Package 'WeightIt'

January 12, 2020

Type Package

Title Weighting for Covariate Balance in Observational Studies

Version 0.8.0

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>, 'CBPS' for covariate balancing propensity score weighting (Imai & Ratkovic, 2014) <doi:10.1111/rssb.12027>, 'ebal' for entropy balancing (Hainmueller, 2012) <doi:10.1093/pan/mpr025>, 'optweight' for optimization-based weights (Zubizarreta, 2015) <doi:10.1080/01621459.2015.1023805>, 'ATE' for empirical balancing calibration weighting (Chan, Yam, & Zhang, 2016) <doi:10.1111/rssb.12129>, and 'SuperLearner' for stacked machine learning-based propensity scores (Pirracchio, Petersen, & van der Laan, 2015) <doi: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 (>= 4.0.0), twang (>= 1.5), CBPS (>= 0.18), ebal (>= 0.1-6), ATE (>= 0.2.0), optweight (>= 0.2.4), SuperLearner (>= 2.0-25), mlogit (>= 0.3-0), MNP (>= 3.1-0), brglm2 (>= 0.5.1), survey, jtools, boot, MASS, gbm (>= 2.1.3), misaem, 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

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

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

Repository CRAN

Date/Publication 2020-01-12 06:00:25 UTC

R topics documented:

`as.weightit` *Create a* `weightit` *object manually*

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)
```

ESS *Compute effective sample size of weighted sample*

Description

Computes the effective sample size (ESS) of a weighted sample, which represents the size of an unweighted sample with approximately the same amount of precision as the weighted sample under consideration.

Usage

```
ESS(w)
```

Arguments

w	a vector of weights

Details

The ESS is calculated as `sum(w)^2/sum(w^2)`.

References

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

See Also

summary.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 = "ps", estimand = "ATE"))
summary(W1)
ESS(W1$weights[W1$treat == 0])
ESS(W1$weights[W1$treat == 1])
```

`get_w_from_ps` *Get weights from propensity scores corresponding to different estimands*

