Package: OutcomeWeights (via r-universe)

October 13, 2026

Type Package

Title Outcome Weights of Treatment Effect Estimators

Version 0.2.0.9000

Description Many treatment effect estimators can be written as weighted outcomes. These weights have established use cases like checking covariate balancing via packages like 'cobalt'. This package takes the original estimator objects and outputs these outcome weights. It builds on the general framework of Knaus (2024) <doi:10.48550/arXiv.2411.11559>. This version is compatible with the 'grf' package and provides an internal implementation of Double Machine Learning.

License GPL-3

Encoding UTF-8

URL https://github.com/MCKnaus/OutcomeWeights

BugReports https://github.com/MCKnaus/OutcomeWeights/issues

Imports ggplot2, grf, methods

LinkingTo Rcpp, RcppArmadillo

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, pkgdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs make

Repository https://mcknaus.r-universe.dev

Date/Publication 2026-04-28 12:40:15 UTC

RemoteUrl https://github.com/mcknaus/outcomeweights

RemoteRef HEAD

RemoteSha 0c94f940b04c14d0247b46842af37752e306b79e

Contents

dml_with_smoother . . . 2
get_outcome_weights . . . 4
get_outcome_weights.causal_forest . . . 5
get_outcome_weights.dml_with_smoother . . . 7
get_outcome_weights.DoubleML . . . 7
get_outcome_weights.instrumental_forest . . . 9
get_outcome_weights.ivreg . . . 10
get_outcome_weights.lm . . . 11
get_outcome_weights.lm_robust . . . 12
get_smoother_weights . . . 12
get_smoother_weights.cv.plasso . . . 13
get_smoother_weights.drf . . . 14
get_smoother_weights.lm . . . 15
get_smoother_weights.lm_robust . . . 16
get_smoother_weights.NuisanceParameters . . . 17
get_smoother_weights.ranger . . . 18
get_smoother_weights.regression_forest . . . 19
get_smoother_weights.rlasso . . . 20
get_smoother_weights.xgb.Booster . . . 21
NuPa_honest_forest . . . 22
pive_weight_maker . . . 23
plot.dml_with_smoother . . . 24
prep_cf_mat . . . 25
standardized_mean_differences . . . 25
summary.dml_with_smoother . . . 26
summary.get_outcome_weights . . . 27
summary.standardized_mean_differences . . . 27

Index **29**

`dml_with_smoother` *Double ML estimators with outcome smoothers*

Description

Existing Double ML implementations are too general to easily extract smoother matrices required to be compatible with the get_forest_weights() method. This motivates yet another Double ML implementation.

Usage

```
dml_with_smoother(
  Y,
  D,
  X,
  Z = NULL,
```

```
  estimators = c("PLR", "PLR_IV", "AIPW_ATE", "Wald_AIPW", "AIPW_ATT", "AIPW_ATU"),
  smoother = "honest_forest",
  n_cf_folds = 5,
  n_reps = 1,
  ...
)
```

Arguments

Y	Numeric vector containing the outcome variable.
D	Optional binary treatment variable.
X	Covariate matrix with N rows and p columns.
Z	Optional binary instrumental variable.
estimators	String (vector) indicating which estimators should be run. Current menu: c("PLR","PLR_IV","AIPW_AT
smoother	Indicate which smoother to be used for nuisance parameter estimation. Currently only available option "honest_forest" from the **grf** package.
n_cf_folds	Number of cross-fitting folds. Default is 5.
n_reps	Number of repetitions of cross-fitting. Default is 1.
...	Options to be passed to smoothers.

Value

A list with three entries:

- results: a list storing the results, influence functions, and score functions of each estimator
- NuPa.hat: a list storing the estimated nuisance parameters and the outcome smoother matrices

References

Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W., & Robins, J. (2018). Double/debiased machine learning for treatment and structural parameters. The Econometrics Journal, 21(1), C1-C68.

Knaus, M. C. (2024). Treatment effect estimators as weighted outcomes, https://arxiv.org/abs/2411.11559.

Examples

```
# Sample from DGP borrowed from grf documentation
n = 200
p = 5
X = matrix(rbinom(n * p, 1, 0.5), n, p)
Z = rbinom(n, 1, 0.5)
Q = rbinom(n, 1, 0.5)
W = Q * Z
tau =  X[, 1] / 2
Y = rowSums(X[, 1:3]) + tau * W + Q + rnorm(n)
```

```
# Run outcome regression and extract smoother matrix
# Run DML and look at results
dml = dml_with_smoother(Y,W,X,Z)
results_dml = summary(dml)
plot(dml)

# Get weights
omega_dml = get_outcome_weights(dml)

# Observe that they perfectly replicate the original estimates
all.equal(as.numeric(omega_dml$omega %*% Y),
          as.numeric(as.numeric(results_dml[,1])))

# The weights can then be passed to the cobalt package for example.
```

`get_outcome_weights` *Outcome weights method*

Description

This is a generic method for getting outcome weights. It calculates the outcome weights for objects created by other packages.

