Package 'WeightIt'

January 16, 2019

Type Package

Title Weighting for Covariate Balance in Observational Studies

Version 0.5.1

Author Noah Greifer [aut, cre]

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

Description Generates weights to form equivalent groups in observational studies by easing and extending the functionality of the R packages 'twang' (Ridgeway et al., 2017) <https://CRAN.R-project.org/package=twang> for generalized boosted modeling, 'CBPS' (Fong, Ratkovic, & Imai, 2018) <https://CRAN.R-project.org/package=CBPS> for covariate balancing propensity score weighting, 'ebal' (Hainmueller, 2014) <https://CRAN.R-project.org/package=ebal> for entropy balancing, 'optweight' (Greifer, 2018) <https://CRAN.R-project.org/package=optweight> for optimization-based weights, 'ATE' (Haris & Chan, 2015) <https://CRAN.R-project.org/package=ATE> for empirical balancing calibration weighting, and SuperLearner (Polley, LeDell, Kenney, & van der Laan, 2018) <https://CRAN.R-project.org/package=SuperLearner>. Also allows for assessment of weights and checking of covariate balance by interfacing directly with 'cobalt' (Greifer, 2018) <https://CRAN.R-project.org/package=cobalt>.

Depends R (>= 3.3.0)

Imports cobalt (>= 3.6.1)

Suggests twang (>= 1.5), CBPS (>= 0.18), ebal (>= 0.1-6), ATE (>= 0.2.0), optweight (>= 0.2.0), SuperLearner, class, gam, mlogit (>= 0.3-0), MNP (>= 3.1-0), survey, jtools, boot, wCorr, gbm (>= 2.1.3), knitr, rmarkdown

License GPL (>= 2)

Encoding UTF-8

LazyData true

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

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

VignetteBuilder knitr

NeedsCompilation no

R topics documented:

---

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.

with.intercept   whether an itnercept (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

> `link{method_user}`, `link{model.matrix}`

Examples

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

make.full.rank(mat) #leaves c2 and c4

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

---

method_cbps	*Covariate Balancing Propensity Score Weighting*

Description

> This page explains the details of estimating weights from covariate balancing propensity scores by setting `method = "cbps"` in the call to `weightit` or `weightitMSM`. This method can be used with binary, multinomial, and continuous treatments.
>
> In general, this method relies on estimating propensity scores using generalized method of moments and then converting those propensity scores into weights using a formula that depends on the desired estimand. This method relies on `CBPS` from the **CBPS** package.

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

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

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

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

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

Additional Arguments

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

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

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

`standardize` is ignored.

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

Note

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

References

Binary treatments

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

Multinomial Treatments

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

Continuous treatments

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

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)

#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)

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

---

method_ebal   *Entropy Balancing*

Description

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

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

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

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

Continuous Treatments: Continuous treatments are not supported.

Longitudinal Treatments: For longitudinal treatments, the weights are the product of the weights estimated at each time point. This method is not guaranteed to yield exact balance at each time point.

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

Additional Arguments

All argument to ebalance can be passed through weightit or weightitMSM, with the following exceptions:

base.weight is ignored because sampling weights are passed using s.weights.

All arguments take on the defaults of those in ebalance.

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

References

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

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 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 a function of the weights subject to exact moment balancing constraints. This method relies on ATE from the **ATE** package.

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

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

Continuous Treatments: Continuous treatments are not supported.

Longitudinal Treatments: For longitudinal treatments, the weights are the product of the weights estimated at each time point. This method is not guaranteed to yield exact balance at each time point.

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

Additional Arguments

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

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

All arguments take on the defaults of those in ATE.

References

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

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 = "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)
```

---

method_gbm   *Propensity Score Weighting Using Generalized Boosted Models*

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. 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**. The algorithm involves choosing a balance criterion to optimize so that balance, rather than prediction, is prioritized.

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

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

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

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

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

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 "s.mean.z" 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 improves performance.

Note

The weights may be very slightly different from those estimated by a call to ps, but this difference is inconsequential.

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

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 = "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 = "gbm", 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 = "gbm", use.kernel = TRUE,
                stop.method = "p.max"))
summary(W3)
bal.tab(W3)
```

---

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. When include.obj = TRUE, the returned object is the npCBPS fit.

