Package ‘ddml’

July 22, 2025

Title Double/Debiased Machine Learning

Version 0.3.0

Date 2024-10-02

Description Estimate common causal parameters using double/debiased machine learning as proposed by Chernozhukov et al. (2018) <doi:10.1111/ectj.12097>. 'ddml' simplifies estimation based on (short-)stacking as discussed in Ahrens et al. (2024) <doi:10.1177/1536867X241233641>, which leverages multiple base learners to increase robustness to the underlying data generating process.

License GPL (>= 3)

URL https://github.com/thomaswiemann/ddml, https://thomaswiemann.com/ddml/

BugReports https://github.com/thomaswiemann/ddml/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 3.6)

Imports methods, stats, AER, MASS, Matrix, nnls, quadprog, glmnet, ranger, xgboost

Suggests sandwich, covr, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Achim Ahrens [aut],
Christian B Hansen [aut],
Mark E Schaffer [aut],
Thomas Wiemann [aut, cre]

Maintainer Thomas Wiemann <wiemann@uchicago.edu>

Repository CRAN

Date/Publication 2024-10-02 20:20:18 UTC

Contents

`AE98` *Random subsample from the data of Angrist & Evans (1991).*

Description

Random subsample from the data of Angrist & Evans (1991).

Usage

```
AE98
```

Format

A data frame with 5,000 rows and 13 variables.

worked Indicator equal to 1 if the mother is employed.

weeksw Number of weeks of employment.

hoursw Hours worked per week.

morekids Indicator equal to 1 if the mother has more than 2 kids.

samesex Indicator equal to 1 if the first two children are of the same sex.

age Age in years.

agefst Age in years at birth of the first child.

black Indicator equal to 1 if the mother is black.

hisp Indicator equal to 1 if the mother is Hispanic.

othrace Indicator equal to 1 if the mother is neither black nor Hispanic.

educ Years of education.

boy1st Indicator equal to 1 if the first child is male.

boy2nd Indicator equal to 1 if the second child is male.

Source

https://dataverse.harvard.edu/dataset.xhtml?persistentId=hdl:1902.1/11288

References

Angrist J, Evans W (1998). "Children and Their Parents' Labor Supply: Evidence from Exogenous Variation in Family Size." American Economic Review, 88(3), 450-477.

`crosspred` *Cross-Predictions using Stacking.*

Description

Cross-predictions using stacking.

Usage

```
crosspred(
  y,
  X,
  Z = NULL,
  learners,
  sample_folds = 2,
  ensemble_type = "average",
  cv_folds = 5,
  custom_ensemble_weights = NULL,
  compute_insample_predictions = FALSE,
  compute_predictions_bylearner = FALSE,
  subsamples = NULL,
  cv_subsamples_list = NULL,
  silent = FALSE,
  progress = NULL,
  auxiliary_X = NULL
)
```

Arguments

y
: The outcome variable.

X
: A (sparse) matrix of predictive variables.

Z
: Optional additional (sparse) matrix of predictive variables.

learners
: May take one of two forms, depending on whether a single learner or stacking with multiple learners is used for estimation of the predictor. If a single learner is used, `learners` is a list with two named elements:

  - `what` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
  - `args` Optional arguments to be passed to `what`.

  If stacking with multiple learners is used, `learners` is a list of lists, each containing four named elements:

  - `fun` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
  - `args` Optional arguments to be passed to `fun`.
  - `assign_X` An optional vector of column indices corresponding to predictive variables in `X` that are passed to the base learner.
  - `assign_Z` An optional vector of column indices corresponding to predictive in `Z` that are passed to the base learner.

  Omission of the `args` element results in default arguments being used in `fun`. Omission of `assign_X` (and/or `assign_Z`) results in inclusion of all variables in `X` (and/or `Z`).

sample_folds
: Number of cross-fitting folds.

ensemble_type
: Ensemble method to combine base learners into final estimate of the conditional expectation functions. Possible values are:

  - `"nnls"` Non-negative least squares.
  - `"nnls1"` Non-negative least squares with the constraint that all weights sum to one.
  - `"singlebest"` Select base learner with minimum MSPE.
  - `"ols"` Ordinary least squares.
  - `"average"` Simple average over base learners.

  Multiple ensemble types may be passed as a vector of strings.

cv_folds
: Number of folds used for cross-validation in ensemble construction.

custom_ensemble_weights
: A numerical matrix with user-specified ensemble weights. Each column corresponds to a custom ensemble specification, each row corresponds to a base learner in `learners` (in chronological order). Optional column names are used to name the estimation results corresponding the custom ensemble specification.

compute_insample_predictions
: Indicator equal to 1 if in-sample predictions should also be computed.

compute_predictions_bylearner
: Indicator equal to 1 if in-sample predictions should also be computed for each learner (rather than the entire ensemble).

`subsamples` List of vectors with sample indices for cross-fitting.

`cv_subsamples_list` List of lists, each corresponding to a subsample containing vectors with subsample indices for cross-validation.

`silent` Boolean to silence estimation updates.

`progress` String to print before learner and cv fold progress.

`auxiliary_X` An optional list of matrices of length `sample_folds`, each containing additional observations to calculate predictions for.

Value

`crosspred` returns a list containing the following components:

`oos_fitted` A matrix of out-of-sample predictions, each column corresponding to an ensemble type (in chronological order).

`weights` An array, providing the weight assigned to each base learner (in chronological order) by the ensemble procedures.

`is_fitted` When `compute_insample_predictions = T`. a list of matrices with in-sample predictions by sample fold.

`auxiliary_fitted` When `auxiliary_X` is not `NULL`, a list of matrices with additional predictions.

`oos_fitted_bylearner` When `compute_predictions_bylearner = T`, a matrix of out-of-sample predictions, each column corresponding to a base learner (in chronological order).

`is_fitted_bylearner` When `compute_insample_predictions = T` and `compute_predictions_bylearner = T`, a list of matrices with in-sample predictions by sample fold.

`auxiliary_fitted_bylearner` When `auxiliary_X` is not `NULL` and `compute_predictions_bylearner = T`, a list of matrices with additional predictions for each learner.

References

Ahrens A, Hansen C B, Schaffer M E, Wiemann T (2023). "ddml: Double/debiased machine learning in Stata." https://arxiv.org/abs/2301.09397

Wolpert D H (1992). "Stacked generalization." Neural Networks, 5(2), 241-259.