Usage

```
get_outcome_weights(object, ...)
```

Arguments

`object`	An estimated object. See get_outcome_weight.<compatible_fct> in the package documentation for compatible object classes.
`...`	Additional arguments specific to object class implementations. See the documentation which object requires which additional arguments.

Value

A list of at least these components:

- omega: matrix (number of point estimates x number of estimation units) of outcome weights
- treat: the treatment indicator to make it compatible with the cobalt package

References

Knaus, M. C. (2024). Treatment effect estimators as weighted outcomes, https://arxiv.org/abs/2411.11559.

get_outcome_weights.causal_forest
Outcome weights for the causal_forest *function*

Description

Post-estimation command to extract outcome weights for causal forest implemented via the causal_forest function from the **grf** package.

Usage

```
## S3 method for class 'causal_forest'
get_outcome_weights(
  object,
  ...,
  S,
  newdata = NULL,
  S.tau = NULL,
  target = "CATE",
  checks = TRUE
)
```

Arguments

object	An object of class causal_forest, i.e. the result of running causal_forest.
...	Pass potentially generic get_outcome_weights options.
S	A smoother matrix reproducing the outcome predictions used in building the instrumental_forest. Obtained by calling get_forest_weights() for the regression_forest object producing the outcome predictions.
newdata	Corresponds to newdata option in predict.causal_forest. If NULL, out-of-bag outcome weights, otherwise for those for the provided test data returned.
S.tau	Required if target != "CATE", then S.tau is the CATE smoother obtained from running get_outcome_weights() with target == "CATE".
target	Target parameter for which outcome weights should be extracted. Either "CATE" or the corresponding value of the target.sample option of the average_treatment_effect function c("all", "treated", "control", "overlap").
checks	Default TRUE checks whether weights numerically replicate original estimates. Only set FALSE if you know what you are doing and need to save computation time.

Value

get_outcome_weights object with omega containing weights and treat the treatment

References

Athey, S., Tibshirani, J., & Wager, S. (2019). Generalized random forest. The Annals of Statistics, 47(2), 1148-1178.

Knaus, M. C. (2024). Treatment effect estimators as weighted outcomes, https://arxiv.org/abs/2411.11559.

Examples

```
# Sample from DGP borrowed from grf documentation
n = 500
p = 10
X = matrix(rnorm(n * p), n, p)
W = rbinom(n, 1, 0.5)
Y = pmax(X[, 1], 0) * W + X[, 2] + pmin(X[, 3], 0) + rnorm(n)

# Run outcome regression and extract smoother matrix
forest.Y = grf::regression_forest(X, Y)
Y.hat = predict(forest.Y)$predictions
outcome_smoother = grf::get_forest_weights(forest.Y)

# Run causal forest with external Y.hats
c.forest = grf::causal_forest(X, Y, W, Y.hat = Y.hat)

# Predict on out-of-bag training samples.
cate.oob = predict(c.forest)$predictions

# Predict using the forest.
X.test = matrix(0, 101, p)
X.test[, 1] = seq(-2, 2, length.out = 101)
cate.test = predict(c.forest, X.test)$predictions

# Calculate outcome weights
omega_oob = get_outcome_weights(c.forest,S = outcome_smoother)
omega_test = get_outcome_weights(c.forest,S = outcome_smoother,newdata = X.test)

# Observe that they perfectly replicate the original CATEs
all.equal(as.numeric(omega_oob$omega %*% Y),
          as.numeric(cate.oob))
all.equal(as.numeric(omega_test$omega %*% Y),
          as.numeric(cate.test))

# Also the ATE estimates are perfectly replicated
omega_ate = get_outcome_weights(c.forest,target = "all",
                                S = outcome_smoother,
                                S.tau = omega_oob$omega)
all.equal(as.numeric(omega_ate$omega %*% Y),
          as.numeric(grf::average_treatment_effect(c.forest, target.sample = "all")[1]))

# The omega weights can be plugged into balancing packages like cobalt
```

get_outcome_weights.dml_with_smoother

Outcome weights for the `dml_with_smoother` *function*

Description

Post-estimation command to extract outcome weights for double ML run with an outcome smoother.

Usage

```
## S3 method for class 'dml_with_smoother'
get_outcome_weights(object, ..., all_reps = FALSE)
```

Arguments

`object`	An object of class `dml_with_smoother`, i.e. the result of running `dml_with_smoother`.
`...`	Pass potentially generic get_outcome_weights options.
`all_reps`	If `TRUE`, outcomes weights of each repetitions passed. Default `FALSE`.

