Package 'WeightIt'

June 25, 2018

Type Package

Title Weighting for Covariate Balance in Observational Studies

Version 0.4.0

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, 'sbw' (Zubizarreta, 2015) <doi:10.1080/01621459.2015.1023805>, and 'ATE' (Haris & Chan, 2015) <https://CRAN.R-project.org/package=ATE> for empirical balancing calibration weighting. 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 (>= 3.3.0)

Suggests twang (>= 1.5), CBPS (>= 0.18), ebal (>= 0.1-6), ATE (>= 0.2.0), 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

Date 2018-06-25

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

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

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2018-06-25 04:43:04 UTC

R topics documented:

`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.
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, 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-tranformed (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` 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 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 and allows for different degrees of optimization in searching for the best solution. 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 treatment 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 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)
```

`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,
         exact = NULL,
         s.weights = NULL,
         ps = NULL,
         moments = 1,
         int = FALSE,
         verbose = 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.
exact	a vector or the names of variables in data for which weighting is to be done within catgories. For example, if exact = "gender", weights will be generated seperately within each level of the variable "gender".
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 callibration weights, and stable balancing 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 callibration weights, and stable balancing weights, whether first-order interactons 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.
`...`	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 some information about each.

`"ps"` **Propensity score weighting using GLM.** 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 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, Morgan, & Zaslavsky (2016), and the weights for the ATM (i.e., average treatment effect in the equivalent sample "pair-matched" with calipers) are computed as in Li & Greene (2013).

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 binary 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.)

For continuous treatments, the generalized propensity score is estimated using linear regression with a normal density, but other families and links are allowed, such as poisson for count treatments, using the `family` and `link` arguments. 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.

For all treatment types except multinomial treatments with a Bayesian probit link, sampling weights are supported, but a warning message from `glm()` may appear.

`"gbm"` **Propensity score weighting using generalized boosted modeling.** This method, which can also be requested as `"gbr"` or `"twang"`, uses functions from the **twang** package to perform generalized boosted modeling to estimate propensity scores that yield balance on the requested covariates. For binary treatments, `ps()` is used, and the ATE, ATT, and ATC can be requested. For multinomial treatments, `mnps()` is used, and the ATE or ATT can be requested. For both, the `weightit()` argument `s.weights` corresponds to the `ps()` and `mnps()` argument `sampw`. The `weightit()` argument `focal` corresponds to the `mnps()` argument `treatATT`. For both,

a single stop method must be supplied to `stop.method`; only one can be entered at a time. The other arguments to `ps()` and `mnps` can be specified in the call to `weightit()`. See `ps` and `mnps` for details.

For continuous treatments, the generalized propensity score is estimated using `ps.cont()`, which exists in `WeightIt`. Balance is optimized by a function of the correlation between the treatment and covariates in the weighted sample. See `ps.cont` for details.

`"cbps"` **Covariate Balancing Propensity Score weighting.** This method uses the `CBPS()` function from the **CBPS** package to estimate propensity scores and weights. It works with binary, multinomial, and continuous treatments. For binary treatments, the ATE, ATT, and ATC can be requested. For multinomial treatments, only the ATE can be requested. The `weightit()` argument `s.weights` corresponds to the `CBPS()` argument `sampling.weights`. `CBPS()` can fit either an over-identified model or a model that only contains covariate balancing conditions; this option is typically specified with the `method` argument to `CBPS()`, but because this argument is already used in `weightit()`, a new argument, `over`, can be specified. `over = FALSE` in `weightit()` is equivalent to `method = "exact"` in `CBPS()`. The other arguments to `CBPS()` can be specified in the call to `weightit()`. See `CBPS` for details.

`"npcbps"` **Non-parametric Covariate Balancing Propensity Score weighting.** This method uses the `npCBPS()` function from the **CBPS** package to estimate weights. It works with binary, multinomial, and continuous treatments. For binary and multinomial treatments, only the ATE can be requested. Sampling weights are not supported. The other arguments to `npCBPS()` can be specified in the call to `weightit()`. See `npCBPS` for details.