See Also

Other utilities: `crossval()`, `shortstacking()`

Examples

```
# Construct variables from the included Angrist & Evans (1998) data
y = AE98[, "worked"]
X = AE98[, c("morekids", "age","agefst","black","hisp","othrace","educ")]

# Compute cross-predictions using stacking with base learners ols and lasso.
#     Two stacking approaches are simultaneously computed: Equally
#     weighted (ensemble_type = "average") and MSPE-minimizing with weights
#     in the unit simplex (ensemble_type = "nnls1"). Predictions for each
```

```
#     learner are also calculated.
crosspred_res <- crosspred(y, X,
                           learners = list(list(fun = ols),
                                           list(fun = mdl_glmnet)),
                           ensemble_type = c("average",
                                             "nnls1",
                                             "singlebest"),
                           compute_predictions_bylearner = TRUE,
                           sample_folds = 2,
                           cv_folds = 2,
                           silent = TRUE)
dim(crosspred_res$oos_fitted) # = length(y) by length(ensemble_type)
dim(crosspred_res$oos_fitted_bylearner) # = length(y) by length(learners)
```

`crossval` *Estimator of the Mean Squared Prediction Error using Cross-Validation.*

Description

Estimator of the mean squared prediction error of different learners using cross-validation.

Usage

```
crossval(
  y,
  X,
  Z = NULL,
  learners,
  cv_folds = 5,
  cv_subsamples = NULL,
  silent = FALSE,
  progress = NULL
)
```

Arguments

`y` The outcome variable.

`X` A (sparse) matrix of predictive variables.

`Z` Optional additional (sparse) matrix of predictive variables.

`learners` `learners` is a list of lists, each containing four named elements:

- `fun` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
- `args` Optional arguments to be passed to `fun`.
- `assign_X` An optional vector of column indices corresponding to variables in `X` that are passed to the base learner.

- `assign_Z` An optional vector of column indices corresponding to variables in `Z` that are passed to the base learner.

Omission of the `args` element results in default arguments being used in `fun`. Omission of `assign_X` (and/or `assign_Z`) results in inclusion of all predictive variables in `X` (and/or `Z`).

`cv_folds`	Number of folds used for cross-validation.
`cv_subsamples`	List of vectors with sample indices for cross-validation.
`silent`	Boolean to silence estimation updates.
`progress`	String to print before learner and cv fold progress.

Value

`crossval` returns a list containing the following components:

`mspe` A vector of MSPE estimates, each corresponding to a base learners (in chronological order).

`oos_resid` A matrix of out-of-sample prediction errors, each column corresponding to a base learners (in chronological order).

`cv_subsamples` Pass-through of `cv_subsamples`. See above.

See Also

Other utilities: `crosspred()`, `shortstacking()`

Examples

```
# Construct variables from the included Angrist & Evans (1998) data
y = AE98[, "worked"]
X = AE98[, c("morekids", "age","agefst","black","hisp","othrace","educ")]

# Compare ols, lasso, and ridge using 4-fold cross-validation
cv_res <- crossval(y, X,
                   learners = list(list(fun = ols),
                                   list(fun = mdl_glmnet),
                                   list(fun = mdl_glmnet,
                                        args = list(alpha = 0))),
                   cv_folds = 4,
                   silent = TRUE)
cv_res$mspe
```

`ddml` *ddml: Double/Debiased Machine Learning in R*

Description

Estimate common causal parameters using double/debiased machine learning as proposed by Chernozhukov et al. (2018). 'ddml' simplifies estimation based on (short-)stacking, which leverages multiple base learners to increase robustness to the underlying data generating process.

References

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

---

ddml_ate *Estimators of Average Treatment Effects.*

Description

Estimators of the average treatment effect and the average treatment effect on the treated.

Usage

```
ddml_ate(
  y,
  D,
  X,
  learners,
  learners_DX = learners,
  sample_folds = 10,
  ensemble_type = "nnls",
  shortstack = FALSE,
  cv_folds = 10,
  custom_ensemble_weights = NULL,
  custom_ensemble_weights_DX = custom_ensemble_weights,
  cluster_variable = seq_along(y),
  subsamples_byD = NULL,
  cv_subsamples_byD = NULL,
  trim = 0.01,
  silent = FALSE
)

ddml_att(
  y,
  D,
  X,
  learners,
  learners_DX = learners,
  sample_folds = 10,
  ensemble_type = "nnls",
  shortstack = FALSE,
  cv_folds = 10,
  custom_ensemble_weights = NULL,
  custom_ensemble_weights_DX = custom_ensemble_weights,
  cluster_variable = seq_along(y),
```

```
  subsamples_byD = NULL,
  cv_subsamples_byD = NULL,
  trim = 0.01,
  silent = FALSE
)
```

Arguments

`y` The outcome variable.

`D` The binary endogenous variable of interest.

`X` A (sparse) matrix of control variables.

`learners` May take one of two forms, depending on whether a single learner or stacking with multiple learners is used for estimation of the conditional expectation functions. If a single learner is used, `learners` is a list with two named elements:

- `what` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
- `args` Optional arguments to be passed to `what`.

If stacking with multiple learners is used, `learners` is a list of lists, each containing four named elements:

- `fun` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
- `args` Optional arguments to be passed to `fun`.
- `assign_X` An optional vector of column indices corresponding to control variables in `X` that are passed to the base learner.

Omission of the `args` element results in default arguments being used in `fun`. Omission of `assign_X` results in inclusion of all variables in `X`.

`learners_DX` Optional argument to allow for different estimators of $E[D|X]$. Setup is identical to `learners`.

`sample_folds` Number of cross-fitting folds.

`ensemble_type` Ensemble method to combine base learners into final estimate of the conditional expectation functions. Possible values are:

- `"nnls"` Non-negative least squares.
- `"nnls1"` Non-negative least squares with the constraint that all weights sum to one.
- `"singlebest"` Select base learner with minimum MSPE.
- `"ols"` Ordinary least squares.
- `"average"` Simple average over base learners.

Multiple ensemble types may be passed as a vector of strings.

`shortstack` Boolean to use short-stacking.

`cv_folds` Number of folds used for cross-validation in ensemble construction.

`custom_ensemble_weights`
A numerical matrix with user-specified ensemble weights. Each column corresponds to a custom ensemble specification, each row corresponds to a base learner in `learners` (in chronological order). Optional column names are used to name the estimation results corresponding the custom ensemble specification.

`custom_ensemble_weights_DX`
: Optional argument to allow for different custom ensemble weights for `learners_DX`. Setup is identical to `custom_ensemble_weights`. Note: `custom_ensemble_weights` and `custom_ensemble_weights_DX` must have the same number of columns.

`cluster_variable`
: A vector of cluster indices.

`subsamples_byD`
: List of two lists corresponding to the two treatment levels. Each list contains vectors with sample indices for cross-fitting.

`cv_subsamples_byD`
: List of two lists, each corresponding to one of the two treatment levels. Each of the two lists contains lists, each corresponding to a subsample and contains vectors with subsample indices for cross-validation.

`trim`
: Number in (0, 1) for trimming the estimated propensity scores at `trim` and `1-trim`.