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,
              subclass = NULL,
              stabilize = FALSE)
```

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.
`subclass`	`numeric`; the number of subclasses to use when computing weights using marginal mean weighting through stratification. If `NULL`, standard inverse probability weights (and their extensions) will be computed; if a number greater than 1, subclasses will be formed and weights will be computed based on subclass membership. `estimand` must be ATE, ATT, or ATC if `subclass` is non-`NULL`. See Details.
`stabilize`	`logical`; whether to compute stabilized weights or not. This simply involves multiplying each unit's weight by the proportion of units in their treatment group. For saturated outcome models and in balance checking, this won't make a difference; otherwise, this can improve performance.

Details

`get_w_from_ps` applies the formula for computing weights from propensity scores for the desired estimand. See the References section 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 multi-category 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.

When `subclass` is not `NULL`, marginal mean weighting through stratification (MMWS) weights are computed. The implementation differs slightly from that described in Hong (2010, 2012). First, subclasses are formed by finding the quantiles of the propensity scores in the target group (for the ATE, all units; for the ATT or ATC, just the units in the focal group). Any subclasses lacking members of a treatment group will be filled in with them from neighboring subclasses so each subclass will always have at least one member of each treatment group. A new subclass-propensity score matrix is formed, where each unit's subclass-propensity score for each treatment value is computed as the proportion of units with that treatment value in the unit's subclass. For example, if a subclass had 10 treated units and 90 control units in it, the subclass-propensity score for being treated would be .1 and the subclass-propensity score for being control would be .9 for all units in the subclass. For multi-category treatments, the propensity scores for each treatment are stratified separately as described in Hong (2012); for binary treatments, only one set of propensity scores are stratified and the subclass-propensity scores for the other treatment are computed as the complement of the propensity scores for the stratified treatment. After the subclass-propensity scores have been computed, the standard propensity score weighting formulas are used to compute the unstabilized MMWS weights. To estimate MMWS weights equivalent to those described in Hong (2010, 2012), `stabilize` must be set to `TRUE`, but, as with standard propensity score weights, this is optional.

`get_w_from_ps` is not compatible with continuous treatments.

Value

A vector of weights.

Author(s)

Noah Greifer

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

- `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

- 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

- Marginal mean weighting through stratification (MMWS)

Hong, G. (2010). Marginal mean weighting through stratification: Adjustment for selection bias in multilevel data. Journal of Educational and Behavioral Statistics, 35(5), 499–531. doi: 10.3102/1076998609359785

Multinomial Treatments

- `estimand = "ATO"`

Li, F., & Li, F. (2019). Propensity score weighting for causal inference with multiple treatments. The Annals of Applied Statistics, 13(4), 2389–2415. doi: 10.1214/19AOAS1282

- `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

- 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

- Marginal mean weighting through stratification

Hong, G. (2012). Marginal mean weighting through stratification: A generalized method for evaluating multivalued and multiple treatments with nonexperimental data. Psychological Methods, 17(1), 44–60. doi: 10.1037/a0024918

See Also

`method_ps`

Examples

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

ps.fit <- glm(treat ~ age + educ + race + married +
                nodegree + re74 + re75, data = lalonde,
```

```
              family = binomial)
ps <- ps.fit$fitted.values

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

treatAB <- factor(ifelse(lalonde$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)

#Using MMWS
w4 <- get_w_from_ps(ps, treat = lalonde$treat,
                    estimand = "ATE", subclass = 20,
                    stabilize = TRUE)

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

gbm.fit <- gbm(T3 ~ age + educ + race + married +
                 nodegree + re74 + re75, data = lalonde,
               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` *Make a matrix full rank*

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

mat	a numeric matrix or data frame to be transformed. Typically this contains covariates. NAs are not allowed.
with.intercept	whether an intercept (i.e., a vector of 1s) should be added to mat 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.

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.

Continuous Treatments: For continuous treatments, the generalized propensity score and weights are estimated using `CBPS`.

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: In the presence of missing data, the following value(s) for `missing` are allowed:

`"ind"` **(default)** 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 `NA`s.

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.

Additional Outputs

obj When include.obj = TRUE, the CB(G)PS model fit. For binary treatments, multinomial treatments with estimand = "ATE" and four or fewer treatment levels, and continuous treatments, the output of the call to CBPS::CBPS. For multinomial treatments with estimand = "ATT" or with more than four treatment levels, a list of CBPS fit objects.

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

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.

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.

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: In the presence of missing data, the following value(s) for `missing` are allowed:

`"ind"` **(default)** 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 `NA`s.

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.

Additional Outputs

`obj` When `include.obj = TRUE`, the entropy balancing model fit. For binary treatments with `estimand = "ATT"`, the output of the call to `ebal::ebalance` or `ebal::ebalance.trim` when `stabilize = TRUE`. For binary treatments with `estimand = "ATE"` and multinomial treatments, a list of outputs of calls to `ebal::ebalance` or `ebal::ebalance.trim`.

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

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.

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.

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: In the presence of missing data, the following value(s) for `missing` are allowed:

