimators' Robustness to Model Misspecification Using Super Learner. *American Journal of Epidemiology*, 181(2), 108–119. doi: [10.1093/aje/kwu253](https://doi.org/10.1093/aje/kwu253)

See Also

[weightit](#), [weightitMSM](#)

Examples

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

#Balancing covariates between treatment groups (binary)
(W1 <- weightit(treat ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "super", estimand = "ATT",
               SL.library = c("SL.glm", "SL.gam",
                             "SL.knn"))))

summary(W1)
bal.tab(W1)

#Balancing covariates with respect to race (multinomial)
(W2 <- weightit(race ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "super", estimand = "ATE",
               SL.library = c("SL.glm", "SL.gam",
                             "SL.knn"))))

summary(W2)
bal.tab(W2)

#Balancing covariates with respect to re75 (continuous)
(W3 <- weightit(re75 ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "super", use.kernel = TRUE,
               SL.library = c("SL.glm", "SL.gam",
                             "SL.ridge"))))

summary(W3)
bal.tab(W3)
```

`method_twang`*Propensity Score Weighting Using Generalized Boosted Models with TWANG*

Description

This page explains the details of estimating weights from generalized boosted model-based propensity scores by setting `method = "twang"` in the call to `weightit` or `weightitMSM`. This method can be used with binary, multinomial, and continuous treatments.

In general, this method relies on estimating propensity scores using generalized boosted modeling and then converting those propensity scores into weights using a formula that depends on the desired estimand. The algorithm involves choosing a balance criterion to optimize so that balance, rather than prediction, is prioritized. For binary and multinomial treatments, this method relies on `ps` from the `twang` package. For continuous treatments, this method relies on `ps.cont` from `WeightIt`.

Binary Treatments: For binary treatments, this method estimates the propensity scores using `ps`. The following estimands are allowed: ATE, ATT, and ATC. The weights for the ATE, ATT, and ATC are computed from the estimated propensity scores using the standard formulas. When the estimand is the ATE, the return propensity score is the probability of being in the "second" treatment group, i.e., `levels(factor(treat))[2]`; when the estimand is the ATC, the returned propensity score is the probability of being in the control (i.e., non-focal) group. When `include.obj = TRUE`, the returned object is the `ps` fit.

Multinomial Treatments: For multinomial treatments, this method estimates the propensity scores using multiple calls to `ps` (which is the same thing that `mnps` in `twang` does behind the scenes). The following estimands are allowed: ATE and ATT. The weights are computed from the estimated propensity scores using the standard formulas. When `include.obj = TRUE`, the returned object is a list of the `ps` fits.

Continuous Treatments: For continuous treatments, the generalized propensity score is estimated using `ps.cont`. When `include.obj = TRUE`, the returned object is the `ps.cont` fit.

Longitudinal Treatments: For longitudinal treatments, the weights are the product of the weights estimated at each time point, which is what `iptw` in `twang` does behind the scenes.

Sampling Weights: Sampling weights are supported through `s.weights` in all scenarios.

Missing Data: Missing data is compatible with generalized boosted modeling. NA values are simply considered their own plausible value for the covariate. When balance statistics are computed to choose the optimal tree, they are computed using only the non-missing values for the variable in question.

Additional Arguments

Any argument to `ps` or `ps.cont` can be passed through `weightit` or `weightitMSM`. The argument `stop.method` is required, and, in contrast to `ps` and `ps.cont`, can only be of length 1. If not provided, a default will be used ("`es.mean`" for binary and multinomial treatments and "`p.mean`" for continuous treatments).

sampw is ignored because sampling weights are passed using `s.weights`.

All other arguments take on the defaults of those in `ps` and `ps.cont`. The main arguments of interest are `n.trees` and `interaction.depth`, and increasing them can improve performance.

Note

See the Note section at [method_gbm](#) for a discussion how this method differs from `method = "gbm"`.

References

Binary treatments

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](https://doi.org/10.1037/1082989X.9.4.403)

Multinomial Treatments

McCaffrey, D. F., Griffin, B. A., Almirall, D., Slaughter, M. E., Ramchand, R., & Burgette, L. F. (2013). A Tutorial on Propensity Score Estimation for Multiple Treatments Using Generalized Boosted Models. *Statistics in Medicine*, 32(19), 3388–3414. doi: [10.1002/sim.5753](https://doi.org/10.1002/sim.5753)

Continuous treatments

Zhu, Y., Coffman, D. L., & Ghosh, D. (2015). A Boosting Algorithm for Estimating Generalized Propensity Scores with Continuous Treatments. *Journal of Causal Inference*, 3(1).

See Also

[weightit](#), [weightitMSM](#), [method_gbm](#)

Examples

```
# Examples take a while to run
## Not run:
library("cobalt")
data("lalonge", package = "cobalt")

#Balancing covariates between treatment groups (binary)
(W1 <- weightit(treat ~ age + educ + married +
               nodegree + re74, data = lalonge,
               method = "twang", estimand = "ATT",
               stop.method = "es.max"))

summary(W1)
bal.tab(W1)

#Balancing covariates with respect to race (multinomial)
(W2 <- weightit(race ~ age + educ + married +
               nodegree + re74, data = lalonge,
               method = "twang", estimand = "ATE",
               stop.method = "ks.mean"))

summary(W2)
bal.tab(W2)
```

```

#Balancing covariates with respect to re75 (continuous)
(W3 <- weightit(re75 ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "twang", use.kernel = TRUE,
               stop.method = "p.max"))

summary(W3)
bal.tab(W3)