`silent`
: Boolean to silence estimation updates.

Details

`ddml_ate` and `ddml_att` provide double/debiased machine learning estimators for the average treatment effect and the average treatment effect on the treated, respectively, in the interactive model given by

$Y = g_0(D, X) + U,$

where (Y, D, X, U) is a random vector such that $\operatorname{supp} D = \{0, 1\}$, $E[U|D, X] = 0$, and $\Pr(D = 1|X) \in (0, 1)$ with probability 1, and g_0 is an unknown nuisance function.

In this model, the average treatment effect is defined as

$\theta_0^{\textrm{ATE}} \equiv E[g_0(1, X) - g_0(0, X)].$

and the average treatment effect on the treated is defined as

$\theta_0^{\textrm{ATT}} \equiv E[g_0(1, X) - g_0(0, X)|D = 1].$

Value

`ddml_ate` and `ddml_att` return an object of S3 class `ddml_ate` and `ddml_att`, respectively. An object of class `ddml_ate` or `ddml_att` is a list containing the following components:

`ate` / `att` A vector with the average treatment effect / average treatment effect on the treated estimates.

`weights` A list of matrices, providing the weight assigned to each base learner (in chronological order) by the ensemble procedure.

`mspe` A list of matrices, providing the MSPE of each base learner (in chronological order) computed by the cross-validation step in the ensemble construction.

`psi_a`, `psi_b` Matrices needed for the computation of scores. Used in `summary.ddml_ate()` or `summary.ddml_att()`.

`oos_pred` List of matrices, providing the reduced form predicted values.

`learners`,`learners_DX`,`cluster_variable`, `subsamples_D0`,`subsamples_D1`, `cv_subsamples_list_D0`,`cv_subsamples` Pass-through of selected user-provided arguments. See above.

References

Ahrens A, Hansen C B, Schaffer M E, Wiemann T (2023). "ddml: Double/debiased machine learning in Stata." https://arxiv.org/abs/2301.09397

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

Wolpert D H (1992). "Stacked generalization." Neural Networks, 5(2), 241-259.

See Also

summary.ddml_ate(), summary.ddml_att()

Other ddml: ddml_fpliv(), ddml_late(), ddml_pliv(), ddml_plm()

Examples

```
# Construct variables from the included Angrist & Evans (1998) data
y = AE98[, "worked"]
D = AE98[, "morekids"]
X = AE98[, c("age","agefst","black","hisp","othrace","educ")]

# Estimate the average treatment effect using a single base learner, ridge.
ate_fit <- ddml_ate(y, D, X,
                    learners = list(what = mdl_glmnet,
                                    args = list(alpha = 0)),
                    sample_folds = 2,
                    silent = TRUE)
summary(ate_fit)

# Estimate the average treatment effect using short-stacking with base
#     learners ols, lasso, and ridge. We can also use custom_ensemble_weights
#     to estimate the ATE using every individual base learner.
weights_everylearner <- diag(1, 3)
colnames(weights_everylearner) <- c("mdl:ols", "mdl:lasso", "mdl:ridge")
ate_fit <- ddml_ate(y, D, X,
                    learners = list(list(fun = ols),
                                    list(fun = mdl_glmnet),
                                    list(fun = mdl_glmnet,
                                         args = list(alpha = 0))),
                    ensemble_type = 'nnls',
                    custom_ensemble_weights = weights_everylearner,
                    shortstack = TRUE,
                    sample_folds = 2,
                    silent = TRUE)
summary(ate_fit)
```

ddml_fpliv *Estimator for the Flexible Partially Linear IV Model.*

Description

Estimator for the flexible partially linear IV model.

Usage

```
ddml_fpliv(
  y,
  D,
  Z,
  X,
  learners,
  learners_DXZ = learners,
  learners_DX = learners,
  sample_folds = 10,
  ensemble_type = "nnls",
  shortstack = FALSE,
  cv_folds = 10,
  enforce_LIE = TRUE,
  custom_ensemble_weights = NULL,
  custom_ensemble_weights_DXZ = custom_ensemble_weights,
  custom_ensemble_weights_DX = custom_ensemble_weights,
  cluster_variable = seq_along(y),
  subsamples = NULL,
  cv_subsamples_list = NULL,
  silent = FALSE
)
```

Arguments

`y` The outcome variable.

`D` A matrix of endogenous variables.

`Z` A (sparse) matrix of instruments.

`X` A (sparse) matrix of control variables.

`learners` May take one of two forms, depending on whether a single learner or stacking with multiple learners is used for estimation of the conditional expectation functions. If a single learner is used, `learners` is a list with two named elements:

- `what` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
- `args` Optional arguments to be passed to `what`.

If stacking with multiple learners is used, `learners` is a list of lists, each containing four named elements:

- `fun` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
- `args` Optional arguments to be passed to `fun`.
- `assign_X` An optional vector of column indices corresponding to control variables in `X` that are passed to the base learner.
- `assign_Z` An optional vector of column indices corresponding to instruments in `Z` that are passed to the base learner.

Omission of the `args` element results in default arguments being used in `fun`. Omission of `assign_X` (and/or `assign_Z`) results in inclusion of all variables in `X` (and/or `Z`).

`learners_DXZ`, `learners_DX`
: Optional arguments to allow for different estimators of $E[D|X,Z]$, $E[D|X]$. Setup is identical to `learners`.

`sample_folds`
: Number of cross-fitting folds.

`ensemble_type`
: Ensemble method to combine base learners into final estimate of the conditional expectation functions. Possible values are:

- `"nnls"` Non-negative least squares.
- `"nnls1"` Non-negative least squares with the constraint that all weights sum to one.
- `"singlebest"` Select base learner with minimum MSPE.
- `"ols"` Ordinary least squares.
- `"average"` Simple average over base learners.

Multiple ensemble types may be passed as a vector of strings.

`shortstack`
: Boolean to use short-stacking.

`cv_folds`
: Number of folds used for cross-validation in ensemble construction.

`enforce_LIE`
: Indicator equal to 1 if the law of iterated expectations is enforced in the first stage.

`custom_ensemble_weights`
: A numerical matrix with user-specified ensemble weights. Each column corresponds to a custom ensemble specification, each row corresponds to a base learner in `learners` (in chronological order). Optional column names are used to name the estimation results corresponding the custom ensemble specification.

`custom_ensemble_weights_DXZ`, `custom_ensemble_weights_DX`
: Optional arguments to allow for different custom ensemble weights for `learners_DXZ`,`learners_DX`. Setup is identical to `custom_ensemble_weights`. Note: `custom_ensemble_weights` and `custom_ensemble_weights_DXZ`,`custom_ensemble_weights_DX` must have the same number of columns.

`cluster_variable`
: A vector of cluster indices.