`"ind"` **(default)** 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 `NA`s.

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.

Additional Outputs

obj When include.obj = TRUE, the empirical balancing callibration model fit. For binary treatments with estimand = "ATT", the output of the call to ATE::ATE. For binary treatments with estimand = "ATE" and multinomial treatments, a list of outputs of calls to ATE::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

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 using a balance-based or prediction-based criterion to optimize in choosing the value of a tuning parameter (the number of trees). This method mimics the functionality of functions in the **twang** package, but has improved performance and more flexible options. See Note for more details.

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-Smirnov 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. Weights can also be computed using marginal mean weighting through stratification for the ATE, ATT, and ATC. See `get_w_from_ps` for details.

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-Smirnov 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. Weights can also be computed using marginal mean weighting through stratification for the ATE, ATT, and ATC. See `get_w_from_ps` for details. The balance that is optimized is that between each non-focal treatment and the focal treatment for the ATT and ATC, between each treatment and the overall unweighted sample for the ATE, and between each treatment and the overall weighted sample for other estimands.

Continuous Treatments: For continuous treatments, this method estimates the generalized propensity score using `gbm.fit` and then optimizes balance using `col_w_corr` for treatment-covariate correlations from **cobalt**.

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: In the presence of missing data, the following value(s) for `missing` are allowed:

`"ind"` **(default)** First, for each variable with missingness, a new missingness indicator variable is created that 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 weight estimation then proceeds with this new formula and set of covariates using surrogate splitting as described below. The covariates output in the resulting `weightit` object will be the original covariates with the `NA`s.

`"surr"` Surrogate splitting is used to process `NA`s. No missingness indicators are created. Nodes are split using only the non-missing values of each variable. To generate predicted values for

each unit, a non-missing variable that operates similarly to the variable with missingness is used as a surrogate. Missing values are ignored when calculating balance statistics to choose the optimal tree.

Additional Arguments

The following additional arguments can be specified:

`stop.method` A string describing the criterion used to select the best weights. When optimizing for balance, this has two parts, a statistic to be computed 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-Smirnov 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"`.

In addition, to optimize the cross-validation error, `stop.method` can be set as `"cv{#}"`, where `{#}` is replaced by a number representing the number of cross-validation folds used (e.g., `"cv5"` for 5-fold cross-validation).

`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 occurs especially with continuous treatments. 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.

`start.tree` The tree at which to start balance checking. If you know the best balance isn't in the first 100 trees, for example, you can set `start.tree = 101` so that balance statistics are not computed on the first 100 trees. This can save some time since balance checking takes up the bulk of the run time for balance-based stopping methods, and is especially useful when running the same model adding more and more trees. The default is 1, i.e., to start from the very first tree in assessing balance.

`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. The lower this value is, the more trees one may have to include to reach the optimum.

`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`.

For continuous treatments only, the following arguments may be supplied:

`density` A function corresponding the conditional density of the treatment. The standardized residuals of the treatment model will be fed through this function to produce the numerator and denominator of the generalized propensity score weights. If blank, `dnorm` is used as recommended by Robins et al. (2000). This can also be supplied as a string containing the name of the function to be called. If the string contains underscores, the call will be split by the underscores and the latter splits will be supplied as arguments to the second argument and beyond. For example, if `density = "dt_2"` is specified, the density used will be that of a t-distribution with 2 degrees of freedom. Using a t-distribution can be useful when extreme outcome values are observed (Naimi et al., 2014). Ignored if `use.kernel = TRUE` (described below).

`use.kernel` If `TRUE`, uses kernel density estimation through the `density` function to estimate the numerator and denominator densities for the weights. If `FALSE` (the default), the argument to the `density` parameter is used instead.

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

Additional Outputs

`info` A list with two entries:

- `best.tree` The number of trees at the optimum. If this is close to `n.trees`, `weightit` should be rerun with a larger value for `n.trees`, and `start.tree` can be set to just below `best.tree`. See example.
- `tree.val` A data.frame with two columns: the first is the number of trees and the second is the value of the criterion corresponding to that tree. Running `plot` on this object will plot the criterion by the number of trees and is a good way to see patterns in the relationship between them and to determine if more trees are needed. See example.

`obj` When `include.obj = TRUE`, the `gbm` fit used to generate the predicted values.

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 because both use the same architecture found in `ps.cont`.

When standardized mean differences are used for the stopping method, standardized mean differences are computed for binary variables as well as for continuous variables, unlike in **cobalt**, which computes raw differences in proportion for binary variables. In addition, when `estimand = "ATE"`, `s.d.denom` is set to "all" in the call to `col_w_smd`, which mirrors the method in **twang** but differs from the default in balance checking with `bal.tab` in **cobalt**, which typically uses `s.d.denom = "pooled"` in this case.

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

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

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). doi: 10.1515/jci20140022

See Also

`weightit`, `weightitMSM`, `method_twang`

Examples