Value

- If `all_reps` == `FALSE`: get_outcome_weights object
- If `all_reps` == `TRUE`: additionally list `omega_all_reps`: A list containing the outcome weights of each repetition.

References

Knaus, M. C. (2024). Treatment effect estimators as weighted outcomes, https://arxiv.org/abs/2411.11559.

get_outcome_weights.DoubleML

Outcome weights for a `DoubleML` *object*

Description

Post-estimation command to extract outcome weights for a `DoubleML` object from the **DoubleML** package.

Usage

```
## S3 method for class 'DoubleML'
get_outcome_weights(object, dml_data, ...)
```

Arguments

object	A DoubleML object of class DoubleMLPLR, DoubleMLPLIV, DoubleMLIIVM, or DoubleMLIRM.
dml_data	The DoubleMLData object used in estimation.
...	Pass potentially generic get_outcome_weights options.

Details

Outcome weights are available for DoubleMLPLR, DoubleMLPLIV, DoubleMLIIVM, and DoubleMLIRM models.

Value

An object of class get_outcome_weights containing:

omega Numeric vector of outcome weights of length N.

treat Numeric vector of treatment assignments.

References

Knaus, M. C. (2024). Treatment effect estimators as weighted outcomes, https://arxiv.org/abs/2411.11559.

Bach P, Kurz MS, Chernozhukov V, Spindler M, Klaassen S (2024). "DoubleML: An Object-Oriented Implementation of Double Machine Learning in R." Journal of Statistical Software, 108(3), 1-56, https://doi.org/10.18637/jss.v108.i03.

Examples

```
## Not run:
set.seed(123)

# Set the parameters
params <- list(alpha = 0, subsample = 1, max_delta_step = 0, base_score = 0)

# Define DML objects
ml_regr_xgb <- do.call(lrn, c(list("regr.xgboost"), params))
ml_l_plr <- ml_regr_xgb$clone()
ml_m_plr <- ml_regr_xgb$clone()
plr_data <- DoubleML::make_plr_CCDDHNR2018(500, dim_x = 10)
plr_obj <- DoubleML::DoubleMLPLR$new(plr_data, ml_l_plr, ml_m_plr, n_folds = 3)

# Fit the model
plr_obj$fit(store_models = TRUE, store_predictions = TRUE)

# Calculate the outcome weights
omega_plr <- get_outcome_weights(plr_obj, plr_data)

# Check equivalence
all.equal(omega_plr$omega %*% plr_data$data$y, plr_obj$all_coef)
```

```
## End(Not run)
```

`get_outcome_weights.instrumental_forest` *Outcome weights for the* `instrumental_forest` *function*

Description

Post-estimation command to extract outcome weights for instrumental forest implemented via the `instrumental_forest` function from the **grf** package.

Usage

```
## S3 method for class 'instrumental_forest'
get_outcome_weights(
  object,
  ...,
  S,
  S.tau = NULL,
  newdata = NULL,
  target = "CLATE",
  compliance.score = NULL,
  checks = TRUE
)
```

Arguments

`object`	An object of class `instrumental_forest`, i.e. the result of running `instrumental_forest`.
`...`	Pass potentially generic get_outcome_weights options.
`S`	A smoother matrix reproducing the outcome predictions used in building the `instrumental_forest`. Obtained by calling `get_forest_weights()` for the `regression_forest` object producing the outcome predictions.
`S.tau`	Required if `target == "all"`, then S.tau is the CLATE smoother obtained from running `get_outcome_weights()` with `target == "CLATE"`.
`newdata`	Corresponds to `newdata` option in `predict.instrumental_forest`. If `NULL`, out-of-bag outcome weights, otherwise for those for the provided test data returned.
`target`	Target parameter for which outcome weights should be extracted `c("CLATE","all")` where the latter corresponds to LATE.
`compliance.score`	Only relevant if `compliance.score` is passed to `average_treatment_effect`. Then pass the same argument here. Otherwise the identical auxiliary forest is internally estimated.
`checks`	Default `TRUE` checks whether weights numerically replicate original estimates. Only set `FALSE` if you know what you are doing and want to save computation time.

Value

get_outcome_weights object with omega containing weights and treat the treatment

References

Athey, S., Tibshirani, J., & Wager, S. (2019). Generalized random forest. The Annals of Statistics, 47(2), 1148-1178.

Knaus, M. C. (2024). Treatment effect estimators as weighted outcomes, https://arxiv.org/abs/2411.11559.

