Package 'dbw'

August 28, 2024

Title Doubly Robust Distribution Balancing Weighting Estimation

Version 1.1.4

Description Implements the doubly robust distribution balancing weighting proposed by Katsumata (2024) <doi:10.1017/psrm.2024.23>, which improves the augmented inverse probability weighting (AIPW) by estimating propensity scores with estimating equations suitable for the pre-specified parameter of interest (e.g., the average treatment effects or the average treatment effects on the treated) and estimating outcome models with the estimated inverse probability weights. It also implements the covariate balancing propensity score proposed by Imai and Ratkovic (2014) <doi:10.1111/rssb.12027> and the entropy balancing weighting proposed by Hainmueller (2012) <doi:10.1093/pan/mpr025>, both of which use covariate balancing conditions in propensity score estimation. The point estimate of the parameter of interest and its uncertainty as well as coefficients for propensity score estimation and outcome regression are produced using the M-estimation. The same functions can be used to estimate average outcomes in missing outcome cases.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Depends R (>= 2.10)

Suggests mgcv

URL https://github.com/hirotokatsumata/dbw

BugReports https://github.com/hirotokatsumata/dbw/issues

NeedsCompilation no

Author Hiroto Katsumata [aut, cre, cph]
(<https://orcid.org/0000-0001-5901-8844>)

Maintainer Hiroto Katsumata <hrt.katsumata@gmail.com>

Repository CRAN

Date/Publication 2024-08-28 08:10:16 UTC

Contents

---

dbw *Doubly robust distribution balancing weighting (DBW) estimation*

Description

dbw estimates a pre-specified parameter of interest (e.g., the average treatment effects (ATE) or the average treatment effects on the treated (ATT)) with the augmented inverse probability weighting (AIPW), where propensity scores are estimated using estimating equations suitable for the parameter of interest and outcome models are estimated using inverse probability weights. dbw can also be used to estimate average outcomes (AO) in missing outcome cases.

Usage

```
dbw(
  formula_y,
  formula_ps,
  estimand = "ATE",
  method = "dbw",
  method_y = "wls",
  data,
  normalize = TRUE,
  vcov = TRUE,
  lambda = 0,
  weights = NULL,
  clevel = 0.95,
  tol = 1e-10,
  init_lambda = 0.01
)
```

Arguments

formula_y	an object of class formula (or one that can be coerced to that class): a symbolic description of the potential outcome model to be fitted. This is a model for potential outcomes, so do not include treatment variable in the model. When you want to use non-DR type estimators, only include "1" in the right hand side of the formula for the Hajek estimator and only include "0" for the Horvitz-Thompson estimator. See example below for more details.

`formula_ps` an object of class `formula` (or one that can be coerced to that class): a symbolic description of the propensity score model to be fitted. For the entropy balancing weighting (`method = "eb"`), variables in the right hand side of the formula will be mean balanced.

`estimand` a character string specifying a parameter of interest. Choose "ATT" for the average treatment effects on the treated estimation, "ATE" for the average treatment effects estimation, "ATC" for the average outcomes estimation in missing outcome cases. You can choose "ATEcombined" for the combined estimation for the average treatment effects estimation when using the covariate balancing weighting (`method = "cb"`).

`method` a character string specifying a method for propensity score estimation. Choose "dbw" for the distribution balancing weighting, "cb" for the covariate balancing weighting, "eb" for the entropy balancing weighting, and "mle" for the logistic regression with the maximum likelihood estimation.

`method_y` a character string specifying a method for potential outcome prediction. Choose "wls" for the linear model, "logit" for the logistic regression, "gam" for the generalized additive model for the continuous outcome, and "gambinom" for the generalized additive model for the binary outcome. Note that variance-covariance matrix is calculated only when `method_y = "wls"` or `method_y = "logit"`.

`data` a data frame (or one that can be coerced to that class) containing the outcomes and the variables in the model.

`normalize` a logical parameter indicating whether to normalize the estimated weights to sum up to one for each treatment group for `method = "dbw"`. Default is TRUE.

`vcov` a logical parameter indicating whether to estimate the variance. Default is TRUE.

`lambda` a parameter taking 0 or larger specifying the degree of the L2-regularization for propensity score estimation. `lambda = 0` (default) means no regularization.