```
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)

## Not run:
#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)

#Using a t(3) density and illustrating the search for
#more trees.
W4a <- weightit(re75 ~ age + educ + married +
                  nodegree + re74, data = lalonde,
                method = "gbm", density = "dt_3",
                stop.method = "p.max",
                n.trees = 10000)

W4a$info$best.tree #10000; optimum hasn't been found
plot(W4a$info$tree.val) #decreasing at right edge

W4b <- weightit(re75 ~ age + educ + married +
                  nodegree + re74, data = lalonde,
                method = "gbm", density = "dt_3",
                stop.method = "p.max",
                start.tree = 10000,
                n.trees = 20000)

W4b$info$best.tree #13417; optimum has been found
plot(W4b$info$tree.val) #increasing at right edge

bal.tab(W4b)


## End(Not run)
```

`method_npcbps` *Nonparametric Covariate Balancing Propensity Score Weighting*

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.

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.

Continuous Treatments: For continuous treatments, this method estimates the weights using npCBPS. The weights are taken from the output of the npCBPS fit object.

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: In the presence of missing data, the following value(s) for `missing` are allowed:

`"ind"` **(default)** 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 npCBPS can be passed through `weightit` or `weightitMSM`.

All arguments take on the defaults of those in npCBPS.

Additional Outputs

`obj` When `include.obj = TRUE`, the nonparametric CB(G)PS model fit. The output of the call to `CBPS::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

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.

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.

Continuous Treatments: For binary treatments, this method estimates the weights using `optweight`. The weights are taken from the output of the `optweight` fit object.

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: In the presence of missing data, the following value(s) for `missing` are allowed:

`"ind"` **(default)** 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 `NA`s.

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`.

Additional Outputs

`info` A list with one entry:

`duals` A data frame of dual variables for each balance constraint.

`obj` When `include.obj = TRUE`, the output of the call to `optweight::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.

Seriously, just use `optweight`. The syntax is almost identical and it's compatible with **cobalt**, too.

References

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

See Also

`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 ordinal treatments, an ordinal regression model is used to estimate generalized propensity scores. 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). Weights can also be computed using marginal mean weighting through stratification for the ATE, ATT, and ATC. See `get_w_from_ps` for details.

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. Weights can also be computed using marginal mean weighting through stratification for the ATE, ATT, and ATC. See get_w_from_ps for details.

Continuous Treatments: For continuous treatments, the generalized propensity score is estimated using linear regression. The conditional density can be specified as normal or another distribution. In addition, kernel density estimation can be used instead of assuming a specific 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.

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: In the presence of missing data, the following value(s) for missing are allowed:

"ind" **(default)** 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.

"saem" For binary treatments with link = "logit", a stochastic approximation version of the EM algorithm (SAEM) is used via the **misaem** package. No additional covariates are created. See Jiang et al. (2019) for information on this method. In some cases, this is a suitable alternative to multiple imputation.

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.

For continuous treatments only, the following arguments may be supplied:

`density` A function corresponding the conditional density of the treatment. The standardized residuals of the treatment model will be fed through this function to produce the numerator and denominator of the generalized propensity score weights. If blank, `dnorm` is used as recommended by Robins et al. (2000). This can also be supplied as a string containing the name of the function to be called. If the string contains underscores, the call will be split by the underscores and the latter splits will be supplied as arguments to the second argument and beyond. For example, if `density = "dt_2"` is specified, the density used will be that of a t-distribution with 2 degrees of freedom. Using a t-distribution can be useful when extreme outcome values are observed (Naimi et al., 2014). Ignored if `use.kernel = TRUE` (described below).

`use.kernel` If `TRUE`, uses kernel density estimation through the `density` function to estimate the numerator and denominator densities for the weights. If `FALSE`, the argument to the `density` parameter is used instead.

`bw, adjust, kernel, n` If `use.kernel = TRUE`, the arguments to the `density` function. 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 passed to `...` in `glm`.

In the presence of missing data with `link = "logit"` and `missing = "saem"`, additional arguments are passed to `miss.saem` and `pred_saem`, except the `method` argument in `pred_saem` is replaced with `saem_method`.

For multi-category treatments with `link = "logit"` or `"probit"`, the default is to use multinomial logistic or probit regression using the **mlogit** package. To request that separate binary logistic or probit regressions are run instead, set `use.mlogit = FALSE`. This can be helpful when `mlogit` is slow or fails to converge.

Additional Outputs

`obj` When `include.obj = TRUE`, the (generalized) propensity score model fit. For binary treatments, the output of the call to `glm`. For binary treatments with missing data and `missing = "saem"`, the output of the call to `misaem::miss.saem`. For ordinal treatments, the output of the call to `polr`. For multinomial treatments with `link = "logit"` or `"probit"` and `use.mlogit = TRUE`, the output of the call to `mlogit::mlogit`. For multinomial treatments with `use.mlogit = FALSE`, a list of the `glm` fits. For multinomial treatments with `link = "br.logit"`, the output of the call to `brglm2::brmultinom`. For multinomial treatments with `link = "bayes.probit"`, the output of the call to `MNP::MNP`. For continuous treatments, the output of the call to `glm` for the predicted values in the denominator density.