`subsamples`
: List of vectors with sample indices for cross-fitting.

`cv_subsamples_list`
: List of lists, each corresponding to a subsample containing vectors with subsample indices for cross-validation.

`silent`
: Boolean to silence estimation updates.

Details

ddml_fpliv provides a double/debiased machine learning estimator for the parameter of interest θ_0 in the partially linear IV model given by

$Y = \theta_0 D + g_0(X) + U,$

where (Y, D, X, Z, U) is a random vector such that $E[U|X, Z] = 0$ and $E[Var(E[D|X, Z]|X)] \neq 0$, and g_0 is an unknown nuisance function.

Value

ddml_fpliv returns an object of S3 class ddml_fpliv. An object of class ddml_fpliv is a list containing the following components:

coef A vector with the θ_0 estimates.

weights A list of matrices, providing the weight assigned to each base learner (in chronological order) by the ensemble procedure.

mspe A list of matrices, providing the MSPE of each base learner (in chronological order) computed by the cross-validation step in the ensemble construction.

iv_fit Object of class ivreg from the IV regression of $Y - \hat{E}[Y|X]$ on $D - \hat{E}[D|X]$ using $\hat{E}[D|X, Z] - \hat{E}[D|X]$ as the instrument.

learners,learners_DX,learners_DXZ, cluster_variable,subsamples, cv_subsamples_list,ensemble_type
Pass-through of selected user-provided arguments. See above.

References

Ahrens A, Hansen C B, Schaffer M E, Wiemann T (2023). "ddml: Double/debiased machine learning in Stata." https://arxiv.org/abs/2301.09397

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

Wolpert D H (1992). "Stacked generalization." Neural Networks, 5(2), 241-259.

See Also

summary.ddml_fpliv(), AER::ivreg()

Other ddml: ddml_ate(), ddml_late(), ddml_pliv(), ddml_plm()

Examples

```
# Construct variables from the included Angrist & Evans (1998) data
y = AE98[, "worked"]
D = AE98[, "morekids"]
Z = AE98[, "samesex", drop = FALSE]
X = AE98[, c("age","agefst","black","hisp","othrace","educ")]

# Estimate the partially linear IV model using a single base learner: Ridge.
fpliv_fit <- ddml_fpliv(y, D, Z, X,
                        learners = list(what = mdl_glmnet,
```

```
                                args = list(alpha = 0)),
                        sample_folds = 2,
                        silent = TRUE)
summary(fpliv_fit)
```

ddml_late — *Estimator of the Local Average Treatment Effect.*

Description

Estimator of the local average treatment effect.

Usage

```
ddml_late(
  y,
  D,
  Z,
  X,
  learners,
  learners_DXZ = learners,
  learners_ZX = learners,
  sample_folds = 10,
  ensemble_type = "nnls",
  shortstack = FALSE,
  cv_folds = 10,
  custom_ensemble_weights = NULL,
  custom_ensemble_weights_DXZ = custom_ensemble_weights,
  custom_ensemble_weights_ZX = custom_ensemble_weights,
  cluster_variable = seq_along(y),
  subsamples_byZ = NULL,
  cv_subsamples_byZ = NULL,
  trim = 0.01,
  silent = FALSE
)
```

Arguments

`y`	The outcome variable.
`D`	The binary endogenous variable of interest.
`Z`	Binary instrumental variable.
`X`	A (sparse) matrix of control variables.
`learners`	May take one of two forms, depending on whether a single learner or stacking with multiple learners is used for estimation of the conditional expectation functions. If a single learner is used, `learners` is a list with two named elements:

- what The base learner function. The function must be such that it predicts a named input y using a named input X.
- args Optional arguments to be passed to what.

If stacking with multiple learners is used, learners is a list of lists, each containing four named elements:

- fun The base learner function. The function must be such that it predicts a named input y using a named input X.
- args Optional arguments to be passed to fun.
- assign_X An optional vector of column indices corresponding to control variables in X that are passed to the base learner.
- assign_Z An optional vector of column indices corresponding to instruments in Z that are passed to the base learner.

Omission of the args element results in default arguments being used in fun. Omission of assign_X (and/or assign_Z) results in inclusion of all variables in X (and/or Z).

learners_DXZ, learners_ZX
: Optional arguments to allow for different estimators of $E[D|X,Z]$, $E[Z|X]$. Setup is identical to learners.

sample_folds
: Number of cross-fitting folds.

ensemble_type
: Ensemble method to combine base learners into final estimate of the conditional expectation functions. Possible values are:

- "nnls" Non-negative least squares.
- "nnls1" Non-negative least squares with the constraint that all weights sum to one.
- "singlebest" Select base learner with minimum MSPE.
- "ols" Ordinary least squares.
- "average" Simple average over base learners.

Multiple ensemble types may be passed as a vector of strings.

shortstack
: Boolean to use short-stacking.

cv_folds
: Number of folds used for cross-validation in ensemble construction.

custom_ensemble_weights
: A numerical matrix with user-specified ensemble weights. Each column corresponds to a custom ensemble specification, each row corresponds to a base learner in learners (in chronological order). Optional column names are used to name the estimation results corresponding the custom ensemble specification.

custom_ensemble_weights_DXZ, custom_ensemble_weights_ZX
: Optional arguments to allow for different custom ensemble weights for learners_DXZ,learners_ZX. Setup is identical to custom_ensemble_weights. Note: custom_ensemble_weights and custom_ensemble_weights_DXZ,custom_ensemble_weights_ZX must have the same number of columns.

cluster_variable
: A vector of cluster indices.

subsamples_byZ
: List of two lists corresponding to the two instrument levels. Each list contains vectors with sample indices for cross-fitting.

cv_subsamples_byZ
: List of two lists, each corresponding to one of the two instrument levels. Each of the two lists contains lists, each corresponding to a subsample and contains vectors with subsample indices for cross-validation.

trim
: Number in (0, 1) for trimming the estimated propensity scores at trim and 1-trim.

silent
: Boolean to silence estimation updates.

Details

ddml_late provides a double/debiased machine learning estimator for the local average treatment effect in the interactive model given by

$Y = g_0(D, X) + U,$

where (Y, D, X, Z, U) is a random vector such that $\operatorname{supp} D = \operatorname{supp} Z = \{0, 1\}$, $E[U|X, Z] = 0$, $E[Var(E[D|X, Z]|X)] \neq 0$, $\Pr(Z = 1|X) \in (0, 1)$ with probability 1, $p_0(1, X) \geq p_0(0, X)$ with probability 1 where $p_0(Z, X) \equiv \Pr(D = 1|Z, X)$, and g_0 is an unknown nuisance function.