`weights` an optional vector of 'prior weights' (e.g. sampling weights) to be used in the fitting process. Should be NULL or a numeric vector.

`clevel` confidence levels. Default is 0.95.

`tol` a tolerance parameter for `method = "dbw"`. Default is 1e-10.

`init_lambda` a parameter for `method = "dbw"` to set the lambda value for the initial values estimation, where $lambda_init = lambda * init_lambda$. Default is 0.01.

Details

The treatment variable (or, response variable in missing outcome cases) must be binary and coded as 0 (for controlled or missing observations) or 1 (for treated or non-missing observations).

When the data frame has incomplete cases, which have NAs for either of the treatment variable, the outcome variable, or explanatory variables either for propensity score or outcome model estimation, dbw conducts listwise deletion. Returned values (e.g., `est_weights`, `ps`, `data`) do not contain values for these deleted cases.

For propensity score estimation, dbw can utilize the distribution balancing weighting (`method = "dbw"`), covariate balancing weighting (`method = "cb"`), entropy balancing weighting (`method =`

"eb"), or standard maximum likelihood estimation (method = "mle"). For the covariate balancing weighting and entropy balancing weighting, dbw runs much faster than the original functions (CBPS and ebalance) by using loss-function-based algorithms, which also results in more accurate covariate balance. For the ATT and ATC estimation, the distribution balancing weighting, covariate balancing weighting, and entropy balancing weighting are theoretically equivalent and dbw implements accordingly.

The parameter of interest is estimated by the AIPW estimator, where inverse probability weights are standardized within each treatment group by being devided by the size of the group after being calculated as $t_i/\pi_i - (1-t_i)/(1-\pi_i)$ for the ATE estimation, $(t_i-\pi_i)/(1-\pi_i)$ for the ATT estimation, $(t_i-\pi_i)/\pi_i$ for the ATC estimation, and t_i/π_i for the missing outcome cases. The resulting inverse probability weights sum to 1 for the distribution balancing weighting, covariate balancing weighting, and entropy balancing weighting estimators without regularization.

The variance-covariance matrix for the parameter of interest and ancillary parameters is calculated using the sandwich variance formula obtained in the M-estimation framework.

When using regularization for propensity score estimation (lambda > 0), you should standardize the covariates for propensity score estimation by std_comp before using dbw. See example below for more details.

For the ATE estimation, it is recommended to specify the estimand as "ATE", but you may specify it as "ATEcombined" when using the covariate balancing weighting. The former utilizes the separated propensity score estimation whereas the latter utilizes the combined estimation, and the former should produce smaller biases and variances. Note that the former estimates two propensity scores for each observation by estimating two propensity score functions with different estimating equations.

For the AO estimation, NA values for the outcome variable for missing cases (the response variable taking "0") are not deleted. For this processing, the outcome variable name must not contain spaces.

Value

dbw returns an object of "dbw" class.

The function summary (i.e., summary.dbw) can be used to obtain or print a summary of the results.

An object of class "dbw" is a list containing the following components:

est	the point estimate of the parameter of interest.
coef_ps	a named vector of coefficients for propensity score estimation. A list of two sets of coefficients for two sets of propensity scores (one for estimating $E[Y(1)]$ and the other for estimating $E[Y(0)]$) is returned when estimand = "ATE".
coef_y	a named vector of coefficients for outcome model estimation. A list of two sets of coefficients for two sets of outcome models (one for estimating $E[Y(1)]$ and the other for estimating $E[Y(0)]$) is returned when estimand = "ATE" and estimand = "ATEcombined".
varcov	the variance-covariance matrix of the coefficients and the parameter of interest.
est_weights	the estimated inverse probability weights.
ps	the estimated propensity scores. A list of two sets of the estimated propensity scores (one for estimating $E[Y(1)]$ and the other for estimating $E[Y(0)]$) is returned when estimand = "ATE".