Examples

```
# Sample from DGP borrowed from grf documentation
n = 2000
p = 5
X = matrix(rbinom(n * p, 1, 0.5), n, p)
Z = rbinom(n, 1, 0.5)
Q = rbinom(n, 1, 0.5)
W = Q * Z
tau =  X[, 1] / 2
Y = rowSums(X[, 1:3]) + tau * W + Q + rnorm(n)

# Run outcome regression and extract smoother matrix
forest.Y = grf::regression_forest(X, Y)
Y.hat = predict(forest.Y)$predictions
outcome_smoother = grf::get_forest_weights(forest.Y)

# Run instrumental forest with external Y.hats
iv.forest = grf::instrumental_forest(X, Y, W, Z, Y.hat = Y.hat)

# Predict on out-of-bag training samples.
iv.pred = predict(iv.forest)$predictions

omega_if = get_outcome_weights(iv.forest, S = outcome_smoother)

# Observe that they perfectly replicate the original CLATEs
all.equal(as.numeric(omega_if$omega %*% Y),
          as.numeric(iv.pred))
```

---

get_outcome_weights.ivreg

Outcome weights for ivreg

Description

Post-estimation command to calculate the outcome weights for Instrumental-Variable Regression implemented via the ivreg command. Automatically takes sampling weights into account if used.

Usage

```
## S3 method for class 'ivreg'
get_outcome_weights(object, treat_pos = 2, ...)
```

Arguments

object	An object of class ivreg, i.e. the result of running ivreg.
treat_pos	The position of the treatment indicator in the model matrix. By default 2, which is appropriate for the case with an intercept and the (endogneous) treatment indicator as the first explanatory variable.
...	Pass potentially generic get_outcome_weights options.

Value

A vector of implied weights.

---

get_outcome_weights.lm

Outcome weights for the lm command

Description

Post-estimation command to calculate the outcome weights for linear regression implemented via the lm command.

Usage

```
## S3 method for class 'lm'
get_outcome_weights(object, treat_pos = 2, ...)
```

Arguments

object	An object of class lm, i.e. the result of running lm.
treat_pos	The position of the treatment indicator in the model matrix. By default 2, which is appropriate for the case with an intercept and the treatment indicator as the first explanatory variable.
...	Pass potentially generic get_outcome_weights options.

Value

get_outcome_weights object with omega containing weights and treat the treatment

get_outcome_weights.lm_robust

Outcome weights for the `lm_robust` *command*

Description

Post-estimation command to calculate the outcome weights for linear regression implemented via the `lm_robust` command.

Usage

```
## S3 method for class 'lm_robust'
get_outcome_weights(object, treat_pos = 2, ...)
```

Arguments

`object`	An object of class `lm_robust`, i.e. the result of running `lm_robust`.
`treat_pos`	The position of the treatment indicator in the model matrix. By default 2, which is appropriate for the case with an intercept and the treatment indicator as the first explanatory variable.
`...`	Pass potentially generic get_outcome_weights options.

Value

get_outcome_weights object with `omega` containing weights and `treat` the treatment

get_smoother_weights *Smoother weights method*

Description

This is a generic method for getting smoother weights. It calculates the smoother weights for objects created by other packages. See get_smoother_weight.<compatible_class> in the package documentation for compatible functions.

Usage

```
get_smoother_weights(object, ...)
```

Arguments

`object`	An object obtained from other packages.
`...`	Additional arguments specific to object class implementations. See the documentation for which objects require which additional arguments.

Value

An $N \times N$ smoother matrix.

`get_smoother_weights.cv.plasso` *Smoother weights for the* `plasso::cv.plasso` *function*

Description

Post-estimation function to extract smoother weights for a post-Lasso regression estimated via the `cv.plasso` function from the **plasso** package.

Usage

```
## S3 method for class 'cv.plasso'
get_smoother_weights(object, Xnew = NULL, ...)
```

Arguments

`object`	An object of class `cv.plasso`, i.e., the result of calling `plasso::cv.plasso()`.
`Xnew`	Optional covariate matrix of the test sample. If `NULL`, the smoother matrix for the training data is returned.
`...`	Additional arguments passed to get_smoother_weights.

Value

An $N \times N$ smoother matrix.

References

Knaus, M. C. (2024). Treatment effect estimators as weighted outcomes, https://arxiv.org/abs/2411.11559.

Examples

```
## Not run:
set.seed(123)
n <- 100
X <- matrix(rnorm(n * 3), ncol = 3)
Y <- 2 * X[, 1] + 3 * X[, 2] - 1 * X[, 3] + rnorm(n)

# Fit post-lasso model
object <- plasso::cv.plasso(x = X, y = Y)

# Get predictions
preds <- predict(object, newx = X, type = "response", s = "optimal", se_rule = 0)$plasso

# Get smoother matrix
```

```
S <- get_smoother_weights(object, Xnew = X)

# Check equivalence
all.equal(as.numeric(S %*% Y), as.numeric(preds))

## End(Not run)
```

`get_smoother_weights.drf` *Smoother weights for the* `drf::drf` *function*

Description

Post-estimation function to extract smoother weights for a distributional random forest model estimated via the `drf` function from the **drf** package.