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

- `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

- 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

- Marginal mean weighting through subclassification

Hong, G. (2010). Marginal mean weighting through stratification: Adjustment for selection bias in multilevel data. Journal of Educational and Behavioral Statistics, 35(5), 499–531. doi: 10.3102/1076998609359785

- Bias-reduced logistic regression

See references for the **brglm2** package.

- SAEM logistic regression for missing data

Jiang, W., Josse, J., & Lavielle, M. (2019). Logistic regression with missing covariates — Parameter estimation, model selection and prediction within a joint-modeling framework. Computational Statistics & Data Analysis, 106907. doi: 10.1016/j.csda.2019.106907

Multinomial Treatments

- `estimand = "ATO"`

Li, F., & Li, F. (2019). Propensity score weighting for causal inference with multiple treatments. The Annals of Applied Statistics, 13(4), 2389–2415. doi: 10.1214/19AOAS1282

- `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

- 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

- Marginal mean weighting through subclassification

Hong, G. (2012). Marginal mean weighting through stratification: A generalized method for evaluating multivalued and multiple treatments with nonexperimental data. Psychological Methods, 17(1), 44–60. doi: 10.1037/a0024918

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.

- Using non-normal conditional densities

Naimi, A. I., Moodie, E. E. M., Auger, N., & Kaufman, J. S. (2014). Constructing Inverse Probability Weights for Continuous Exposures: A Comparison of Methods. Epidemiology, 25(2), 292–299. doi: 10.1097/EDE.0000000000000053

See Also

weightit, weightitMSM, get_w_from_ps

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,

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). Weights can also be computed using marginal mean weighting through stratification for the ATE, ATT, and ATC. See `get_w_from_ps` for details.

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. Weights can also be computed using marginal mean weighting through stratification for the ATE, ATT, and ATC. See `get_w_from_ps` for details.

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.

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: In the presence of missing data, the following value(s) for `missing` are allowed:

`"ind"` **(default)** 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. 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 `NA`s.

Additional Arguments

`discrete` if `TRUE`, uses discrete SuperLearner, which simply selects the best performing method. Default `FALSE`, which finds the optimal combination of predictions for the libraries using `SL.method`.

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`.

For continuous treatments only, the following arguments may be supplied:

`density` A function corresponding the conditional density of the treatment. The standardized residuals of the treatment model will be fed through this function to produce the numerator and denominator of the generalized propensity score weights. If blank, `dnorm` is used as recommended by Robins et al. (2000). This can also be supplied as a string containing the name of the function to be called. If the string contains underscores, the call will be split by the underscores and the latter splits will be supplied as arguments to the second argument and beyond. For example, if `density = "dt_2"` is specified, the density used will be that of a t-distribution with 2 degrees of freedom. Using a t-distribution can be useful when extreme outcome values are observed (Naimi et al., 2014). Ignored if `use.kernel = TRUE` (described below).

`use.kernel` If `TRUE`, uses kernel density estimation through the `density` function to estimate the numerator and denominator densities for the weights. If `FALSE`, the argument to the `density` parameter is used instead.

`bw, adjust, kernel, n` If `use.kernel = TRUE`, the arguments to the `density` function. 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`, whether to plot the estimated density.

Additional Outputs

`info` For binary treatments, a list with two entries, `coef` and `cvRisk`. For multinomial treatments, a list of lists with these two entries, one for each treatment level:

`coef` The coefficients in the linear combination of the predictions from each method in `SL.library`. Higher values indicate that the corresponding method plays a larger role in determining the resulting predicted value, and values close to zero indicate that the method plays little role in determining the predicted value. When `discrete = TRUE`, these correspond to the coefficients that would have been estimated had `discrete` been `TRUE`.

`cvRisk` The cross-validation risk for each method in `SL.library`. Higher values indicate that the method has worse cross-validation accuracy.

`obj` When `include.obj = TRUE`, the SuperLearner fit(s) used to generate the predicted values. For binary and continuous treatments, the output of the call to `SuperLearner::SuperLearner`. For multinomial treatments, a list of outputs to calls to `SuperLearner::SuperLearner`.

Note

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

References

- Binary treatments

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

- Continuous treatments

Kreif, N., Grieve, R., Díaz, I., & Harrison, D. (2015). Evaluation of the Effect of a Continuous Treatment: A Machine Learning Approach with an Application to Treatment for Traumatic Brain Injury. Health Economics, 24(9), 1213–1228. doi: 10.1002/hec.3189

See method_ps for additional references.

See Also

weightit, weightitMSM, get_w_from_ps

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.stepAIC",
                               "SL.glm.interaction")))
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.stepAIC",
                               "SL.glm.interaction")))
summary(W2)
bal.tab(W2)

#Balancing covariates with respect to re75 (continuous)
#assuming t(8) conditional density for treatment
(W3 <- weightit(re75 ~ age + educ + married +