In this model, the local average treatment effect is defined as

$\theta_0^{\textrm{LATE}} \equiv E[g_0(1, X) - g_0(0, X)|p_0(1, X) > p(0, X)].$

Value

ddml_late returns an object of S3 class ddml_late. An object of class ddml_late is a list containing the following components:

late A vector with the average treatment effect estimates.

weights A list of matrices, providing the weight assigned to each base learner (in chronological order) by the ensemble procedure.

mspe A list of matrices, providing the MSPE of each base learner (in chronological order) computed by the cross-validation step in the ensemble construction.

psi_a, psi_b Matrices needed for the computation of scores. Used in summary.ddml_late().

oos_pred List of matrices, providing the reduced form predicted values.

learners,learners_DXZ,learners_ZX, cluster_variable,subsamples_Z0, subsamples_Z1,cv_subsamples_list_Z0,
Pass-through of selected user-provided arguments. See above.

References

Ahrens A, Hansen C B, Schaffer M E, Wiemann T (2023). "ddml: Double/debiased machine learning in Stata." https://arxiv.org/abs/2301.09397

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

Imbens G, Angrist J (1004). "Identification and Estimation of Local Average Treatment Effects." Econometrica, 62(2), 467-475.

Wolpert D H (1992). "Stacked generalization." Neural Networks, 5(2), 241-259.

See Also

summary.ddml_late()

Other ddml: ddml_ate(), ddml_fpliv(), ddml_pliv(), ddml_plm()

Examples

```
# Construct variables from the included Angrist & Evans (1998) data
y = AE98[, "worked"]
D = AE98[, "morekids"]
Z = AE98[, "samesex"]
X = AE98[, c("age","agefst","black","hisp","othrace","educ")]

# Estimate the local average treatment effect using a single base learner,
#     ridge.
late_fit <- ddml_late(y, D, Z, X,
                      learners = list(what = mdl_glmnet,
                                      args = list(alpha = 0)),
                      sample_folds = 2,
                      silent = TRUE)
summary(late_fit)

# Estimate the local average treatment effect using short-stacking with base
#     learners ols, lasso, and ridge. We can also use custom_ensemble_weights
#     to estimate the ATE using every individual base learner.
weights_everylearner <- diag(1, 3)
colnames(weights_everylearner) <- c("mdl:ols", "mdl:lasso", "mdl:ridge")
late_fit <- ddml_late(y, D, Z, X,
                      learners = list(list(fun = ols),
                                      list(fun = mdl_glmnet),
                                      list(fun = mdl_glmnet,
                                           args = list(alpha = 0))),
                      ensemble_type = 'nnls',
                      custom_ensemble_weights = weights_everylearner,
                      shortstack = TRUE,
                      sample_folds = 2,
                      silent = TRUE)
summary(late_fit)
```

---

ddml_pliv *Estimator for the Partially Linear IV Model.*

Description

Estimator for the partially linear IV model.

Usage

```
ddml_pliv(
  y,
```

```
  D,
  Z,
  X,
  learners,
  learners_DX = learners,
  learners_ZX = learners,
  sample_folds = 10,
  ensemble_type = "nnls",
  shortstack = FALSE,
  cv_folds = 10,
  custom_ensemble_weights = NULL,
  custom_ensemble_weights_DX = custom_ensemble_weights,
  custom_ensemble_weights_ZX = custom_ensemble_weights,
  cluster_variable = seq_along(y),
  subsamples = NULL,
  cv_subsamples_list = NULL,
  silent = FALSE
)
```

Arguments

`y` The outcome variable.

`D` A matrix of endogenous variables.

`Z` A matrix of instruments.

`X` A (sparse) matrix of control variables.

`learners` May take one of two forms, depending on whether a single learner or stacking with multiple learners is used for estimation of the conditional expectation functions. If a single learner is used, `learners` is a list with two named elements:

- `what` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
- `args` Optional arguments to be passed to `what`.

If stacking with multiple learners is used, `learners` is a list of lists, each containing four named elements:

- `fun` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
- `args` Optional arguments to be passed to `fun`.
- `assign_X` An optional vector of column indices corresponding to control variables in `X` that are passed to the base learner.
- `assign_Z` An optional vector of column indices corresponding to instruments in `Z` that are passed to the base learner.

Omission of the `args` element results in default arguments being used in `fun`. Omission of `assign_X` (and/or `assign_Z`) results in inclusion of all variables in `X` (and/or `Z`).

`learners_DX`, `learners_ZX`

Optional arguments to allow for different base learners for estimation of $E[D|X]$, $E[Z|X]$. Setup is identical to `learners`.

sample_folds Number of cross-fitting folds.

ensemble_type Ensemble method to combine base learners into final estimate of the conditional expectation functions. Possible values are:

- "nnls" Non-negative least squares.
- "nnls1" Non-negative least squares with the constraint that all weights sum to one.
- "singlebest" Select base learner with minimum MSPE.
- "ols" Ordinary least squares.
- "average" Simple average over base learners.

Multiple ensemble types may be passed as a vector of strings.

shortstack Boolean to use short-stacking.

cv_folds Number of folds used for cross-validation in ensemble construction.

custom_ensemble_weights
A numerical matrix with user-specified ensemble weights. Each column corresponds to a custom ensemble specification, each row corresponds to a base learner in learners (in chronological order). Optional column names are used to name the estimation results corresponding the custom ensemble specification.

custom_ensemble_weights_DX, custom_ensemble_weights_ZX
Optional arguments to allow for different custom ensemble weights for learners_DX,learners_ZX. Setup is identical to custom_ensemble_weights. Note: custom_ensemble_weights and custom_ensemble_weights_DX,custom_ensemble_weights_ZX must have the same number of columns.

cluster_variable
A vector of cluster indices.

subsamples List of vectors with sample indices for cross-fitting.

cv_subsamples_list
List of lists, each corresponding to a subsample containing vectors with subsample indices for cross-validation.

silent Boolean to silence estimation updates.

Details

ddml_pliv provides a double/debiased machine learning estimator for the parameter of interest θ_0 in the partially linear IV model given by

$Y = \theta_0 D + g_0(X) + U,$

where (Y, D, X, Z, U) is a random vector such that $E[Cov(U, Z|X)] = 0$ and $E[Cov(D, Z|X)] \neq 0$, and g_0 is an unknown nuisance function.

Value

ddml_pliv returns an object of S3 class ddml_pliv. An object of class ddml_pliv is a list containing the following components:

coef A vector with the θ_0 estimates.

`weights` A list of matrices, providing the weight assigned to each base learner (in chronological order) by the ensemble procedure.

`mspe` A list of matrices, providing the MSPE of each base learner (in chronological order) computed by the cross-validation step in the ensemble construction.

`iv_fit` Object of class `ivreg` from the IV regression of $Y - \hat{E}[Y|X]$ on $D - \hat{E}[D|X]$ using $Z - \hat{E}[Z|X]$ as the instrument. See also `AER::ivreg()` for details.