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

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

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

> **Sampling Weights:** Sampling weights are not supported.

Additional Arguments

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

All arguments take on the defaults of those in npCBPS.

References

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

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 = "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)
```

---

method_optweight   *Optimization-Based Weighting*

Description

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

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

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

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

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

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

Longitudinal Treatments: For longitudinal treatments, optweight estimates weights that simultaneously satisfy balance constraints at all time points, so only one model is fit to obtain the weights. Using method = "optweight" in weightitMSM cause is.MSM.method to be set to TRUE by default. Setting it to FALSE will run one model for each time point and multiply the weights together, a method that is not recommended.

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

Additional Arguments

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

targets cannot be used and is ignored.

All arguments take on the defaults of those in optweight.

Note

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

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

Seriously, just use optweight.

References

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

See Also

weightit, weightitMSM

Examples

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

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

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

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

---

method_ps   *Propensity Score Weighting Using Generalized Linear Models*

Description

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

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

Binary Treatments: For binary treatments, this method estimates the propensity scores using `glm`. An additional argument is link, which uses the same options as link in `family`. The default link is "logit", but others, including "probit", are allowed. The following estimands are allowed: ATE, ATT, ATC, ATO, and ATM. The weights for the ATE, ATT, and ATC are computed from the estimated propensity scores using the standard formulas, the weights for the ATO are computed as in Li & Li (2018), and the weights for the ATM (i.e., average treatment effect in the equivalent sample "pair-matched" with calipers) are computed as in Yoshida et al (2017). When include.obj = TRUE, the returned object is the glm fit.

Multinomial Treatments: For multinomial treatments, the propensity scores are estimated using multinomial regression from one of two functions depending on the requested link: for logit ("logit") and probit ("probit") links, mlogit from the **mlogit** package is used, and for the Bayesian probit ("bayes.probit") link, mnp from the **MNP** package is used. If mlogit in not installed, a series of binomial regressions using glm will be run instead, with estimated propensities normalized to sum to 1. These are the only three links allowed for multinomial treatments at this time. (These methods can fail to converge, yielding errors that may seem foreign.) The following estimands are allowed: ATE, ATT, ATO, and ATM. The weights for each estimand are computed using the standard formulas or those mentioned above. When include.obj = TRUE, the returned object is the fit object from mlogit or mnp or the list of glm fit objects if binomial models are used.

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

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

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

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`. For multinomial treatments, `link` can be "logit", "probit", or "bayes.probit". For continuous treatments, `link` can be any of those allowed by `gaussian`.

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

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

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

References

Binary treatments

- `estimand = "ATO"`

Li, F., Morgan, K. L., & Zaslavsky, A. M. (2016). Balancing Covariates via Propensity Score Weighting. Journal of the American Statistical Association, 0(ja), 0–0.

- `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](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](10.1080/00273171.2011.568786)

Multinomial Treatments

- `estimand = "ATO"`

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

- `estimand = "ATM"`

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

Continuous treatments

method = ″ps″

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

See Also

weightit, weightitMSM

Examples

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

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

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

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

---

method_super   *Propensity Score Weighting Using SuperLearner*

Description

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

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

Binary Treatments: For binary treatments, this method estimates the propensity scores using [SuperLearner](#) in the **SuperLearner** package. The following estimands are allowed: ATE, ATT, ATC, ATO, and ATM. The weights for the ATE, ATT, and ATC are computed from the estimated propensity scores using the standard formulas, the weights for the ATO are computed as in Li & Li (2018), and the weights for the ATM (i.e., average treatment effect in the equivalent sample "pair-matched" with calipers) are computed as in Yoshida et al (2017). When `include.obj = TRUE`, the returned object is the SuperLearner fit.

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

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

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

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

Additional Arguments

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

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

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

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

The following additional arguments can be specified:

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

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

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

References

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

See Also

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

Examples

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

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

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

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

method_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
  if (estimand == "ATT") w <- ifelse(treat == focal, 1, ps/(1-ps))
  else if (estimand == "ATE") w <- ifelse(treat == 1, 1/ps, 1/(1-ps))

  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)

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)
```

---

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).