## End(Not run)

```

method_user

User-Defined Functions for Estimating Weights

Description

This page explains the details of estimating weights using a user-defined function. The function must take in arguments that are passed to it by `weightit` or `weightitMSM` and return a vector of weights or a list containing the weights.

To supply user-defined function, the function object should be entered directly to `method`; for example, for a function `fun`, `method = fun`.

Point Treatments: The following arguments are automatically passed to the user-defined function, which should have named parameters corresponding to them:

- `treat`: a vector of treatment status for each unit. This comes directly from the left hand side of the formula passed to `weightit` and so will have its type (e.g., numeric, factor, etc.), which may need to be converted.
- `covs`: a data frame of covariate values for each unit. This comes directly from the right hand side of the formula passed to `weightit`. The covariates are processed so that all columns are numeric; all factor variables are split into dummies and all interactions are evaluated. All levels of factor variables are given dummies, so the matrix of the covariates is not full rank. Users can use `make_full_rank`, which accepts a numeric matrix or data frame and removes columns to make it full rank, if a full rank covariate matrix is desired.
- `s.weights`: a numeric vector of sampling weights, one for each unit.
- `ps`: a numeric vector of propensity scores.
- `subset`: a logical vector the same length as `treat` that is `TRUE` for units to be included in the estimation and `FALSE` otherwise. This is used to subset the input objects when `exact` is used. `treat`, `covs`, `s.weights`, and `ps`, if supplied, will already have been subsetted by `subset`.
- `estimand`: a character vector of length 1 containing the desired estimand. The characters will have been converted to uppercase. If "ATC" was supplied to `estimand`, `weightit` sets `focal` to the control level (usually 0 or the lowest level of `treat`) and sets `estimand` to "ATT".
- `focal`: a character vector of length 1 containing the focal level of the treatment when the estimand is the ATT (or the ATC as detailed above). `weightit` ensures the value of `focal` is a level of `treat`.
- `stabilize`: a logical vector of length 1. It is not processed by `weightit` before it reaches the fitting function.

- `moments`: a numeric vector of length 1. It is not processed by `weightit` before it reaches the fitting function except that `as.integer` is applied to it. This is used in other methods to determine whether polynomials of the entered covariates are to be used in the weight estimation.
- `int`: a logical vector of length 1. It is not processed by `weightit` before it reaches the fitting function. This is used in other methods to determine whether interactions of the entered covariates are to be used in the weight estimation.

None of these parameters are required to be in the fitting function. These are simply those that are automatically available.

In addition, any additional arguments supplied to `weightit` will be passed on to the fitting function. `weightit` ensures the arguments correspond to the parameters of the fitting function and throws an error if an incorrectly named argument is supplied and the fitting function doesn't include ... as a parameter.

The fitting function must output either a numeric vector of weights or a list (or list-like object) with an entry named wither "w" or "weights". If a list, the list can contain other named entries, but only entries named "w", "weights", "ps", and "fit.obj" will be processed. "ps" is a vector of propensity scores and "fit.obj" should be an object used in the fitting process that a user may want to examine and that is included in the `weightit` output object as "obj" when `include.obj = TRUE`. The "ps" and "fit.obj" components are optional, but "weights" or "w" is required.

Longitudinal Treatments: Longitudinal treatments can be handled either by running the fitting function for point treatments for each time point and multiplying the resulting weights together or by running a method that accommodates multiple time points and outputs a single set of weights. For the former, `weightitMSM` can be used with the user-defined function just as it is with `weightit`. The latter method is not yet accommodated by `weightitMSM`, but will be someday, maybe.

See Also

[weightit](#), [weightitMSM](#)

Examples

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

#A user-defined version of method = "ps"
my.ps <- function(treat, covs, estimand, focal = NULL) {
  covs <- make_full_rank(covs)
  d <- data.frame(treat, covs)
  f <- formula(d)
  ps <- glm(f, data = d, family = "binomial")$fitted
  w <- get_w_from_ps(ps, treat = treat, estimand = estimand,
                    focal = focal)

  return(list(w = w, ps = ps))
}

#Balancing covariates between treatment groups (binary)
```

```

(W1 <- weightit(treat ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = my.ps, estimand = "ATT"))
summary(W1)
bal.tab(W1)

#Balancing covariates for longitudinal treatments
# my.ps is used at each time point.
library("twang")
data("iptwExWide", package = "twang")
(W2 <- weightitMSM(list(tx1 ~ age + gender + use0,
                       tx2 ~ tx1 + use1 + age + gender + use0,
                       tx3 ~ tx2 + use2 + tx1 + use1 + age + gender + use0),
                 data = iptwExWide,
                 method = my.ps))

summary(W2)
bal.tab(W2)

# Kernel balancing using the KBAL package, available
# using devtools::install_github("chadazlett/KBAL").
# Only the ATT and ATC are available. Use 'kbal.method'
# instead of 'method' in weightit() to choose between
# "ebal" and "el".

## Not run:
kbal.fun <- function(treat, covs, estimand, focal, ...) {
  args <- list(...)
  if (is_not_null(focal))
    treat <- as.numeric(treat == focal)
  else if (estimand != "ATT")
    stop("estimand must be 'ATT' or 'ATC'.", call. = FALSE)
  if ("kbal.method" %in% names(args)) {
    names(args)[names(args) == "kbal.method"] <- "method"
  }
  args[names(args) %nin% setdiff(names(formals(KBAL::kbal)),
                                c("X", "D"))] <- NULL
  k.out <- do.call(KBAL::kbal, c(list(X = covs, D = treat),
                                args))
  w <- k.out$w
  return(list(w = w))
}

(Wk <- weightit(treat ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = kbal.fun, estimand = "ATT",
               kbal.method = "ebal"))

summary(Wk)
bal.tab(Wk, disp.ks = TRUE)

## End(Not run)

```

ps.cont

*Generalized Propensity Score Estimation using GBM***Description**

ps.cont calculates generalized propensity scores and corresponding weights using boosted linear regression as implemented in [gbm](#). This function extends [ps](#) in [twang](#) to continuous treatments. The syntax and output are largely the same. The GBM parameter defaults are those found in [Zhu, Coffman, & Ghosh \(2015\)](#).

Note: ps.cont will phased out when [twang](#) adds functionality for continuous treatments.