`learners`,`learners_DX`,`learners_ZX`, `cluster_variable`, `subsamples`, `cv_subsamples_list`,`ensemble_type` Pass-through of selected user-provided arguments. See above.

References

Ahrens A, Hansen C B, Schaffer M E, Wiemann T (2023). "ddml: Double/debiased machine learning in Stata." https://arxiv.org/abs/2301.09397

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

Kleiber C, Zeileis A (2008). Applied Econometrics with R. Springer-Verlag, New York.

Wolpert D H (1992). "Stacked generalization." Neural Networks, 5(2), 241-259.

See Also

`summary.ddml_pliv()`, `AER::ivreg()`

Other ddml: `ddml_ate()`, `ddml_fpliv()`, `ddml_late()`, `ddml_plm()`

Examples

```
# Construct variables from the included Angrist & Evans (1998) data
y = AE98[, "worked"]
D = AE98[, "morekids"]
Z = AE98[, "samesex"]
X = AE98[, c("age","agefst","black","hisp","othrace","educ")]

# Estimate the partially linear IV model using a single base learner, ridge.
pliv_fit <- ddml_pliv(y, D, Z, X,
                      learners = list(what = mdl_glmnet,
                                      args = list(alpha = 0)),
                      sample_folds = 2,
                      silent = TRUE)
summary(pliv_fit)
```

`ddml_plm` *Estimator for the Partially Linear Model.*

Description

Estimator for the partially linear model.

Usage

```
ddml_plm(
  y,
  D,
  X,
  learners,
  learners_DX = learners,
  sample_folds = 10,
  ensemble_type = "nnls",
  shortstack = FALSE,
  cv_folds = 10,
  custom_ensemble_weights = NULL,
  custom_ensemble_weights_DX = custom_ensemble_weights,
  cluster_variable = seq_along(y),
  subsamples = NULL,
  cv_subsamples_list = NULL,
  silent = FALSE
)
```

Arguments

`y` The outcome variable.

`D` A matrix of endogenous variables.

`X` A (sparse) matrix of control variables.

`learners` May take one of two forms, depending on whether a single learner or stacking with multiple learners is used for estimation of the conditional expectation functions. If a single learner is used, `learners` is a list with two named elements:

- `what` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
- `args` Optional arguments to be passed to `what`.

If stacking with multiple learners is used, `learners` is a list of lists, each containing four named elements:

- `fun` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
- `args` Optional arguments to be passed to `fun`.
- `assign_X` An optional vector of column indices corresponding to control variables in `X` that are passed to the base learner.

Omission of the `args` element results in default arguments being used in `fun`. Omission of `assign_X` results in inclusion of all variables in `X`.

`learners_DX` Optional argument to allow for different estimators of $E[D|X]$. Setup is identical to `learners`.

`sample_folds` Number of cross-fitting folds.

`ensemble_type` Ensemble method to combine base learners into final estimate of the conditional expectation functions. Possible values are:

- "nnls" Non-negative least squares.
- "nnls1" Non-negative least squares with the constraint that all weights sum to one.
- "singlebest" Select base learner with minimum MSPE.
- "ols" Ordinary least squares.
- "average" Simple average over base learners.

Multiple ensemble types may be passed as a vector of strings.

shortstack Boolean to use short-stacking.

cv_folds Number of folds used for cross-validation in ensemble construction.

custom_ensemble_weights
A numerical matrix with user-specified ensemble weights. Each column corresponds to a custom ensemble specification, each row corresponds to a base learner in learners (in chronological order). Optional column names are used to name the estimation results corresponding the custom ensemble specification.

custom_ensemble_weights_DX
Optional argument to allow for different custom ensemble weights for learners_DX. Setup is identical to custom_ensemble_weights. Note: custom_ensemble_weights and custom_ensemble_weights_DX must have the same number of columns.

cluster_variable
A vector of cluster indices.

subsamples List of vectors with sample indices for cross-fitting.

cv_subsamples_list
List of lists, each corresponding to a subsample containing vectors with subsample indices for cross-validation.

silent Boolean to silence estimation updates.

Details

ddml_plm provides a double/debiased machine learning estimator for the parameter of interest θ_0 in the partially linear model given by

$Y = \theta_0 D + g_0(X) + U,$

where (Y, D, X, U) is a random vector such that $E[Cov(U, D|X)] = 0$ and $E[Var(D|X)] \neq 0$, and g_0 is an unknown nuisance function.

Value

ddml_plm returns an object of S3 class ddml_plm. An object of class ddml_plm is a list containing the following components:

coef A vector with the θ_0 estimates.

weights A list of matrices, providing the weight assigned to each base learner (in chronological order) by the ensemble procedure.

mspe A list of matrices, providing the MSPE of each base learner (in chronological order) computed by the cross-validation step in the ensemble construction.

ols_fit Object of class lm from the second stage regression of $Y - \hat{E}[Y|X]$ on $D - \hat{E}[D|X]$.

learners,learners_DX,cluster_variable, subsamples, cv_subsamples_list, ensemble_type
Pass-through of selected user-provided arguments. See above.

References

Ahrens A, Hansen C B, Schaffer M E, Wiemann T (2023). "ddml: Double/debiased machine learning in Stata." https://arxiv.org/abs/2301.09397

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

Wolpert D H (1992). "Stacked generalization." Neural Networks, 5(2), 241-259.

See Also

summary.ddml_plm()

Other ddml: ddml_ate(), ddml_fpliv(), ddml_late(), ddml_pliv()

Examples

```
# Construct variables from the included Angrist & Evans (1998) data
y = AE98[, "worked"]
D = AE98[, "morekids"]
X = AE98[, c("age","agefst","black","hisp","othrace","educ")]

# Estimate the partially linear model using a single base learner, ridge.
plm_fit <- ddml_plm(y, D, X,
                    learners = list(what = mdl_glmnet,
                                    args = list(alpha = 0)),
                    sample_folds = 2,
                    silent = TRUE)
summary(plm_fit)

# Estimate the partially linear model using short-stacking with base learners
#     ols, lasso, and ridge. We can also use custom_ensemble_weights
#     to estimate the ATE using every individual base learner.
weights_everylearner <- diag(1, 3)
colnames(weights_everylearner) <- c("mdl:ols", "mdl:lasso", "mdl:ridge")
plm_fit <- ddml_plm(y, D, X,
                    learners = list(list(fun = ols),
                                    list(fun = mdl_glmnet),
                                    list(fun = mdl_glmnet,
                                         args = list(alpha = 0))),
                    ensemble_type = 'nnls',
                    custom_ensemble_weights = weights_everylearner,
                    shortstack = TRUE,
                    sample_folds = 2,
                    silent = TRUE)
summary(plm_fit)
```

mdl_glm — *Wrapper for stats::glm().*

Description

Simple wrapper for stats::glm().