Usage

```
## S3 method for class 'drf'
get_smoother_weights(object, Xnew = NULL, ...)
```

Arguments

`object`	An object of class `drf`, i.e., the result of calling `drf::drf()`.
`Xnew`	Optional covariate matrix of the test sample. If `NULL`, the smoother matrix for the training data is returned.
`...`	Additional arguments passed to get_smoother_weights.

Value

An $N \times N$ smoother matrix.

Examples

```
## Not run:
set.seed(123)
n <- 100
X <- matrix(rnorm(n * 3), ncol = 3)
Y <- 2 * X[, 1] + 3 * X[, 2] - 1 * X[, 3] + rnorm(n)

# Fit distributional forest
object <- drf::drf(X = X, Y = Y)

# Get predictions
preds <- predict(object, newdata = X, functional = "mean")[["mean"]]

# Get smoother matrix
S <- get_smoother_weights(object)
```

```
# Check equivalence
all.equal(as.numeric(S %*% Y), as.numeric(preds))

## End(Not run)
```

`get_smoother_weights.lm` *Smoother weights for the* `stats::lm` *function*

Description

Post-estimation function to extract smoother weights for a linear regression estimated via the `lm` function from the **stats** package.

Usage

```
## S3 method for class 'lm'
get_smoother_weights(object, X, Xnew, ...)
```

Arguments

`object`	An object of class `lm`, i.e., the result of calling `stats::lm()`.
`X`	Covariate matrix of the training sample.
`Xnew`	Optional covariate matrix of the test sample. If `NULL`, the smoother matrix for the training data is returned.
`...`	Additional arguments passed to get_smoother_weights.

Value

An $N \times N$ smoother matrix.

Examples

```
## Not run:
set.seed(123)
n <- 100
X <- matrix(rnorm(n * 3), ncol = 3)
Y <- 2 * X[, 1] + 3 * X[, 2] - 1 * X[, 3] + rnorm(n)

# Fit linear model
object <- lm(Y ~ 0 + X)

# Get fitted values
preds <- object$fitted.values

# Get smoother matrix
```

```
S <- get_smoother_weights(object, X = X, Xnew = X)

# Check equivalence
all.equal(as.numeric(S %*% Y), as.numeric(preds))

## End(Not run)
```

`get_smoother_weights.lm_robust` *Smoother weights for the* `estimatr::lm_robust` *function*

Description

Post-estimation function to extract smoother weights for a linear regression estimated via the `lm_robust` function from the **estimatr** package.

Usage

```
## S3 method for class 'lm_robust'
get_smoother_weights(object, X, Xnew, ...)
```

Arguments

`object`	An object of class `lm_robust`, i.e., the result of calling `estimatr::lm_robust()`.
`X`	Covariate matrix of the training sample.
`Xnew`	Optional covariate matrix of the test sample. If `NULL`, the smoother matrix for the training data is returned.
`...`	Additional arguments passed to get_smoother_weights.

Value

An $N \times N$ smoother matrix.

Examples

```
## Not run:
set.seed(123)
n <- 100
X <- matrix(rnorm(n * 3), ncol = 3)
Y <- 2 * X[, 1] + 3 * X[, 2] - 1 * X[, 3] + rnorm(n)

# Fit linear model
object <- estimatr::lm_robust(Y ~ 0 + X)

# Get fitted values
preds <- object$fitted.values
```

```
# Get smoother matrix
S <- get_smoother_weights(object, X = X, Xnew = X)

# Check equivalence
all.equal(as.numeric(S %*% Y), as.numeric(preds))

## End(Not run)
```

`get_smoother_weights.NuisanceParameters`
Smoother weights for outcome models

Description

Extract smoother matrices for nuisance parameter outcome models from a `NuisanceParameters` object created by `nuisance_parameters`.

Usage

```
## S3 method for class 'NuisanceParameters'
get_smoother_weights(object, NuPa, quiet = TRUE, subset = NULL, ...)
```

Arguments

`object`	A `NuisanceParameters` object created by `nuisance_parameters`.
`NuPa`	Character vector specifying which nuisance parameters to extract smoothers for. Available outcome nuisance parameters are `c("Y.hat", "Y.hat.d", "Y.hat.z")`.
`quiet`	If `FALSE`, the method currently computed is printed to the console.
`subset`	Logical vector indicating which observations to use when determining ensemble weights. If not provided, all observations are used.
`...`	Additional arguments.

Value

A list of matrices of dimension `nrow(X)` × `nrow(X)` containing ensemble smoother matrices

`get_smoother_weights.ranger` *Smoother weights for the* `ranger::ranger` *function*

Description

Post-estimation function to extract smoother weights for a random forest model estimated via the `ranger` function from the **ranger** package.