Usage

```
ps.cont(formula, data,
        n.trees = 20000,
        interaction.depth = 4,
        shrinkage = 0.0005,
        bag.fraction = 1,
        print.level = 0,
        verbose = FALSE,
        stop.method,
        sampw = NULL,
        optimize = 1,
        use.kernel = FALSE,
        ...)
## S3 method for class 'ps.cont'
summary(object, ...)
## S3 method for class 'ps.cont'
plot(x, ...)
## S3 method for class 'ps.cont'
boxplot(x, ...)
```

Arguments

formula	A formula for the propensity score model with the treatment indicator on the left side of the formula and the potential confounding variables on the right side.
data	The dataset in the form of a data frame, which should include treatment assignment as well as the covariates specified in formula.
n.trees	The number of GBM iterations passed on to gbm . The more, the better the final solution will be, but the more time it will take.
interaction.depth	The interaction.depth passed on to gbm .
shrinkage	The shrinkage passed on to gbm .
bag.fraction	The bag.fraction passed on to gbm .

print.level	Currently ignored.
verbose	If TRUE, information will be printed to monitor the the progress of the fitting.
stop.method	A method or methods of measuring and summarizing balance across pretreatment variables. Current options are p.max, p.mean, p.rms, s.max, s.mean, and s.rms. p refers to the Pearson correlation and s refers to the Spearman correlation. These are summarized across the pretreatment variables by the maximum (max), the mean (mean), or the square root of the mean of the squares (rms).
sampw	Optional sampling weights.
optimize	A numeric value, either 0, 1, or 2. If 0, balance will be checked for every tree, and the tree with the best balance will be the one used to generate the final weights. If 1, the default, balance will be checked for a subset of trees, and then <code>optimize</code> will be used to find the tree with the best balance within the tree interval chosen. If 2, <code>optimize</code> will be used to find the tree that yields the best balance. 0 takes the longest but is guaranteed to find the best balance among the trees. 2 is the quickest but will often choose a tree that that suboptimal balance, though not by much. 1 is a compromise between speed and comprehensiveness and is the algorithm implemented in twang .
use.kernel	Whether to use kernel density estimation as implemented in <code>density</code> to estimate the numerator of the weights. If TRUE, <code>density</code> will be used. If FALSE, the default, a normal density will be assumed and will be estimated using <code>dnorm</code> .
object, x	A <code>ps.cont</code> object.
...	For <code>ps.cont</code> , if <code>use.density</code> = TRUE, additional arguments to <code>density</code> , which is used to produce the density for the numerator of the weights. These include <code>bw</code> , <code>adjust</code> , <code>kernel</code> , and <code>n</code> . The default values are the defaults for <code>density</code> , except <code>n</code> , which is 10 times the number of units. For <code>summary.ps.cont</code> , additional arguments affecting the summary produced.

Details

`ps.cont` extends `ps` in **twang** to continuous treatments. It estimates weights from a series of trees and then outputs the weights that optimize a user-set criterion. The criterion employed involves the correlation between the treatment and each covariate. In a fully balanced sample, the treatment will have a correlation of 0 with covariates sufficient for removing confounding. Zhu, Coffman, & Ghosh (2015), who were the first to describe GBM for propensity score weighting with continuous treatments, recommend this procedure and provided R code to implement the methods they describe. `ps.cont` adapts their syntax to make it consistent with that of `ps` in **twang**. As in Zhu et al. (2015), when the Pearson correlation is requested, weighted biserial correlations will be computed for binary covariates.

The weights are estimated as the marginal density of the treatment divided by the conditional density of the treatment on the covariates for each unit. For the marginal density, a kernel density estimator can be implemented using the `density` function. For the conditional density, a Gaussian density is assumed. Note that with treatment with outlying values, extreme weights can be produced, so it is important to examine the weights and trim them if necessary.

It is recommended to use as many trees as possible, though this requires more computation time, especially with `use.optimize` set to 0. There is little difference between using Pearson and Spearman

correlations or between using the raw correlations and the Z-transformed correlations. Typically the only gbm-related options that should be changed are the interaction depth and number of trees.

Missing data is not allowed in the covariates because of the ambiguity in computing correlations with missing values.

`summary.ps.cont` compresses the information in the `desc` component of the `ps.cont` object into a short summary table describing the size of the dataset and the quality of the generalized propensity score weights, in a similar way to `summary.ps`.

`plot.ps.cont` and `boxplot.ps.cont` function almost identically to `plot.ps` and `boxplot.ps`. See the help pages there for more information. Note that for `plot.ps`, only options 1, 2, and 6 are available for the `plots` argument. When `use.optimize = 2`, option 1 is not available.

Value

Returns an object of class `ps` and `ps.cont`, a list containing

<code>gbm.obj</code>	The returned <code>gbm</code> object.
<code>treat</code>	The treatment variable.
<code>desc</code>	a list containing balance tables for each method selected in <code>stop.method</code> . Includes a component for the unweighted analysis names “unw”. Each <code>desc</code> component includes a list with the following components: <ul style="list-style-type: none"> ess The effective sample size n The number of subjects max.p.cor The largest absolute Pearson correlation across the covariates mean.p.cor The mean absolute Pearson correlation of the covariates rmse.p.cor The root mean squared Pearson correlation across the covariates max.s.cor The largest absolute Spearman correlation across the covariates mean.s.cor The mean absolute Spearman correlation of the covariates rmse.s.cor The root mean squared Spearman correlation across the covariates bal.tab a table summarizing the quality of the weights for yielding low treatment-covariate correlations. This table is best extracted using <code>bal.table</code>. n.trees The estimated optimal number of <code>gbm</code> iterations to optimize the loss function for the associated <code>stop.methods</code>
<code>ps</code>	a data frame containing the estimated generalized propensity scores. Each column is associated with one of the methods selected in <code>stop.methods</code> .
<code>w</code>	a data frame containing the propensity score weights. Each column is associated with one of the methods selected in <code>stop.methods</code> . If sampling weights are given then these are incorporated into the weights.
<code>estimand</code>	NULL
<code>datestamp</code>	Records the date of the analysis.
<code>parameters</code>	Saves the <code>ps.cont</code> call.
<code>alerts</code>	NULL
<code>iters</code>	A sequence of iterations used in the GBM fits used by <code>plot.ps.cont</code> .

balance	The balance summary for each tree examined, with a column for each stop.method. If optimize = 0, this will contain balance summaries for all trees. If optimize = 1, this will contain balance summaries for the subset of trees corresponding to iters. If optimize = 2, this will be NULL.
n. trees	Maximum number of trees considered in GBM fit.
data	Data as specified in the data argument.