Usage

```
mdl_glm(y, X, ...)
```

Arguments

y	The outcome variable.
X	The feature matrix.
...	Additional arguments passed to glm. See stats::glm() for a complete list of arguments.

Value

mdl_glm returns an object of S3 class mdl_glm as a simple mask of the return object of stats::glm().

See Also

stats::glm()

Other ml_wrapper: mdl_glmnet(), mdl_ranger(), mdl_xgboost(), ols()

Examples

```
glm_fit <- mdl_glm(sample(0:1, 100, replace = TRUE),
                   matrix(rnorm(1000), 100, 10))
class(glm_fit)
```

mdl_glmnet — *Wrapper for glmnet::glmnet().*

Description

Simple wrapper for glmnet::glmnet() and glmnet::cv.glmnet().

Usage

```
mdl_glmnet(y, X, cv = TRUE, ...)
```

Arguments

y	The outcome variable.
X	The (sparse) feature matrix.
cv	Boolean to indicate use of lasso with cross-validated penalty.
...	Additional arguments passed to glmnet. See glmnet::glmnet() and glmnet::cv.glmnet() for a complete list of arguments.

Value

mdl_glmnet returns an object of S3 class mdl_glmnet as a simple mask of the return object of glmnet::glmnet() or glmnet::cv.glmnet().

References

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." Journal of Statistical Software, 33(1), 1–22.

Simon N, Friedman J, Hastie T, Tibshirani R (2011). "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent." Journal of Statistical Software, 39(5), 1–13.

See Also

glmnet::glmnet(),glmnet::cv.glmnet()

Other ml_wrapper: mdl_glm(), mdl_ranger(), mdl_xgboost(), ols()

Examples

```
glmnet_fit <- mdl_glmnet(rnorm(100), matrix(rnorm(1000), 100, 10))
class(glmnet_fit)
```

---

mdl_ranger *Wrapper for ranger::ranger().*

Description

Simple wrapper for ranger::ranger(). Supports regression (default) and probability forests (set probability = TRUE).

Usage

```
mdl_ranger(y, X, ...)
```

Arguments

y	The outcome variable.
X	The feature matrix.
...	Additional arguments passed to ranger. See ranger::ranger() for a complete list of arguments.

Value

`mdl_ranger` returns an object of S3 class `ranger` as a simple mask of the return object of `ranger::ranger()`.

References

Wright M N, Ziegler A (2017). "ranger: A fast implementation of random forests for high dimensional data in C++ and R." Journal of Statistical Software 77(1), 1-17.

See Also

`ranger::ranger()`

Other ml_wrapper: `mdl_glmnet()`, `mdl_glm()`, `mdl_xgboost()`, `ols()`

Examples

```
ranger_fit <- mdl_ranger(rnorm(100), matrix(rnorm(1000), 100, 10))
class(ranger_fit)
```

---

mdl_xgboost *Wrapper for `xgboost::xgboost()`.*

Description

Simple wrapper for `xgboost::xgboost()` with some changes to the default arguments.

Usage

```
mdl_xgboost(y, X, nrounds = 500, verbose = 0, ...)
```

Arguments

`y`	The outcome variable.
`X`	The (sparse) feature matrix.
`nrounds`	max number of boosting iterations.
`verbose`	If 0, xgboost will stay silent. If 1, it will print information about performance. If 2, some additional information will be printed out. Note that setting `verbose > 0` automatically engages the `cb.print.evaluation(period=1)` callback function.
`...`	Additional arguments passed to `xgboost`. See `xgboost::xgboost()` for a complete list of arguments.

Value

`mdl_xgboost` returns an object of S3 class `mdl_xgboost` as a simple mask to the return object of `xgboost::xgboost()`.

References

Chen T, Guestrin C (2011). "Xgboost: A Scalable Tree Boosting System." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.

See Also

`xgboost::xgboost()`

Other ml_wrapper: `mdl_glmnet()`, `mdl_glm()`, `mdl_ranger()`, `ols()`

Examples

```
xgboost_fit <- mdl_xgboost(rnorm(50), matrix(rnorm(150), 50, 3),
                           nrounds = 1)
class(xgboost_fit)
```

`ols` *Ordinary least squares.*

Description

Simple implementation of ordinary least squares that computes with sparse feature matrices.

Usage

```
ols(y, X, const = TRUE, w = NULL)
```

Arguments

`y`	The outcome variable.
`X`	The feature matrix.
`const`	Boolean equal to `TRUE` if a constant should be included.
`w`	A vector of weights for weighted least squares.

Value

`ols` returns an object of S3 class `ols`. An object of class `ols` is a list containing the following components:

`coef` A vector with the regression coefficents.

`y`, `X`, `const`, `w` Pass-through of the user-provided arguments. See above.

See Also

Other ml_wrapper: `mdl_glmnet()`, `mdl_glm()`, `mdl_ranger()`, `mdl_xgboost()`

Examples

```
ols_fit <- ols(rnorm(100), cbind(rnorm(100), rnorm(100)), const = TRUE)
ols_fit$coef
```

`print.summary.ddml_ate` *Print Methods for Treatment Effect Estimators.*

Description

Print methods for treatment effect estimators.

Usage

```
## S3 method for class 'summary.ddml_ate'
print(x, digits = 3, ...)

## S3 method for class 'summary.ddml_att'
print(x, digits = 3, ...)

## S3 method for class 'summary.ddml_late'
print(x, digits = 3, ...)
```

Arguments

x	An object of class summary.ddml_ate, summary.ddml_att, and ddml_late, as returned by summary.ddml_ate(), summary.ddml_att(), and summary.ddml_late(), respectively.
digits	The number of significant digits used for printing.
...	Currently unused.

Value

NULL.

Examples

```
# Construct variables from the included Angrist & Evans (1998) data
y = AE98[, "worked"]
D = AE98[, "morekids"]
X = AE98[, c("age","agefst","black","hisp","othrace","educ")]

# Estimate the average treatment effect using a single base learner, ridge.
ate_fit <- ddml_ate(y, D, X,
                    learners = list(what = mdl_glmnet,
                                    args = list(alpha = 0)),
                    sample_folds = 2,
                    silent = TRUE)
summary(ate_fit)
```

print.summary.ddml_fpliv

Print Methods for Treatment Effect Estimators.

Description

Print methods for treatment effect estimators.

Usage

```
## S3 method for class 'summary.ddml_fpliv'
print(x, digits = 3, ...)

## S3 method for class 'summary.ddml_pliv'
print(x, digits = 3, ...)

## S3 method for class 'summary.ddml_plm'
print(x, digits = 3, ...)
```

Arguments

x	An object of class summary.ddml_plm, summary.ddml_pliv, and summary.ddml_fpliv, as returned by summary.ddml_plm(), summary.ddml_pliv(), and summary.ddml_fpliv(), respectively.
digits	Number of significant digits used for priniting.
...	Currently unused.