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](gbm).

shrinkage   The shrinkage passed on to [gbm](gbm).

bag.fraction   The bag.fraction passed on to [gbm](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, p.mean.z, p.rms.z, s.max, s.mean, s.rms, s.mean.z, and s.rms.z. p refers to the Pearson correlation and s refers to the Spearman correlation, both implemented in the **wCorr** package. 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). The correlations can first be Fisher's Z-transformed (z) or not.

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 25 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.

`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](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](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 25 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

```
## 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)
```

---

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.

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 '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`.
...	arguments passed to [print](#).

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.

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

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 contin-
                 uous 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 Struc-
tural 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)
```

---

weightit   *Generate Balancing Weights*

Description

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

Usage

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

```
          include.obj = FALSE,
          ...)

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

Arguments

formula	a formula with a treatment variable on the left hand side and the covariates to be balanced on the right hand side. See [glm](#) for more details. Interactions and functions of covariates are allowed.
data	an optional data set in the form of a data frame that contains the variables in formula.
method	a string of length 1 containing the name of the method that will be used to estimate weights. See Details below for allowable options. The default is ″ps″.
estimand	the desired estimand. For binary treatments, can be "ATE", "ATT", "ATC", and, for some methods, "ATO" or "ATM". For multinomial treatments, can be "ATE" or "ATT". The default for both is "ATE". This argument is ignored for continuous treatments. See Details for more information.
stabilize	logical; whether or not to stabilize the weights. For the methods that involve estimating propensity scores, this involves multiplying each unit's weight by the sum of the weights in that unit's treatment group. For the ″ebal″ method, this involves using ebalance.trim to reduce the variance of the weights. Default is FALSE.
focal	when multinomial treatments are used and the "ATT" is requested, which group to consider the "treated" or focal group. This group will not be weighted, and the other groups will be weighted to be more like the focal group.
by	a vector or the names of variables in data for which weighting is to be done within categories. For example, if 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.
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.
moments	numeric; for entropy balancing, empirical balancing calibration weights, optimization-based weights, the greatest moment of the covariate distribution to be balanced. For example, if moments = 3, for all non-categorical covariates, the mean, second moment (variance), and third moments (skew) of the covariates will be balanced. This argument is ignored for other methods; to balance powers of the covariates, appropriate functions must be entered in formula.

int	logical; for entropy balancing, empirical balancing calibration weights, and optimization-based weights, whether first-order interactions of the covariates are to be balanced (essentially balancing the covariances between covariates). This argument is ignored for other methods; to balance interactions between the variables, appropriate functions must be entered in formula.
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).

- "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.

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 multinomial treatment.
obj	When include.obj = TRUE, the fit object.

Author(s)

Noah Greifer

Examples

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

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

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

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

---

weightitMSM	*Generate Balancing Weights*

Description

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

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

Usage

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

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

Arguments

formula.list   a list of formulas corresponding to each time point with the time-specific treat-
                ment 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 con-
                tain all covariates to be balanced at that time point (i.e., treatments and covari-
                ates 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 in-
                volves 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 us-
                ing the observed treatment probabilities in the numerator (for binary and multi-
                nomial treatments). This may yield an error if some combinations are not ob-
                served. 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 pre-
                vious 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 stabi-
                lization 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 vector or the names of variables in data for which weighting is to be done within categories. For example, if 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.

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 moment of the covariate distribution to be balanced. For example, if moments = 3, for all non-categorical covariates, the mean, second moment (variance), and third moments (skew) of the covariates will be balanced. This argument is ignored for other methods; to balance powers of the covariates, appropriate functions must be entered in 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 (essentially balancing the covariances between covariates). This argument is ignored for other methods; to balance interactions between the variables, appropriate functions must be entered in formula. See the specific methods help pages for information on whether they accept int.

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

Value

A weightitMSM object with the following elements:

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.
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")
(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")
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(get.w(W), get.w(W2))
```

Index