The NULL entries exist so the output object is similar to that of ps in **twang**.

Author(s)

Noah Greifer

ps.cont is heavily adapted from the R code in Zhu, Coffman, & Ghosh (2015). In contrast with their code, ps.cont uses weighted Pearson and Spearman correlations rather than probability weighted bootstrapped correlations, allows for different degrees of optimization in searching for the best solution, and allows for the use of kernel density estimation for the generalized propensity score. ps.cont also takes inspiration from ps in **twang**.

References

Zhu, Y., Coffman, D. L., & Ghosh, D. (2015). A Boosting Algorithm for Estimating Generalized Propensity Scores with Continuous Treatments. *Journal of Causal Inference*, 3(1). doi: [10.1515/jci20140022](https://doi.org/10.1515/jci20140022)

See Also

[weightit](#) for its implementation using weightit syntax. [ps](#) and [mnps](#) for GBM with binary and multinomial treatments. [gbm](#) for the underlying machinery and explanation of the parameters.

Examples

```
# Examples take a long time
## Not run:
library("cobalt")
data("lalonge", package = "cobalt")

#Balancing covariates with respect to re75
psc.out <- ps.cont(re75 ~ age + educ + married +
  nodegree + race + re74, data = lalonge,
  stop.method = c("p.mean", "p.max"),
  use.optimize = 2)
summary(psc.out)
twang::bal.table(psc.out) #twang's bal.table

## End(Not run)
```

sbps

*Subgroup Balancing Propensity Score***Description**

Implements the subgroup balancing propensity score (SBPS), which is an algorithm that attempts to achieve balance in subgroups by sharing information from the overall sample and subgroups. (Dong, Zhang, Zeng, & Li, 2019; DZZL). Each subgroup can use either weights estimated using the whole sample, weights estimated using just that subgroup, or a combination of the two. The optimal combination is chosen as that which minimizes an imbalance criterion that includes subgroup as well as overall balance.

Usage

```
sbps(obj, obj2 = NULL,
      moderator = NULL,
      formula = NULL,
      data = NULL,
      smooth = FALSE,
      full.search)

## S3 method for class 'weightit.sbps'
print(x, ...)

## S3 method for class 'weightit.sbps'
summary(object, top = 5,
         ignore.s.weights = FALSE, ...)

## S3 method for class 'summary.weightit.sbps'
print(x, ...)
```

Arguments

obj	a <code>weightit</code> object containing weights estimated in the overall sample.
obj2	a <code>weightit</code> object containing weights estimated in the subgroups. Typically this has been estimated by including <code>by</code> in the call to <code>weightit()</code> . Either <code>obj2</code> or <code>moderator</code> must be specified.
moderator	optional; a string containing the name of the variable in <code>data</code> for which weighting is to be done within subgroups or a one-sided formula with the subgrouping variable on the right-hand side. This argument is analogous to the <code>by</code> argument in <code>weightit()</code> , and in fact it is passed on to <code>by</code> . Either <code>obj2</code> or <code>moderator</code> must be specified.
formula	an optional formula with the covariates for which balance is to be optimized. If not specified, the formula in <code>obj\$call</code> will be used.

<code>data</code>	an optional data set in the form of a data frame that contains the variables in formula or moderator.
<code>smooth</code>	logical; whether the smooth version of the SBPS should be used. This is only compatible with <code>weightit</code> methods that return a propensity score.
<code>full.search</code>	logical; when <code>smooth = FALSE</code> , whether every combination of subgroup and overall weights should be evaluated. If <code>FALSE</code> , a stochastic search as described in <code>DZZL</code> will be used instead. If <code>TRUE</code> , all 2^R combinations will be checked, where <code>R</code> is the number of subgroups, which can take a long time with many subgroups. If unspecified, will default to <code>TRUE</code> if <code>R <= 8</code> and <code>FALSE</code> otherwise.
<code>x</code>	a <code>weightit.sbps</code> or <code>summary.weightit.sbps</code> object; the output of a call to <code>sbps</code> or <code>summary.weightit.sbps</code> .
<code>object</code>	a <code>weightit.sbps</code> object; the output of a call to <code>sbps</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.
<code>...</code>	for <code>print</code> , arguments passed to <code>print</code> . Ignored otherwise.

Details

The SBPS relies on two sets of weights: one estimated in the overall sample and one estimated within each subgroup. The algorithm decides whether each subgroup should use the weights estimated in the overall sample or those estimated in the subgroup. There are 2^R permutations of overall and subgroup weights, where `R` is the number of subgroups. The optimal permutation is chosen as that which minimizes a balance criterion as described in `DZZL`. The balance criterion used here is, for binary and multinomial treatments, the sum of the squared standardized mean differences within subgroups and overall, which are computed using `col_w_smd` in **cobalt**, and for continuous treatments, the sum of the squared correlations between each covariate and treatment within subgroups and overall, which are computed using `col_w_corr` in **cobalt**.

The smooth version estimates weights that determine the relative contribution of the overall and subgroup propensity scores to a weighted average propensity score for each subgroup. If `P_O` are the propensity scores estimated in the overall sample and `P_S` are the propensity scores estimated in each subgroup, the smooth SBPS finds `R` coefficients `C` so that for each subgroup, the ultimate propensity score is $C * P_S + (1 - C) * P_O$, and weights are computed from this propensity score. The coefficients are estimated using `optim` with `method = "L-BFGS-B"`. When `C` is estimated to be 1 or 0 for each subgroup, the smooth SBPS coincides with the standard SBPS.