Usage

```
## S3 method for class 'ranger'
get_smoother_weights(object, X = NULL, Xnew = NULL, ...)
```

Arguments

`object`	An object of class `ranger`, i.e., the result of calling `ranger::ranger()`.
`X`	Optional covariate matrix of the training sample; used if not stored in the fitted object.
`Xnew`	Covariate matrix of test sample. If not provided, prediction is done for the training sample.
`...`	Additional arguments passed to get_smoother_weights.

Value

An $N \times N$ smoother matrix.

Examples

```
## Not run:
set.seed(123)
n <- 100
X <- matrix(rnorm(n * 3), ncol = 3, dimnames = list(NULL, 1:3))
Y <- 2 * X[, 1] + 3 * X[, 2] - 1 * X[, 3] + rnorm(n)

# Fit random forest
object <- ranger::ranger(x = X, y = Y, keep.inbag = TRUE)

# Get predictions
preds <- predict(object, data = X)$predictions

# Get smoother matrix
S <- get_smoother_weights(object, X = X)

# Check equivalence
all.equal(as.numeric(S %*% Y), as.numeric(preds))

## End(Not run)
```

`get_smoother_weights.regression_forest` *Smoother weights for the* `grf::regression_forest` *function*

Description

Post-estimation function to extract smoother weights for a random forest model estimated via the `regression_forest` function from the **grf** package.

Usage

```
## S3 method for class 'regression_forest'
get_smoother_weights(object, Xnew = NULL, ...)
```

Arguments

`object`	An object of class `regression_forest`, i.e., the result of calling `grf::regression_forest()`.
`Xnew`	Optional covariate matrix of the test sample. If `NULL`, the smoother matrix for the training data is returned.
`...`	Additional arguments passed to get_smoother_weights.

Value

An $N \times N$ smoother matrix.

References

Athey, S., Tibshirani, J., & Wager, S. (2019). Generalized random forest. The Annals of Statistics, 47(2), 1148-1178.

Knaus, M. C. (2024). Treatment effect estimators as weighted outcomes, https://arxiv.org/abs/2411.11559.

Examples

```
## Not run:
set.seed(123)
n <- 100
X <- matrix(rnorm(n * 3), ncol = 3)
Y <- 2 * X[, 1] + 3 * X[, 2] - 1 * X[, 3] + rnorm(n)

# Fit regression forest
object <- grf::regression_forest(X = X, Y = Y)

# Get predictions
preds <- predict(object, newdata = X)$predictions

# Get smoother matrix
S <- get_smoother_weights(object)
```

```
# Check equivalence
all.equal(as.numeric(S %*% Y), as.numeric(preds))

## End(Not run)
```

`get_smoother_weights.rlasso` *Smoother weights for the* `hdm::rlasso` *function*

Description

Post-estimation function to extract smoother weights for a post-Lasso regression estimated via the `rlasso` function from the **hdm** package.

Usage

```
## S3 method for class 'rlasso'
get_smoother_weights(object, Xnew = NULL, ...)
```

Arguments

`object`	An object of class `rlasso`, i.e., the result of calling `hdm::rlasso()`.
`Xnew`	Optional covariate matrix of the test sample. If `NULL`, the smoother matrix for the training data is returned.
`...`	Additional arguments passed to get_smoother_weights.

Value

An $N \times N$ smoother matrix.

References

Knaus, M. C. (2024). Treatment effect estimators as weighted outcomes, https://arxiv.org/abs/2411.11559.

Examples

```
## Not run:
set.seed(123)
n <- 100
X <- matrix(rnorm(n * 3), ncol = 3)
Y <- 2 * X[, 1] + 3 * X[, 2] - 1 * X[, 3] + rnorm(n)

# Fit post-lasso model
object <- hdm::rlasso(x = X, y = Y, post = TRUE)
```

```
# Get predictions
preds <- predict(object, newdata = X)

# Get smoother matrix
S <- get_smoother_weights(object, Xnew = X)

# Check equivalence
all.equal(as.numeric(S %*% Y), as.numeric(preds))

## End(Not run)
```

`get_smoother_weights.xgb.Booster` *Smoother weights for the* `xgboost::xgb.train` *function*

Description

Post-estimation function to extract smoother weights for an XGBoost model estimated via the `xgb.train` function from the **xgboost** package.

Usage

```
## S3 method for class 'xgb.Booster'
get_smoother_weights(object, X = NULL, Y = NULL, Xnew = NULL, ...)
```

Arguments

`object`	An object of class `xgb.Booster`, i.e., the result of calling `xgboost::xgb.train()`.
`X`	Covariate matrix of the training sample.
`Y`	Vector of outcomes of the training sample.
`Xnew`	Optional covariate matrix of the test sample. If `NULL`, the smoother matrix for the training data is returned.
`...`	Additional arguments passed to get_smoother_weights.

Value

An $N \times N$ smoother matrix.