predicted_y	the predicted outcomes. A list of two sets of the predicted outcomes (one for estimating $E[Y(1)]$ and the other for estimating $E[Y(0)]$) is returned when estimand = "ATE" and estimand = "ATEcombined".
converged	logical. Were the propensity score estimation algorithms judged to have converged?
effn	the effective sample size for the parameter of interest estimation.
effn_original	the effective sample size with the initial weights.
estimand	the parameter of interest specified.
method	the method for propensity score estimation specified.
method_y	the method for outcome model estimation specified.
response	the treatment vector. The response (non-missingness) vector when the missing outcome cases.
outcome	the outcome vector.
original_weights	the weights initially supplied, a vector of 1s if none were.
ci	a matrix of the confidence intervals for the parameter of interest.
formula_y	the outcome model formula specified.
formula_ps	the propensity score model formula specified.
call	the matched call.
data	the data argument.
normalize	a logical argument indicating whether to normalize the estimated weights for each treatment group for method = "dbw".
lambda	the parameter specifying the degree of the L2-regularization.

Author(s)

Hiroto Katsumata

References

Katsumata, Hiroto. 2024. "How Should We Estimate Inverse Probability Weights with Possibly Misspecified Propensity Score Models?" Political Science Research and Methods.

Imai, Kosuke and Marc Ratkovic. 2014. "Covariate Balancing Propensity Score." Journal of the Royal Statistical Society, Series B (Statistical Methodology) 76 (1): 243–63.

Hainmueller, Jens. 2012. "Entropy Balancing for Causal Effects: A Multivariate Reweighting Method to Produce Balanced Samples in Observational Studies." Political Analysis 20 (1): 25–46.