`"ebal"` **Entropy balancing.** This method uses the `ebalance()` function from the **ebal** package to estimate weights. It works with binary and multinomial treatments. For binary treatments, the ATE, ATT, and ATC can be requested. For multinomial treatments, the ATE and ATT can be requested. If the ATT is requetsed with a mutlinomial treatment, one treatment level must be entered to `focal` to serve as the "treated". Sampling weights are supported and are automatically entered into `base.weight` in `ebal()`. When `stabilize = TRUE`, `ebalance.trim()` is used to trim and reduce the variance of the weights. The other arguments to `ebalance()` can be specified in the call to `weightit()`. See `ebalance` for details.

`"sbw"` **Stable balancing weights.** This method uses the `sbw()` function from the **sbw** package to estimate weights. For binary treatments, the ATE, ATT, and ATC can be requested. For multinomial treatments, the ATE and ATT can be requested. If the ATT is requested with a multinomial treatment, one treatment level must be entered to `focal` to serve as the "treated". Sampling weights are supported for all estimands. The other arguments to `sbw()` can be specified in the call to `weightit()`. See sbw for details. There are some slight difference between the default options in `weightit()` and `sbw()`; importantly, in `weightit()` when `bal_tols_sd` is `TRUE` (the default), the standardized mean difference is not used for categorical variables, and the demoninator of the standardized mean difference corresponds to the standard deviation of the target group (e.g., for the ATT, the denominator is the standard deviation of the treated group).

`"ebcw"` **Empirical balancing calibration weighting.** This method uses the `ATE()` function from the **ATE** package to estimate weights. It works with binary and multinomial treatments. For binary treatments, the ATE, ATT, and ATC can be requested. For multinomial treatments, the ATE and ATT can be requested, and an argument ot `focal` must be specified for the ATT. Sampling weights are supported for all estimands. The other arguments to `ATE()` can be specified in the call to `weightit()`. See `ATE` for details.

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.
`data`	The data.frame originally entered to `weightit()`.
`estimand`	The estimand requested.
`method`	The weight estimation method specified.
`ps`	The estimated or provided propensity scores.
`s.weights`	The provided sampling weights.
`focal`	The focal variable if the ATT was requested with a multinomial treatment.

Note

The **sbw** package is not on CRAN and can be downloaded with the following code: `install.packages("http://www.jrzub`

The other packages are on CRAN and can be installed in the regular way.

Author(s)

Noah Greifer

The code used for using generalized boosted modeling with continuous treatments was adapted from that which appeared in Zhu, Coffman, & Ghosh (2015).

References

Binary treatments

`method = "ps"`

- `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

- 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

`method = "gbm"`

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

`method = "cbps"`

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

`method = "npcbps"`

Fong, Christian, Chad Hazlett, and Kosuke Imai (2015). Parametric and Nonparametric Covariate Balancing Propensity Score for General Treatment Regimes. Unpublished Manuscript. <http://imai.princeton.edu/research/files/CBGPS.pdf>

`method = "ebal"`

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

`method = "sbw"`

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

`method = "ebcw"`

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

Multinomial Treatments

`method = "ps"`

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

`method = "gbm"`

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.

`method = "gbm"`

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).

`method = "cbps"`

Fong, C., Hazlett, C. , and Imai, K. (2015). Parametric and Nonparametric Covariate Balancing Propensity Score for General Treatment Regimes. Unpublished Manuscript. <http://imai.princeton.edu/research/files/CBGPS.pdf>

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,
            exact = NULL,
            s.weights = NULL,
            num.formula = NULL,
```

```
            moments = 1,
            int = FALSE,
            verbose = 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. See Details below.
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 specifiy 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.
exact	a vector or the names of variables in data for which weighting is to be done within catgories. For example, if exact = "gender", weights will be generated seperately within each level of the variable "gender".
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 entropy balancing, empirical balancing callibration weights, and stable balancing 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 callibration weights, and stable balancing weights, whether first-order interactons 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.
`...`	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

`weightitMSM()` works by seperating the estimation of weights into seperate 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 seperate 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>

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