If `obj2` is not specified and `moderator` is, `sbps()` will attempt to refit the model specified in `obj` with the `moderator` in the `by` argument. This relies on the environment in which `obj` was created to be intact and can take some time if `obj` was hard to fit. It's safer to estimate `obj` and `obj2` (the latter simply by including the `moderator` in the `by` argument) and supply these to SBPS.

Value

A `weightit.sbps` object, which inherits from `weightit`. This contains all the information in `obj` with the weights, propensity scores, `call`, and possibly covariates updated from `sbps()`. In addition,

the `prop.subgroup` component contains the values of the coefficients `C` for the subgroups (which are either 0 or 1 for the standard SBPS), and the `moderator` component contains a `data.frame` with the moderator.

This object has its own summary methods and is compatible with **cobalt** functions. The `cluster` argument should be used with **cobalt** functions to accurately reflect the performance of the weights in balancing the subgroups.

Author(s)

Noah Greifer

References

Dong, J., Zhang, J. L., Zeng, S., & Li, F. (2019). Subgroup Balancing Propensity Score.

See Also

[weightit](#), [summary.weightit](#)

Examples

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

#Balancing covariates between treatment groups
(W1 <- weightit(treat ~ age + educ + married +
               nodegree + race + re74, data = lalonde,
               method = "ps", estimand = "ATT"))

(W2 <- weightit(treat ~ age + educ + married +
               nodegree + race + re74, data = lalonde,
               method = "ps", estimand = "ATT",
               by = "race"))

S <- sbps(W1, W2)
print(S)
summary(S)
bal.tab(S, cluster = "race")

#Could also have run
# sbps(W1, moderator = "race")

S_ <- sbps(W1, W2, smooth = TRUE)
print(S_)
summary(S_)
bal.tab(S_, cluster = "race")
```

summary.weightit *Print and Summarize Output*

Description

summary generates a summary of the weightit or weightitMSM object to evaluate the properties of the estimated weights. plot plots the distribution of the weights.

Usage

```
## S3 method for class 'weightit'
summary(object, top = 5,
        ignore.s.weights = FALSE, ...)

## S3 method for class 'summary.weightit'
print(x, ...)

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

## S3 method for class 'weightitMSM'
summary(object, top = 5,
        ignore.s.weights = FALSE, ...)

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

Arguments

object	a weightit or weightitMSM object; the output of a call to weightit or weightitMSM.
top	how many of the largest and smallest weights to display. Default is 5.
ignore.s.weights	whether or not to ignore sampling weights when computing the weight summary. If FALSE, the default, the estimated weights will be multiplied by the sampling weights (if any) before values are computed.
x	a summary.weightit or summary.weightitMSM object; the output of a call to summary.weightit or summary.weightitMSM.
...	for print, arguments passed to <code>print</code> . For plot, additional arguments passed to <code>hist</code> to determine the number of bins, though <code>geom_histogram</code> from ggplot2 is actually used to create the plot.

Value

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

<code>weight.range</code>	The range (minimum and maximum) weight for each treatment group.
<code>weight.top</code>	The units with the greatest weights in each treatment group; how many are included is determined by <code>top</code> .
<code>weight.ratio</code>	The ratio of the largest weight to the smallest weight in each treatment group and overall.
<code>coef.of.var</code>	The coefficient of variation (standard deviation divided by mean) of the weights in each treatment group and overall.
<code>effective.sample.size</code>	The effective sample size for each treatment group before and after weighting.

For longitudinal treatments (i.e., `weightitMSM` objects), a list of the above elements for each treatment period.

`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.

Author(s)

Noah Greifer

See Also

[weightit](#), [weightitMSM](#), [summary](#)

Examples

```
# See example at ?weightit or ?weightitMSM
```

<code>trim</code>	<i>Trim Large Weights</i>
-------------------	---------------------------

Description

Trims (i.e., truncates) large weights by setting all weights higher than that at a given quantile to the weight at the quantile. This can be useful in controlling extreme weights, which can reduce effective sample size by enlarging the variability of the weights.

Usage

```
## S3 method for class 'weightit'
trim(x, at = .99, lower = FALSE, ...)

## S3 method for class 'numeric'
trim(x, at = .99, lower = FALSE, treat = NULL, ...)
```

Arguments

x	A <code>weightit</code> object or a vector of weights.
at	numeric; either the quantile of the weights above which weights are to be trimmed. A single number between .5 and 1, or the number of weights to be trimmed (e.g., <code>at = 3</code> for the top 3 weights to be set to the 4th largest weight)
lower	logical; whether also to trim at the lower quantile (e.g., for <code>at = .9</code> , trimming at both .1 and .9, or for <code>at = 3</code> , trimming the top and bottom 3 weights).
treat	A vector of treatment status for each unit. This should always be included when <code>x</code> is numeric, but you can get away with leaving it out if the treatment is continuous or the estimand is the ATE for binary or multinomial treatments.
...	Not used.

Details

`trim` takes in a `weightit` object (the output of a call to `weightit` or `weightitMSM`) or a numeric vector of weights and trims them to the specified quantile. All weights above that quantile are set to the weight at that quantile. If `lower = TRUE`, all weights below 1 minus the quantile are to set the weight at 1 minus the quantile. In general, trimming weights decreases balance but also decreases the variability of the weights, improving precision at the potential expense of unbiasedness (Cole & Hernán, 2008). See Lee, Lessler, and Stuart (2011) and Thoemmes and Ong (2015) for discussions and simulation results of trimming weights at various quantiles.

When using `trim` on a numeric vector of weights, it is helpful to include the treatment vector as well. The helps determine the type of treatment and estimand, which are used to specify how trimming is performed. In particular, if the estimand is determined to be the ATT or ATC, the weights of the target (i.e., focal) group are ignored, since they should all be equal to 1. Otherwise, if the estimand is the ATE or the treatment is continuous, all weights are considered for trimming.