See Also

summary.dbw, std_comp, gam

Examples

```
# Simulation from Kang and Shafer (2007) and Imai and Ratkovic (2014)
# ATE estimation
# True ATE is 10
tau <- 10
set.seed(12345)
n <- 1000
X <- matrix(stats::rnorm(n * 4, mean = 0, sd = 1), nrow = n, ncol = 4)
prop <- 1 / (1 + exp(X[, 1] - 0.5 * X[, 2] +
                     0.25 * X[, 3] + 0.1 * X[, 4]))
treat <- rbinom(n, 1, prop)
y <- 210 +
     27.4 * X[, 1] + 13.7 * X[, 2] + 13.7 * X[, 3] + 13.7 * X[, 4] +
     tau * treat + stats::rnorm(n = n, mean = 0, sd = 1)
ybinom <- (y > 210) + 0
df0 <- data.frame(X, treat, y, ybinom)
colnames(df0) <- c("x1", "x2", "x3", "x4", "treat", "y", "ybinom")

# Variables for a misspecified model
Xmis <- data.frame(x1mis = exp(X[, 1] / 2),
                   x2mis = X[, 2] * (1 + exp(X[, 1]))^(-1) + 10,
                   x3mis = (X[, 1] * X[, 3] / 25 + 0.6)^3,
                   x4mis = (X[, 2] + X[, 4] + 20)^2)

# Data frame and formulas for propensity score estimation
df <- data.frame(df0, Xmis)
formula_ps_c <- stats::as.formula(treat ~ x1 + x2 + x3 + x4)
formula_ps_m <- stats::as.formula(treat ~ x1mis + x2mis +
                                          x3mis + x4mis)

# Formula for a misspecified outcome model
formula_y <- stats::as.formula(y ~ x1mis + x2mis + x3mis + x4mis)


# Correct propensity score model

# Distribution balancing weighting with normalization and
#   without regularization
fitdbwc <- dbw(formula_y = formula_y, formula_ps = formula_ps_c,
               estimand = "ATE", method = "dbw",
               method_y = "wls", data = df, normalize = TRUE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
fitdbwc
summary(fitdbwc)

# Covariate balancing weighting function without regularization
fitcbwc <- dbw(formula_y = formula_y, formula_ps = formula_ps_c,
               estimand = "ATE", method = "cb",
               method_y = "wls", data = df, normalize = TRUE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
```

```
summary(fitcbwc)

# Entropy balancing weighting function without regularization
fitebwc <- dbw(formula_y = formula_y, formula_ps = formula_ps_c,
               estimand = "ATE", method = "eb",
               method_y = "wls", data = df, normalize = TRUE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
summary(fitebwc)

# Standard logistic regression
fitmlec <- dbw(formula_y = formula_y, formula_ps = formula_ps_c,
               estimand = "ATE", method = "mle",
               method_y = "wls", data = df, normalize = FALSE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
summary(fitmlec)

# Distribution balancing weighting without normalization and
#   without regularization
fitdbwcnn <- dbw(formula_y = formula_y, formula_ps = formula_ps_c,
                 estimand = "ATE", method = "dbw",
                 method_y = "wls", data = df, normalize = FALSE,
                 vcov = TRUE, lambda = 0, weights = NULL,
                 clevel = 0.95)
summary(fitdbwcnn)


# Misspecified propensity score model

# Distribution balancing weighting with normalization and
#   without regularization
fitdbwm <- dbw(formula_y = formula_y, formula_ps = formula_ps_m,
               estimand = "ATE", method = "dbw",
               method_y = "wls", data = df, normalize = TRUE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
summary(fitdbwm)

# Covariate balancing weighting function without regularization
fitcbwm <- dbw(formula_y = formula_y, formula_ps = formula_ps_m,
               estimand = "ATE", method = "cb",
               method_y = "wls", data = df, normalize = TRUE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
summary(fitcbwm)

# Entropy balancing weighting function without regularization
fitebwm <- dbw(formula_y = formula_y, formula_ps = formula_ps_m,
               estimand = "ATE", method = "eb",
               method_y = "wls", data = df, normalize = TRUE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
```

```
summary(fitebwm)

# Standard logistic regression
fitmlem <- dbw(formula_y = formula_y, formula_ps = formula_ps_m,
               estimand = "ATE", method = "mle",
               method_y = "wls", data = df, normalize = FALSE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
summary(fitmlem)

# Distribution balancing weighting without normalization and
#   without regularization
fitdbwmnn <- dbw(formula_y = formula_y, formula_ps = formula_ps_m,
               estimand = "ATE", method = "dbw",
               method_y = "wls", data = df, normalize = FALSE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
summary(fitdbwmnn)

# Distribution balancing weighting with normalization and
#   with regularization
# Standardization
res_std_comp <- std_comp(formula_y = formula_y,
                         formula_ps = formula_ps_m,
                         estimand = "ATE", method_y = "wls",
                         data = df, std = TRUE,
                         weights = NULL)
# Estimation
fitdbwmr <- dbw(formula_y = formula_y,
                formula_ps = res_std_comp$formula_ps,
                estimand = "ATE", method = "dbw", method_y = "wls",
                data = res_std_comp$data, normalize = TRUE,
                vcov = TRUE, lambda = 0.01,
                weights = res_std_comp$weights, clevel = 0.95)
summary(fitdbwmr)

# Covariate balancing weighting function with an estimating equation
#  for the original covariate balancing propensity score method
fitcbwmcmb <- dbw(formula_y = formula_y, formula_ps = formula_ps_m,
                  estimand = "ATEcombined", method = "cb",
                  method_y = "wls", data = df, normalize = TRUE,
                  vcov = TRUE, lambda = 0, weights = NULL,
                  clevel = 0.95)
summary(fitcbwmcmb)



# Formula for a misspecified outcome model for the GAM
library(mgcv)
formula_y_gam <- stats::as.formula(y ~ s(x1mis) + s(x2mis) +
                                       s(x3mis)+ s(x4mis))

# Distribution balancing weighting with the GAM
```

```
fitdbwmg <- dbw(formula_y = formula_y_gam, formula_ps = formula_ps_m,
                estimand = "ATE", method = "dbw",
                method_y = "gam", data = df, normalize = TRUE,
                vcov = TRUE, lambda = 0, weights = NULL,
                clevel = 0.95)
summary(fitdbwmg)


# Binary outcome case
# Empirically correct ATE
ybinom_t <- (y + (1 - treat) * 10 > 210) + 0
ybinom_c <- (y - treat * 10 > 210) + 0
ATEbinom <- mean(ybinom_t - ybinom_c)
ATEbinom

# Formula for a misspecified binary outcome model
formula_y_bin <- stats::as.formula(ybinom ~ x1mis + x2mis +
                                           x3mis + x4mis)

# Distribution balancing weighting for the binary outcome
fitdbwmbin <- dbw(formula_y = formula_y_bin,
                  formula_ps = formula_ps_m,
                  estimand = "ATE", method = "dbw",
                  method_y = "logit", data = df, normalize = TRUE,
                  vcov = TRUE, lambda = 0, weights = NULL,
                  clevel = 0.95)
summary(fitdbwmbin)


# Standard logistic regression with the Horvitz-Thompson estimator
fitmlem_ht <- dbw(formula_y = y ~ 0, formula_ps = formula_ps_m,
                  estimand = "ATE", method = "mle",
                  method_y = "wls", data = df, normalize = FALSE,
                  vcov = TRUE, lambda = 0, weights = NULL,
                  clevel = 0.95)
summary(fitmlem_ht)

# Standard logistic regression with the Hajek estimator
fitmlem_hj <- dbw(formula_y = y ~ 1, formula_ps = formula_ps_m,
                  estimand = "ATE", method = "mle",
                  method_y = "wls", data = df, normalize = FALSE,
                  vcov = TRUE, lambda = 0, weights = NULL,
                  clevel = 0.95)
summary(fitmlem_hj)
```