Note

For smoother extraction, the XGBoost model must be trained with: `alpha = 0`, `subsample = 1`, `max_delta_step = 0`.

Examples

```
## Not run:
set.seed(123)
n <- 100
X <- matrix(rnorm(n * 3), ncol = 3)
Y <- 2 * X[, 1] + 3 * X[, 2] - 1 * X[, 3] + rnorm(n)

# Set the parameters
nrounds <- 100
params <- list(alpha = 0, subsample = 1, max_delta_step = 0)

# Prepare the data
d_train <- xgboost::xgb.DMatrix(data = as.matrix(X), label = Y)
d_test <- xgboost::xgb.DMatrix(data = as.matrix(X))

# Fit XGBoost
object <- xgboost::xgb.train(data = d_train, nrounds = nrounds, params = params)

# Get predictions
preds <- predict(object, newdata = d_test)

# Get smoother matrix
S <- get_smoother_weights(object, X, Y)

# Check equivalence
all.equal(as.numeric(S %*% Y), as.numeric(preds))

## End(Not run)
```

NuPa_honest_forest *Nuisance parameter estimation via honest random forest*

Description

This function estimates different nuisance parameters using the honest random forest implementation of the 'grf' package

Usage

```
NuPa_honest_forest(
  NuPa = c("Y.hat", "Y.hat.d", "Y.hat.z", "D.hat", "D.hat.z", "Z.hat"),
  X,
  Y = NULL,
  D = NULL,
  Z = NULL,
  n_cf_folds = 5,
  n_reps = 1,
  cluster = NULL,
```

```
  progress = FALSE,
  ...
)
```

Arguments

NuPa	String vector specifying the nuisance parameters to be estimated. Currently supported: c("Y.hat","Y.hat.d","Y.hat.z","D.hat","D.hat.z","Z.hat")
X	Covariate matrix with N rows and p columns.
Y	Optional numeric vector containing the outcome variable.
D	Optional binary treatment variable.
Z	Optional binary instrumental variable.
n_cf_folds	Number of cross-fitting folds. Default is 5.
n_reps	Number of repetitions of cross-fitting. Default is 1.
cluster	Optional vector of cluster variable if cross-fitting should account for clusters.
progress	If TRUE, progress of nuisance parameter estimation reported.
...	Options passed to the regression_forest.

Value

List of two lists.

- predictions contains the requested nuisance parameters
- smoothers contains the smoother matrices of requested outcome nuisance parameters
- cf_mat Array of dimension n_reps x N x n_cf_folds storing indicators representing the folds used in estimation.

References

Wager, S., & Athey, S. (2018). Estimation and inference of heterogeneous treatment effects using random forests. Journal of the American Statistical Association, 113(523), 1228-1242.

---

pive_weight_maker *Outcome weights maker for pseudo-IV estimators.*

Description

This is a generic function taking pseudo-instrument, pseudo-treatment and the transformation matrix as inputs and returning outcome weights

Usage

```
pive_weight_maker(Z.tilde, D.tilde, T_mat)
```

Arguments

Z.tilde	Numeric vector of pseudo-instrument outcomes.
D.tilde	Numeric vector of pseudo-treatment.
T_mat	Transformation matrix

Value

A vector of outcome weights.

References

Knaus, M. C. (2024). Treatment effect estimators as weighted outcomes, soon on 'arXiv'.

---

plot.dml_with_smoother

plot method for class dml_with_smoother

Description

plot method for class dml_with_smoother

Usage

```
## S3 method for class 'dml_with_smoother'
plot(x, ..., alpha = 0.05, contrast = FALSE)
```

Arguments

x	Object of class dml_with_smoother.
...	Pass generic plot options.
alpha	Significance level for confidence intervals (default 0.05).
contrast	Shows the differences between the coefficients.

Value

ggplot with point estimates and confidence intervals.

`prep_cf_mat` *Creates matrix of binary cross-fitting fold indicators (N x # cross-folds)*

Description

Creates matrix of binary cross-fitting fold indicators (N x # cross-folds)

Usage

```
prep_cf_mat(n, cf, w_mat = NULL, cl = NULL)
```

Arguments

`n`	Number of observations.
`cf`	Number of cross-fitting folds.
`w_mat`	Optional logical matrix of treatment indicators (N x T+1). If specified, cross-fitting folds will preserve the treatment ratios from full sample.
`cl`	Optional vector of cluster variable if cross-fitting should account for clusters.

Value

Logical matrix of cross-fitting folds (N x # folds).

`standardized_mean_differences` *Calls C++ implementation to calculate standardized mean differences.*

Description

Calculates standardized mean differences between treated and controls and towards target means for an outcome weights matrix with potentially many rows like for CATEs.