Value

If the input is a `weightit` object, the output will be a `weightit` object with the weights replaced by the trimmed weights, while will have an additional attribute, "`trim`", equal to the quantile of trimming.

If the input is a numeric vector of weights, the output will be a numeric vector of the trimmed weights, again with the aforementioned attribute.

Author(s)

Noah Greifer

References

- Cole, S. R., & Hernán, M. Á. (2008). Constructing Inverse Probability Weights for Marginal Structural Models. *American Journal of Epidemiology*, 168(6), 656–664.
- Lee, B. K., Lessler, J., & Stuart, E. A. (2011). Weight Trimming and Propensity Score Weighting. *PLoS ONE*, 6(3), e18174.
- Thoemmes, F., & Ong, A. D. (2016). A Primer on Inverse Probability of Treatment Weighting and Marginal Structural Models. *Emerging Adulthood*, 4(1), 40–59.

See Also

```
link{weightit}, link{weightitMSM}
```

Examples

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

(W <- weightit(treat ~ age + educ + married +
              nodegree + re74, data = lalonde,
              method = "ps", estimand = "ATT"))
summary(W)

#Trimming the top and bottom 5 weights
trim(W, at = 5, lower = TRUE)

#Trimming at 90th percentile
(W.trim <- trim(W, at = .9))

summary(W.trim)
#Note that only the control weights were trimmed

#Trimming a numeric vector of weights
weights <- cobalt::get.w(W)

all.equal(trim(weights, at = .9, treat = lalonde$treat),
          W.trim$weights)

#Using made up data and as.weightit()
treat <- rbinom(500, 1, .3)
weights <- rchisq(500, df = 2)
W <- as.weightit(weights= weights, treat = treat,
                 estimand = "ATE")

summary(W)
summary(trim(W, at = .95))
```

weightit

Generate Balancing Weights

Description

weightit allows for the easy generation of balancing weights using a variety of available methods for binary, continuous, and multinomial treatments. Many of these methods exist in other packages, which weightit calls; these packages must be installed to use the desired method. Also included are print and summary methods for examining the output.

Usage

```
weightit(formula,
          data = NULL,
          method = "ps",
          estimand = "ATE",
          stabilize = FALSE,
          focal = NULL,
          by = NULL,
          s.weights = NULL,
          ps = NULL,
          moments = 1,
          int = FALSE,
          verbose = FALSE,
          include.obj = FALSE,
          ...)

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

Arguments

formula	a formula with a treatment variable on the left hand side and the covariates to be balanced on the right hand side. 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.
method	a string of length 1 containing the name of the method that will be used to estimate weights. See Details below for allowable options. The default is "ps".
estimand	the desired estimand. For binary and multinomial treatments, can be "ATE", "ATT", "ATC", and, for some methods, "ATO" or "ATM". The default for both is "ATE". This argument is ignored for continuous treatments. See the individual pages for each method for more information on which estimands are allowed with each method and what literature to read to interpret these estimands.
stabilize	logical; whether or not to stabilize the weights. For the methods that involve estimating propensity scores, this involves multiplying each unit's weight by the sum of the weights in that unit's treatment group. For the "ebal" method, this involves using <code>ebalance.trim</code> to reduce the variance of the weights. Default is FALSE.
focal	when multinomial treatments are used and the "ATT" is requested, 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".
by	a string containing the name of the variable in data for which weighting is to be done within categories or a one-sided formula with the stratifying variable on the right-hand side. For example, if <code>by = "gender"</code> or <code>by = ~ gender</code> , weights

	will be generated separately within each level of the variable "gender". The argument used to be called <code>exact</code> , which will still work but with a message. Only one by variable is allowed.
<code>s.weights</code>	A vector of sampling weights or the name of a variable in data that contains sampling weights. These can also be matching weights if weighting is to be used on matched data.
<code>ps</code>	A vector of propensity scores or the name of a variable in data containing propensity scores. If not NULL, <code>method</code> is ignored, and the propensity scores will be used to create weights. <code>formula</code> must include the treatment variable in data, but the listed covariates will play no role in the weight estimation.
<code>moments</code>	numeric; for entropy balancing, empirical balancing calibration weights, optimization-based weights, the greatest moment of the covariate distribution to be balanced. For example, if <code>moments = 3</code> , for all non-categorical covariates, the mean, second moment (variance), and third moments (skew) of the covariates will be balanced. This argument is ignored for other methods; to balance powers of the covariates, appropriate functions must be entered in <code>formula</code> .
<code>int</code>	logical; for entropy balancing, empirical balancing calibration weights, and optimization-based weights, whether first-order interactions of the covariates are to be balanced (essentially balancing the covariances between covariates). This argument is ignored for other methods; to balance interactions between the variables, appropriate functions must be entered in <code>formula</code> .
<code>verbose</code>	whether to print additional information output by the fitting function.
<code>include.obj</code>	whether to include in the output any fit objects created in the process of estimating the weights. For example, with <code>method = "ps"</code> , the <code>glm</code> objects containing the propensity score model will be included. See Details for information on what object will be included if TRUE.
<code>...</code>	other arguments for functions called by <code>weightit</code> that control aspects of fitting that are not covered by the above arguments. See Details.
<code>x</code>	a <code>weightit</code> object; the output of a call to <code>weightit</code> .

Details

The primary purpose of `weightit` is as a dispatcher to other functions in other packages that perform the estimation of balancing weights. These functions are identified by a name, which is used in `method` to request them. Each method has some slight distinctions in how it is called, but in general, simply entering the method will cause `weightit` to generate the weights correctly using the function. To use each method, the package containing the function must be installed, or else an error will appear. Below are the methods allowed and links to pages containing more information about them, including additional arguments and outputs (e.g., when `include.obj = TRUE`) and how missing values are treated.