---

plot.dbw *Summarize and plot covariate balance*

Description

Summarizes and plots covariate balance between treatment and control groups before and after the distribution balancing weighting.

Usage

```
## S3 method for class 'dbw'
plot(
  x,
  addcov = NULL,
  standardize = TRUE,
  plot = TRUE,
  absolute = TRUE,
  threshold = 0,
  sort = TRUE,
  ...
)
```

Arguments

x	an object of class "dbw", usually, a result of a call to dbw.
addcov	a one-sided formula specifying additional covariates whose balance is checked. Covariates containing NAs are automatically dropped.
standardize	a logical value indicating whether weighted mean differences are standardized or not.
plot	a logical value indicating whether a covariate balance plot is displayed.
absolute	a logical value indicating whether the absolute values of differences in weighted means are used in the covariate balance plot.
threshold	an optional numeric vector used as threshold markers in the covariate balance plot.
sort	a logical value indicating whether covariates in the covariate balance plot are sorted by the values of differences in the weighted means before the distribution balancing weighting.
...	other graphical parameters (see par).

Details

Estimated weights are normalized to sum to one. Position of the legend is determined internally. Set absolute = TRUE and sort = TRUE for choosing a better location automatically.

Value

A matrix whose rows are the covariates and columns are the differences in the (un)standardized weighted mean between the treatment and control groups before (diff.un) and after (diff.adj) the distribution balancing weighting. The standardized weighted mean is the weighted mean divided by the standard deviation of the covariate for the target population (the treatment group for the average treatment effects on the treated estimation, the control group for the average treatment effects on the

controlled estimation and the whole population for the other quantity of interest). The differences in the categorical variables are not standardized.

Author(s)