                  nodegree + re74, data = lalonde,
                method = "super", density = "dt_8",
                SL.library = c("SL.glm", "SL.ridge",
                               "SL.glm.interaction")))
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**.

See the Note section at method_gbm for a discussion of how this method differs from method = "gbm". In general, using method = "gbm" is preferred because it is faster and more flexible. method = "twang" will likely become defunct in future versions.

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.

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.

Continuous Treatments: For continuous treatments, the generalized propensity score is estimated using ps.cont.

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: In the presence of missing data, the following value(s) for missing are allowed:

"ind" **(default)** 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 weight estimation then proceeds with this new formula and set of covariates using surrogate splitting as described below. The covariates output in the resulting weightit object will be the original covariates with the NAs.

"surr" Surrogate splitting is used to process NAs. No missingness indicators are created. Nodes are split using only the non-missing values of each variable. To generate predicted values for each unit, a non-missing variable that operates similarly to the variable with missingness is used as a surrogate. Missing values are ignored when calculating balance statistics to choose the optimal tree.

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.

Additional Outputs

`obj` When `include.obj = TRUE`, the `twang` model fit. For binary treatments, the output of the call to `twang::ps`. For multinomial treatments, a list of `ps` fit objects. For continuous treatments, the output of the call to `ps.cont`.

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

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

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("lalonde", package = "cobalt")

#Balancing covariates between treatment groups (binary)
(W1 <- weightit(treat ~ age + educ + married +
                  nodegree + re74, data = lalonde,
                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 = lalonde,
                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 it's 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("chadhazlett/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 optimize will be used to find the tree with the best balance within the tree interval chosen. If 2, optimize 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 density to estimate the numerator of the weights. If TRUE, density will be used. If FALSE, the default, a normal density will be assumed and will be estimated using dnorm.

object, x — A ps.cont object.

... — For ps.cont, if use.density = TRUE, additional arguments to density, which is used to produce the density for the numerator of the weights. These include bw, adjust, kernel, and n. The default values are the defaults for density, except n, which is 10 times the number of units.

For summary.ps.cont, 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

gbm.obj	The returned gbm object.
treat	The treatment variable.
desc	a list containing balance tables for each method selected in stop.method. Includes a component for the unweighted analysis names "unw". Each desc component includes a list with the following components: **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 bal.table. **n.trees** The estimated optimal number of gbm iterations to optimize the loss function for the associated stop.methods
ps	a data frame containing the estimated generalized propensity scores. Each column is associated with one of the methods selected in stop.methods.
w	a data frame containing the propensity score weights. Each column is associated with one of the methods selected in stop.methods. If sampling weights are given then these are incorporated into the weights.
estimand	NULL
datestamp	Records the date of the analysis.
parameters	Saves the ps.cont call.
alerts	NULL
iters	A sequence of iterations used in the GBM fits used by plot.ps.cont.
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

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("lalonde", package = "cobalt")

#Balancing covariates with respect to re75
psc.out <- ps.cont(re75 ~ age + educ + married +
                nodegree + race + re74, data = lalonde,
                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 weightit object containing weights estimated in the overall sample.
obj2	a weightit object containing weights estimated in the subgroups. Typically this has been estimated by including by in the call to weightit(). Either obj2 or moderator must be specified.
moderator	optional; a string containing the name of the variable in data 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 by argument in weightit(), and in fact it is passed on to by. Either obj2 or moderator must be specified.
formula	an optional formula with the covariates for which balance is to be optimized. If not specified, the formula in obj$call will be used.
data	an optional data set in the form of a data frame that contains the variables in formula or moderator.
smooth	logical; whether the smooth version of the SBPS should be used. This is only compatible with weightit methods that return a propensity score.
full.search	logical; when smooth = FALSE, whether every combination of subgroup and overall weights should be evaluated. If FALSE, a stochastic search as described in DZZL will be used instead. If TRUE, all 2^R combinations will be checked, where R is the number of subgroups, which can take a long time with many subgroups. If unspecified, will default to TRUE if R <= 8 and FALSE otherwise.
x	a weightit.sbps or summary.weightit.sbps object; the output of a call to sbps or summary.weightit.sbps.
object	a weightit.sbps object; the output of a call to sbps.
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.

...
: for print, arguments passed to print. 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 within races
(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 `print`. For `plot`, additional arguments passed to `hist` to determine the number of bins, though `geom_histogram` 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:

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

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
```