Usage

```
standardized_mean_differences(X, treat, omega, target = NULL)
```

Arguments

`X`	Covariate matrix with N rows and p columns.
`treat`	Binary treatment variable.
`omega`	Outcome weights matrix with dimension number of weight vectors for which balancing should be checked x number of training units.
`target`	Optional matrix with dimension number of weight vectors for which balancing should be checked x p indicating the target values the covariates should be balanced towards. If NULL, average of X used as target of ATE.

Value

3D-array of dimension p x 6 x number of weight vectors for which balancing should be checked where the second dimension provides the following quantities:

- "Mean 0": The weighted control mean
- "Mean 1": The weighted treated mean
- "SMD balancing": Standardized mean differences for covariate balancing (Mean 1 - Mean 0) / sd(X)
- "SMD targeting 0": Standardized mean difference to assess targeting of control (Mean 0 - target) / sd(X)
- "SMD targeting 1": Standardized mean difference to assess targeting of treated (Mean 1 - target) / sd(X)

References

Rosenbaum, P. R., & Rubin, D. B. (1984). Reducing bias in observational studies using subclassification on the propensity score. Journal of the American Statistical Association, 79 (387), 516–524.

`summary.dml_with_smoother`

`summary` *method for class* `dml_with_smoother`

Description

`summary` method for class `dml_with_smoother`

Usage

```
## S3 method for class 'dml_with_smoother'
summary(object, contrast = FALSE, quiet = FALSE, ...)
```

Arguments

`object`	Object of class `dml_with_smoother`.
`contrast`	Tests the differences between the coefficients.
`quiet`	If TRUE, results are passed but not printed.
`...`	further arguments passed to `printCoefmat`

Value

Invisible matrix with estimator(s) in the rows and c("Estimate","SE","t","p") in the columns.

summary.get_outcome_weights
summary *method for class* outcome_weights

Description

Calculates several summary measures of potentially many outcome weights.

Usage

```
## S3 method for class 'get_outcome_weights'
summary(object, quiet = FALSE, digits = 4, epsilon = 1e-04, ...)
```

Arguments

object	get_outcome_weights object.
quiet	If TRUE, results are passed but not printed.
digits	Number of digits to be displayed. Default 4.
epsilon	Threshold below which in absolute values non-zero but small values should be displayed as < ...
...	further arguments passed to printCoefmat

Value

3D-array of dimension

- c("Control","Treated") x
- number of point estimates x
- c("Minimum weight","Maximum weight","% Negative","Sum largest 10%","Sum of weights","Sum of absolute weights")

summary.standardized_mean_differences
summary *method for class* standardized_mean_differences

Description

Calls a C++ function to quickly summarize potentially many standardized mean differences.

Usage

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

Arguments

`object`	Object of class `standardized_mean_differences`.
`...`	further arguments passed to summary method.

Value

3D-array of dimension

- c("Maximum absolute SMD","Mean absolute SMD","Median absolute SMD", / % of absolute SMD > 20", "# / % of absolute SMD > 10","# / % of absolute SMD > 5") x
- c("Balancing","Targeting") x
- number of weight vectors for which balancing should be checked

References

Rosenbaum, P. R., & Rubin, D. B. (1984). Reducing bias in observational studies using subclassification on the propensity score. Journal of the American Statistical Association, 79 (387), 516–524.

Index

average_treatment_effect, *5*, *9*

causal_forest, *5*

dml_with_smoother, 2, *7*, *24*, *26*

get_outcome_weights, 4, *5*, *7–12*, *27*
get_outcome_weights.causal_forest, 5
get_outcome_weights.dml_with_smoother, 7
get_outcome_weights.DoubleML, 7
get_outcome_weights.instrumental_forest, 9
get_outcome_weights.ivreg, 10
get_outcome_weights.lm, 11
get_outcome_weights.lm_robust, 12
get_smoother_weights, 12, *13–16*, *18–21*
get_smoother_weights.cv.plasso, 13
get_smoother_weights.drf, 14
get_smoother_weights.lm, 15
get_smoother_weights.lm_robust, 16
get_smoother_weights.NuisanceParameters, 17
get_smoother_weights.ranger, 18
get_smoother_weights.regression_forest, 19
get_smoother_weights.rlasso, 20
get_smoother_weights.xgb.Booster, 21

instrumental_forest, *5*, *9*
ivreg, *10*, *11*

lm, *11*
lm_robust, *12*

nuisance_parameters, *17*
NuPa_honest_forest, 22

pive_weight_maker, 23
plot, *24*
plot.dml_with_smoother, 24
predict.causal_forest, *5*
predict.instrumental_forest, *9*
prep_cf_mat, 25

regression_forest, *5*, *9*, *23*

standardized_mean_differences, 25, *27*, *28*
summary.dml_with_smoother, 26
summary.get_outcome_weights, 27
summary.standardized_mean_differences, 27