- ["ps"](#) - Propensity score weighting using generalized linear models.
- ["gbm"](#) - Propensity score weighting using generalized boosted modeling.
- ["cbps"](#) - Covariate Balancing Propensity Score weighting.
- ["npcbps"](#) - Non-parametric Covariate Balancing Propensity Score weighting.
- ["ebal"](#) - Entropy balancing.

- "ebcw" - Empirical balancing calibration weighting.
- "optweight" - Optimization-based weighting.
- "super" - Propensity score weighting using SuperLearner.
- "user-defined" - Weighting using a user-defined weighting function.

Value

A `weightit` 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>estimand</code>	The estimand requested.
<code>method</code>	The weight estimation method specified.
<code>ps</code>	The estimated or provided propensity scores. Estimated propensity scores are returned for binary treatments and only when method is "ps", "gbm", "cbps", or "super".
<code>s.weights</code>	The provided sampling weights.
<code>focal</code>	The focal variable if the ATT was requested with a multinomial treatment.
<code>by</code>	A <code>data.frame</code> containing the by variable when specified.
<code>obj</code>	When <code>include.obj = TRUE</code> , the fit object.

Author(s)

Noah Greifer

Examples

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

#Balancing covariates between treatment groups (binary)
(W1 <- weightit(treat ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "ps", estimand = "ATT"))
summary(W1)
bal.tab(W1)

#Balancing covariates with respect to race (multinomial)
(W2 <- weightit(race ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "ebal", estimand = "ATE"))
summary(W2)
bal.tab(W2)

#Balancing covariates with respect to re75 (continuous)
```

```
(W3 <- weightit(re75 ~ age + educ + married +
               nodegree + re74, data = lalonde,
               method = "cbps", over = FALSE))
summary(W3)
bal.tab(W3)
```

weightitMSM

Generate Balancing Weights

Description

weightitMSM allows for the easy generation of balancing weights for marginal structural models for time-varying treatments using a variety of available methods for binary, continuous, and multinomial treatments. Many of these methods exist in other packages, which `weightit` calls; these packages must be installed to use the desired method. Also included are `print` and `summary` methods for examining the output.

Currently only "wide" data sets, where each row corresponds to a unit's entire variable history, are supported. You can use `reshape` or other functions to transform your data into this format; see example below.

Usage

```
weightitMSM(formula.list,
             data = NULL,
             method = "ps",
             stabilize = FALSE,
             by = NULL,
             s.weights = NULL,
             num.formula = NULL,
             moments = 1,
             int = FALSE,
             verbose = FALSE,
             include.obj = FALSE,
             is.MSM.method,
             weightit.force = FALSE,
             ...)

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

Arguments

`formula.list` a list of formulas corresponding to each time point with the time-specific treatment variable on the left hand side and pre-treatment covariates to be balanced

on the right hand side. The formulas must be in temporal order, and must contain all covariates to be balanced at that time point (i.e., treatments and covariates featured in early formulas should appear in later ones). Interactions and functions of covariates are allowed.

data	an optional data set in the form of a data frame that contains the variables in the formulas in <code>formula.list</code> . This must be a wide data set with exactly one row per unit.
method	a string of length 1 containing the name of the method that will be used to estimate weights. See <code>weightit</code> for allowable options. The default is "ps", which estimates the weights using generalized linear models.
stabilize	logical; whether or not to stabilize the weights. Stabilizing the weights involves fitting a model predicting treatment at each time point from treatment status at prior time points. If TRUE, a saturated model will be fit, essentially using the observed treatment probabilities in the numerator (for binary and multinomial treatments). This may yield an error if some combinations are not observed. Default is FALSE. To manually specify stabilization model formulas, use <code>num.formula</code> .
num.formula	optional; a one-sided formula with the stabilization factors (other than the previous treatments) on the right hand side, which adds, for each time point, the stabilization factors to a model saturated with previous treatments. See Cole & Hernán (2008) for a discussion of how to specify this model; including stabilization factors can change the estimand without proper adjustment, and should be done with caution. Unless you know what you are doing, we recommend setting <code>stabilize = TRUE</code> and ignoring <code>num.formula</code> .
by	a string containing the name of the variable in <code>data</code> for which weighting is to be done within categories or a one-sided formula with the stratifying variable on the right-hand side. For example, if <code>by = "gender"</code> or <code>by = ~gender</code> , weights will be generated separately within each level of the variable "gender". The argument used to be called <code>exact</code> , which will still work but with a message. Only one <code>by</code> variable is allowed.
s.weights	a vector of sampling weights or the name of a variable in <code>data</code> that contains sampling weights. These are ignored for some methods.
moments	numeric; for some methods, the greatest moment of the covariate distribution to be balanced. For example, if <code>moments = 3</code> , for all non-categorical covariates, the mean, second moment (variance), and third moments (skew) of the covariates will be balanced. This argument is ignored for other methods; to balance powers of the covariates, appropriate functions must be entered in <code>formula</code> . See the specific methods help pages for information on whether they accept moments.
int	logical; for some methods, whether first-order interactions of the covariates are to be balanced (essentially balancing the covariances between covariates). This argument is ignored for other methods; to balance interactions between the variables, appropriate functions must be entered in <code>formula</code> . See the specific methods help pages for information on whether they accept <code>int</code> .
verbose	whether to print additional information output by the fitting function.
include.obj	whether to include in the output a list of the fit objects created in the process of estimating the weights at each time point. For example, with <code>method = "ps"</code> , a

	list of the <code>glm</code> objects containing the propensity score models at each time point will be included. See the help pages for each method for information on what object will be included if TRUE.
<code>is.MSM.method</code>	whether the method estimates weights for multiple time points all at once (TRUE) or by estimating weights at each time point and then multiplying them together (FALSE). This is only relevant for <code>method = "optweight"</code> , which estimates weights for longitudinal treatments all at once, and for user-specified functions.
<code>weightit.force</code>	several methods are not valid for estimating weights with longitudinal treatments, and will produce an error message if attempted. Set to TRUE to bypass this error message.
<code>...</code>	other arguments for functions called by <code>weightit</code> that control aspects of fitting that are not covered by the above arguments. See Details at weightit .
<code>x</code>	a <code>weightitMSM</code> object; the output of a call to <code>weightitMSM</code> .