---

trim *Trim Large Weights*

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 weightit 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., at = 3 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 at = .9, trimming at both .1 and .9, or for at = 3, trimming the top and bottom 3 weights).
treat	A vector of treatment status for each unit. This should always be included when x 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 multi-category 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,
         subclass = NULL,
         missing = NULL,
```

```
          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" for propensity score weighting.
estimand	the desired estimand. For binary and multi-category 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 proportion of units in their treatment group. For the "ebal" method, this involves using ebalance.trim to reduce the variance of the weights. Default is FALSE.
focal	when multi-category 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 containingg 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 by = "gender" or by = ~ gender, weights will be generated separately within each level of the variable "gender". The argument used to be called exact, which will still work but with a message. Only one by variable is allowed; to stratify by multiply variables simultaneously, create a new variable that is a full cross of those variables using interaction.
s.weights	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.
ps	A vector of propensity scores or the name of a variable in data containing propensity scores. If not NULL, method is ignored, and the propensity scores will be used to create weights. formula must include the treatment variable in data, but the listed covariates will play no role in the weight estimation. Using

ps is similar to calling get_w_from_ps directly, but produces a full weightit object rather than just producing weights.

moments — numeric; for some methods, the greatest power of each covariate to be balanced. For example, if moments = 3, for each non-categorical covariate, the covariate, its square, and its cube will be balanced. This argument is ignored for other methods; to balance powers of the covariates, appropriate functions must be entered in formula. 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. This argument is ignored for other methods; to balance interactions between the variables, appropriate functions must be entered in formula. See the specific methods help pages for information on whether they accept int.