Hiroto Katsumata

Examples

```
# Simulation from Kang and Shafer (2007) and Imai and Ratkovic (2014)
# ATE estimation
# True ATE is 10
tau <- 10
set.seed(12345)
n <- 1000
X <- matrix(stats::rnorm(n * 4, mean = 0, sd = 1), nrow = n, ncol = 4)
prop <- 1 / (1 + exp(X[, 1] - 0.5 * X[, 2] +
                     0.25 * X[, 3] + 0.1 * X[, 4]))
treat <- rbinom(n, 1, prop)
y <- 210 +
     27.4 * X[, 1] + 13.7 * X[, 2] + 13.7 * X[, 3] + 13.7 * X[, 4] +
     tau * treat + stats::rnorm(n = n, mean = 0, sd = 1)
ybinom <- (y > 210) + 0
df0 <- data.frame(X, treat, y, ybinom)
colnames(df0) <- c("x1", "x2", "x3", "x4", "treat", "y", "ybinom")

# Variables for a misspecified model
Xmis <- data.frame(x1mis = exp(X[, 1] / 2),
                   x2mis = X[, 2] * (1 + exp(X[, 1]))^(-1) + 10,
                   x3mis = (X[, 1] * X[, 3] / 25 + 0.6)^3,
                   x4mis = (X[, 2] + X[, 4] + 20)^2)

# Data frame and formulas for propensity score estimation
df <- data.frame(df0, Xmis)
formula_ps_c <- stats::as.formula(treat ~ x1 + x2 + x3 + x4)
formula_ps_m <- stats::as.formula(treat ~ x1mis + x2mis +
                                          x3mis + x4mis)

# Formula for a misspecified outcome model
formula_y <- stats::as.formula(y ~ x1mis + x2mis + x3mis + x4mis)

# Set plot margins
oldpar <- par(no.readonly = TRUE) # Just for adjusting plot margins
par(mar = c(5.1, 5.1, 4.1, 2.1)) # Just for adjusting plot margins

# Misspecified propensity score model

# Distribution balancing weighting without regularization
fitdbwm <- dbw(formula_y = formula_y, formula_ps = formula_ps_m,
               estimand = "ATE", method = "dbw",
               method_y = "wls", data = df, normalize = TRUE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
```

```
plot(fitdbwm, addcov = ~ x1 + x2 + x3 + x4, threshold = c(0.1, 0.2))

# Non-absolute values
plot(fitdbwm, addcov = ~ x1 + x2 + x3 + x4, absolute = FALSE,
     threshold = c(0.1, 0.2))

# No sorting
plot(fitdbwm, addcov = ~ x1 + x2 + x3 + x4,
     threshold = c(0.1, 0.2), sort = FALSE)


# Covariate balancing weighting function without regularization
fitcbwm <- dbw(formula_y = formula_y, formula_ps = formula_ps_m,
               estimand = "ATE", method = "cb",
               method_y = "wls", data = df, normalize = TRUE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
plot(fitcbwm, addcov = ~ x1 + x2 + x3 + x4, threshold = c(0.1, 0.2))

# Entropy balancing weighting function without regularization
fitebwm <- dbw(formula_y = formula_y, formula_ps = formula_ps_m,
               estimand = "ATE", method = "eb",
               method_y = "wls", data = df, normalize = TRUE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
plot(fitebwm, addcov = ~ x1 + x2 + x3 + x4, threshold = c(0.1, 0.2))

# Standard logistic regression
fitmlem <- dbw(formula_y = formula_y, formula_ps = formula_ps_m,
               estimand = "ATE", method = "mle",
               method_y = "wls", data = df, normalize = FALSE,
               vcov = TRUE, lambda = 0, weights = NULL,
               clevel = 0.95)
plot(fitmlem, addcov = ~ x1 + x2 + x3 + x4, threshold = c(0.1, 0.2))

par(oldpar)

# Display the covariate balance matrix
cb <- plot(fitdbwm, addcov = ~ x1 + x2 + x3 + x4, plot = FALSE)
cb
```

`print.dbw` *Print the distribution balancing weighting estimation results*

Description

Prints a fitted dbw object.

Usage

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

Arguments

x	an object of class "dbw", usually, a result of a call to dbw.
...	additional arguments to be passed to print.

Value

No retrun value, called for side effects.

Author(s)

Hiroto Katsumata

See Also

dbw, print

---

std_comp *Generate standardized complete cases*

Description

Standardizes covariates for propensity score estimation before the distribution balancing weighting dbw with the regularization.

Usage

```
std_comp(
  formula_y,
  formula_ps,
  estimand = "ATE",
  method_y = "wls",
  data,
  std = TRUE,
  weights = NULL
)
```

Arguments

formula_y	an object of class formula (or one that can be coerced to that class): a symbolic description of the potential outcome model to be fitted. When you want to use non-DR type estimators, only include "1" in the right hand side of the formula for the Hajek estimator and only include "0" for the Horvitz-Thompson estimator. See example of dbw for more details.
formula_ps	an object of class formula (or one that can be coerced to that class): a symbolic description of the propensity score model to be fitted. For the entropy balancing weighting (method = "eb"), variables in the right hand side of the formula will be mean balanced.
estimand	a character string specifying a parameter of interest. Choose "ATT" for the average treatment effects on the treated estimation, "ATE" for the average treatment effects estimation, "ATC" for the average outcomes estimation in missing outcome cases. You can choose "ATEcombined" for the combined estimation for the average treatment effects estimation when using the covariate balancing weighting (method = "cb").
method_y	a character string specifying a method for potential outcome prediction. Choose "wls" for the linear model, "logit" for the logistic regression, "gam" for the generalized additive model for the continuous outcome, and "gambinom" for the generalized additive model for the binary outcome.
data	a data frame (or one that can be coerced to that class) containing the outcomes and the variables in the model.
std	a logical parameter indicating whether to standardize the covariates for propensity score estimation.
weights	an optional vector of 'prior weights' (e.g. sampling weights) to be used in the fitting process. Should be NULL or a numeric vector.