Details

In general, `weightitMSM` works by separating the estimation of weights into separate procedures for each time period based on the formulas provided. For each formula, `weightitMSM` simply calls `weightit` to that formula, collects the weights for each time period, and multiplies them together to arrive at longitudinal balancing weights.

Each formula should contain all the covariates to be balanced on. For example, the formula corresponding to the second time period should contain all the baseline covariates, the treatment variable at the first time period, and the time-varying covariates that took on values after the first treatment and before the second. Currently, only wide data sets are supported, where each unit is represented by exactly one row that contains the covariate and treatment history encoded in separate variables.

The "gbm" method, which calls `ps` in **twang**, yields the same results to a call to `iptw` in **twang**. However, the `cbps` method, which calls `CBPS` in **CBPS**, will yield different results from `CBMSM` in **CBPS** because `CBMSM` takes a different approach to generating weights than simply estimating several time-specific models.

Value

A `weightitMSM` 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 time-varying treatment variables.
<code>covs.list</code>	A list of the covariates used in the fitting at each time point. Only includes the raw covariates, which may have been altered in the fitting process.
<code>data</code>	The <code>data.frame</code> originally entered to <code>weightitMSM</code> .
<code>estimand</code>	"ATE", currently the only estimand for MSMs with binary or multinomial treatments.
<code>method</code>	The weight estimation method specified.
<code>ps.list</code>	A list of the estimated propensity scores (if any) at each time point.
<code>s.weights</code>	The provided sampling weights.
<code>by</code>	A <code>data.frame</code> containing the <code>by</code> variable when specified.
<code>stabilization</code>	The stabilization factors, if any.

Author(s)

Noah Greifer

References

Cole, S. R., & Hernán, M. A. (2008). Constructing Inverse Probability Weights for Marginal Structural Models. *American Journal of Epidemiology*, 168(6), 656–664. doi: [10.1093/aje/kwn164](https://doi.org/10.1093/aje/kwn164)

See Also

[weightit](#) for information on the allowable methods

Examples

```
library("twang")
# library("cobalt")

data("iptwExWide", package = "twang")
(W <- weightitMSM(list(tx1 ~ age + gender + use0,
                      tx2 ~ tx1 + use1 + age + gender + use0,
                      tx3 ~ tx2 + use2 + tx1 + use1 + age + gender + use0),
                 data = iptwExWide,
                 method = "ps"))

summary(W)
# bal.tab(W)

##Going from long to wide data
data("iptwExLong", package = "twang")
wide_data <- reshape(iptwExLong$covariates, #long data
                    timevar = "time",      #time variable
                    v.names = c("use", "tx"), #time-varying
                    idvar = "ID",          #time-stable
                    direction = "wide",
                    sep = "")

(W2 <- weightitMSM(list(tx1 ~ age + gender + use1,
                       tx2 ~ tx1 + use2 + age + gender + use1,
                       tx3 ~ tx2 + use3 + tx1 + use2 + age +
                           gender + use1),
                  data = wide_data,
                  method = "ps"))

summary(W2)

all.equal(W$weights, W2$weights)
```

Index

`as.weightit`, 2
`as.weightitMSM` (`as.weightit`), 2
ATE, 11

`bal.table`, 32
bias-corrected generalized linear models, 20
`boxplot.ps`, 32
`boxplot.ps.cont` (`ps.cont`), 30
`brmultinom`, 20

CBMSM, 8, 16
CBPS, 7
`col_w_corr`, 35
`col_w_ks`, 13
`col_w_smd`, 13, 35

density, 20, 21, 23, 31
`dnorm`, 31

`ebalance`, 9, 10
`ebalance.trim`, 10

family, 19

`gbm`, 30, 32, 33
`gbm.cont` (`ps.cont`), 30
`gbm.fit`, 13, 14
`geom_histogram`, 37
`get_w_from_ps`, 4, 13
`glm`, 19, 41

`hist`, 37

`iptw`, 25

`listWrappers`, 23

`make_full_rank`, 6, 27
`method_cbps`, 7
`method_ebal`, 9

`method_ebcw`, 11
`method_gbm`, 12, 26
`method_npcbps`, 15
`method_optweight`, 17
`method_ps`, 4, 5, 19
`method_super`, 22
`method_twang`, 15, 25
`method_user`, 6, 7, 27
`mlogit`, 20
`mnp`, 20
`mnps`, 25, 33
`model.matrix`, 7

`npCBPS`, 16

`optim`, 35
`optimize`, 31
`optweight`, 17, 18

`plot.ps`, 32
`plot.ps.cont` (`ps.cont`), 30
`plot.summary.weightit`
(`summary.weightit`), 37
`polr`, 20
`print`, 35, 37
`print.summary.weightit`
(`summary.weightit`), 37
`print.summary.weightit.sbps` (`sbps`), 34
`print.summary.weightitMSM`
(`summary.weightit`), 37
`print.weightit` (`weightit`), 40
`print.weightit.sbps` (`sbps`), 34
`print.weightitMSM` (`weightitMSM`), 44
`ps`, 25, 30, 33
`ps.cont`, 25, 30

`reshape`, 44

SBPS (`sbps`), 34
`sbps`, 34
`summary`, 38

summary.ps, 32
summary.ps.cont (ps.cont), 30
summary.weightit, 36, 37
summary.weightit.sbps (sbps), 34
summary.weightitMSM (summary.weightit),
37
SuperLearner, 22, 23

trim, 14, 38

user-defined methods, 6

WeightIt (weightit), 40
weightit, 7, 9–13, 15–19, 22, 24–26, 28, 33,
36, 38, 40, 44–47
weightitMSM, 7, 9, 10, 13, 15–19, 22, 24–26,
28, 38, 44