subclass — numeric; the number of subclasses to use for computing weights using marginal mean weighting with subclasses (MMWS). If NULL, standard inverse probability weights (and their extensions) will be computed; if a number greater than 1, subclasses will be formed and weights will be computed based on subclass membership. Attempting to set a non-NULL value for methods that don't compute a propensity score will result in an error; see each method's help page for information on whether MMWS weights are compatible with the method. See get_w_from_ps for details and references.

missing — character; how missing data should be handled. The options and defaults depend on the method used. Ignored if no missing data is present. It should be noted that multiple imputation outperforms all available missingness methods available in **weightit** and should probably be used instead. See the **MatchThem** package for the use of weightit with multiply imputed data.

verbose — whether to print additional information output by the fitting function.

include.obj — whether to include in the output any fit objects created in the process of estimating the weights. For example, with method = "ps", the glm objects containing the propensity score model will be included. See Details for information on what object will be included if TRUE.

... — other arguments for functions called by weightit that control aspects of fitting that are not covered by the above arguments. See Details.

x — a weightit object; the output of a call to weightit.

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:

weights	The estimated weights, one for each unit.
treat	The values of the treatment variable.
covs	The covariates used in the fitting. Only includes the raw covariates, which may have been altered in the fitting process.
estimand	The estimand requested.
method	The weight estimation method specified.
ps	The estimated or provided propensity scores. Estimated propensity scores are returned for binary treatments and only when method is "ps", "gbm", "cbps", or "super".
s.weights	The provided sampling weights.
focal	The focal variable if the ATT was requested with a multi-category treatment.
by	A data.frame containing the by variable when specified.
obj	When include.obj = TRUE, the fit object.

Author(s)

Noah Greifer

See Also

weightitMSM for estimating weights with sequential (i.e., longitudinal) treatments for use in estimating marginal strcutural models (MSMs).

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 (multi-category)
(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,
            missing = NULL,
            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 formula.list. 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 weightit 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 num.formula.
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 stabilize = TRUE and ignoring num.formula.
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 by = "gender" or by = ~ gender, weights will be generated separately within each level of the variable "gender". The argument used to be called exact, which will still work but with a message. Only one by variable is allowed.
s.weights	a vector of sampling weights or the name of a variable in data that contains sampling weights. These are ignored for some methods.
moments	numeric; for some methods, the greatest power of each covariate to be balanced. For example, if moments = 3, for each non-categorical covariate, the covariate, its square, and its cube will be balanced. This argument is ignored for other methods; to balance powers of the covariates, appropriate functions must be entered in formula. 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. This argument is ignored for other methods; to balance interactions between the variables, appropriate functions must be entered in formula. See the specific methods help pages for information on whether they accept int.
missing	character; how missing data should be handled. The options and defaults depend on the method used. Ignored if no missing data is present. It should be noted that multiple imputation outperforms all available missingness methods available in **weightit** and should probably be used instead. See the **MatchThem** package for the use of weightit with multiply imputed data.
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 method = "ps", a list of the glm 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.
is.MSM.method	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 method = "optweight"), which estimates weights for longitudinal treatments all at once, and for user-specified functions.
weightit.force	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.
...	other arguments for functions called by weightit that control aspects of fitting that are not covered by the above arguments. See Details at weightit.
x	a weightitMSM object; the output of a call to weightitMSM.

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 "twang" 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:

weights	The estimated weights, one for each unit.

treat.list	A list of the values of the time-varying treatment variables.
covs.list	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.
data	The data.frame originally entered to weightitMSM.
estimand	"ATE", currently the only estimand for MSMs with binary or multinomial treatments.
method	The weight estimation method specified.
ps.list	A list of the estimated propensity scores (if any) at each time point.
s.weights	The provided sampling weights.
by	A data.frame containing the by variable when specified.
stabilization	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

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