Details

std_comp first extracts complete cases for both propensity score estimation and outcome model estimation. Then it standardizes covariates for propensity score estimation by takeing the provided weights into account. The returned data frame is the "design matrix", which contains a set of dummy variables (depending on the contrasts) for factors and similarly expanded interaction terms.

For the AO estimation, NA values for the outcome variable for missing cases (the response variable taking "0") are not deleted. For this processing, the outcome variable name must not contain spaces.

Value

data	the complete-case data frame containing the outcome variable, the response (treatment) variable, and the standardized covariates for propensity score estimation.
weights	the initially-supplied weights for the complete cases, a vector of 1s if none were.
formula_ps	the automatically-created formula for propensity score estimation, which includes expanded factors and interactions.
mean_x	the weighted mean of each covariates.
sd_x	the weighted standard deviation of each covariates.

Author(s)

Hiroto Katsumata

See Also

dbw

Examples

```
# Simulation from Kang and Shafer (2007) and Imai and Ratkovic (2014)
# ATE estimation
# True ATE is 10
tau <- 10
set.seed(12345)
n <- 1000
X <- matrix(stats::rnorm(n * 4, mean = 0, sd = 1), nrow = n, ncol = 4)
prop <- 1 / (1 + exp(X[, 1] - 0.5 * X[, 2] +
                     0.25 * X[, 3] + 0.1 * X[, 4]))
treat <- rbinom(n, 1, prop)
y <- 210 +
     27.4 * X[, 1] + 13.7 * X[, 2] + 13.7 * X[, 3] + 13.7 * X[, 4] +
     tau * treat + stats::rnorm(n = n, mean = 0, sd = 1)
ybinom <- (y > 210) + 0
df0 <- data.frame(X, treat, y, ybinom)
colnames(df0) <- c("x1", "x2", "x3", "x4", "treat", "y", "ybinom")

# Variables for a misspecified model
Xmis <- data.frame(x1mis = exp(X[, 1] / 2),
                   x2mis = X[, 2] * (1 + exp(X[, 1]))^(-1) + 10,
                   x3mis = (X[, 1] * X[, 3] / 25 + 0.6)^3,
                   x4mis = (X[, 2] + X[, 4] + 20)^2)

# Data frame and formulas for propensity score estimation
df <- data.frame(df0, Xmis)
formula_ps_c <- stats::as.formula(treat ~ x1 + x2 + x3 + x4)
formula_ps_m <- stats::as.formula(treat ~ x1mis + x2mis +
                                          x3mis + x4mis)

# Formula for a misspecified outcome model
formula_y <- stats::as.formula(y ~ x1mis + x2mis + x3mis + x4mis)


# Distribution balancing weighting with regularization
# Standardization
res_std_comp <- std_comp(formula_y = formula_y,
                         formula_ps = formula_ps_m,
                         estimand = "ATE", method_y = "wls",
                         data = df, std = TRUE,
                         weights = NULL)
fitdbwmr <- dbw(formula_y = formula_y,
                formula_ps = res_std_comp$formula_ps,
                estimand = "ATE", method = "dbw", method_y = "wls",
```

```
                data = res_std_comp$data, vcov = TRUE,
                lambda = 0.01, weights = res_std_comp$weights,
                clevel = 0.95)
summary(fitdbwmr)
```

---

summary.dbw *Summarize the distribution balancing weighting estimation results*

Description

Prints a summary of a fitted dbw object.

Usage

```
## S3 method for class 'dbw'
summary(object, ...)
```

Arguments

object	an object of class "dbw", usually, a result of a call to dbw.
...	additional arguments to be passed to summary.

Details

Prints a summary of a dbw object, in a format similar to glm.

Value

call	the matched call.
est	the point estimate of the parameter of interest.
coefficients	a table including coefficients, standard errors, z-values, and two-sided p-values.
effn	the effective sample size for the parameter of interest estimation.

Author(s)

Hiroto Katsumata

See Also

dbw, summary

Examples

```
# For examples see example(dbw)
```

Index