Value

NULL.

Examples

```
# Construct variables from the included Angrist & Evans (1998) data
y = AE98[, "worked"]
D = AE98[, "morekids"]
X = AE98[, c("age","agefst","black","hisp","othrace","educ")]

# Estimate the partially linear model using a single base learner, ridge.
plm_fit <- ddml_plm(y, D, X,
                    learners = list(what = mdl_glmnet,
                                    args = list(alpha = 0)),
                    sample_folds = 2,
                    silent = TRUE)
summary(plm_fit)
```

shortstacking *Predictions using Short-Stacking.*

Description

Predictions using short-stacking.

Usage

```
shortstacking(
  y,
  X,
  Z = NULL,
  learners,
  sample_folds = 2,
  ensemble_type = "average",
  custom_ensemble_weights = NULL,
  compute_insample_predictions = FALSE,
  subsamples = NULL,
  silent = FALSE,
  progress = NULL,
  auxiliary_X = NULL,
  shortstack_y = y
)
```

Arguments

`y` The outcome variable.

`X` A (sparse) matrix of predictive variables.

`Z` Optional additional (sparse) matrix of predictive variables.

`learners` May take one of two forms, depending on whether a single learner or stacking with multiple learners is used for estimation of the predictor. If a single learner is used, `learners` is a list with two named elements:

- `what` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
- `args` Optional arguments to be passed to `what`.

If stacking with multiple learners is used, `learners` is a list of lists, each containing four named elements:

- `fun` The base learner function. The function must be such that it predicts a named input `y` using a named input `X`.
- `args` Optional arguments to be passed to `fun`.
- `assign_X` An optional vector of column indices corresponding to predictive variables in `X` that are passed to the base learner.
- `assign_Z` An optional vector of column indices corresponding to predictive in `Z` that are passed to the base learner.

Omission of the args element results in default arguments being used in fun. Omission of assign_X (and/or assign_Z) results in inclusion of all variables in X (and/or Z).

sample_folds
: Number of cross-fitting folds.

ensemble_type
: Ensemble method to combine base learners into final estimate of the conditional expectation functions. Possible values are:
  - "nnls" Non-negative least squares.
  - "nnls1" Non-negative least squares with the constraint that all weights sum to one.
  - "singlebest" Select base learner with minimum MSPE.
  - "ols" Ordinary least squares.
  - "average" Simple average over base learners.

  Multiple ensemble types may be passed as a vector of strings.

custom_ensemble_weights
: A numerical matrix with user-specified ensemble weights. Each column corresponds to a custom ensemble specification, each row corresponds to a base learner in learners (in chronological order). Optional column names are used to name the estimation results corresponding the custom ensemble specification.

compute_insample_predictions
: Indicator equal to 1 if in-sample predictions should also be computed.

subsamples
: List of vectors with sample indices for cross-fitting.

silent
: Boolean to silence estimation updates.

progress
: String to print before learner and cv fold progress.

auxiliary_X
: An optional list of matrices of length sample_folds, each containing additional observations to calculate predictions for.

shortstack_y
: Optional vector of the outcome variable to form short-stacking predictions for. Base learners are always trained on y.

Value

shortstack returns a list containing the following components:

oos_fitted
: A matrix of out-of-sample predictions, each column corresponding to an ensemble type (in chronological order).

weights
: An array, providing the weight assigned to each base learner (in chronological order) by the ensemble procedures.

is_fitted
: When compute_insample_predictions = T. a list of matrices with in-sample predictions by sample fold.

auxiliary_fitted
: When auxiliary_X is not NULL, a list of matrices with additional predictions.

oos_fitted_bylearner
: A matrix of out-of-sample predictions, each column corresponding to a base learner (in chronological order).

is_fitted_bylearner
: When compute_insample_predictions = T, a list of matrices with in-sample predictions by sample fold.

`auxiliary_fitted_bylearner` When `auxiliary_X` is not `NULL`, a list of matrices with additional predictions for each learner.

Note that unlike `crosspred`, `shortstack` always computes out-of-sample predictions for each base learner (at no additional computational cost).

References

Ahrens A, Hansen C B, Schaffer M E, Wiemann T (2023). "ddml: Double/debiased machine learning in Stata." https://arxiv.org/abs/2301.09397

Wolpert D H (1992). "Stacked generalization." Neural Networks, 5(2), 241-259.

See Also

Other utilities: `crosspred()`, `crossval()`

Examples

```
# Construct variables from the included Angrist & Evans (1998) data
y = AE98[, "worked"]
X = AE98[, c("morekids", "age","agefst","black","hisp","othrace","educ")]

# Compute predictions using shortstacking with base learners ols and lasso.
#     Two stacking approaches are simultaneously computed: Equally
#     weighted (ensemble_type = "average") and MSPE-minimizing with weights
#     in the unit simplex (ensemble_type = "nnls1"). Predictions for each
#     learner are also calculated.
shortstack_res <- shortstacking(y, X,
                                learners = list(list(fun = ols),
                                                list(fun = mdl_glmnet)),
                                ensemble_type = c("average",
                                                  "nnls1",
                                                  "singlebest"),
                                sample_folds = 2,
                                silent = TRUE)
dim(shortstack_res$oos_fitted) # = length(y) by length(ensemble_type)
dim(shortstack_res$oos_fitted_bylearner) # = length(y) by length(learners)
```

`summary.ddml_ate` *Inference Methods for Treatment Effect Estimators.*

Description

Inference methods for treatment effect estimators. By default, standard errors are heteroskedasitictyrobust. If the `ddml` estimator was computed using a `cluster_variable`, the standard errors are also cluster-robust by default.

Usage

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

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

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

Arguments

`object`	An object of class `ddml_ate`, `ddml_att`, and `ddml_late`, as fitted by `ddml_ate()`, `ddml_att()`, and `ddml_late()`, respectively.
`...`	Currently unused.

Value

A matrix with inference results.

Examples

```
# Construct variables from the included Angrist & Evans (1998) data
y = AE98[, "worked"]
D = AE98[, "morekids"]
X = AE98[, c("age","agefst","black","hisp","othrace","educ")]

# Estimate the average treatment effect using a single base learner, ridge.
ate_fit <- ddml_ate(y, D, X,
                    learners = list(what = mdl_glmnet,
                                    args = list(alpha = 0)),
                    sample_folds = 2,
                    silent = TRUE)
summary(ate_fit)
```

`summary.ddml_fpliv` *Inference Methods for Partially Linear Estimators.*

Description

Inference methods for partially linear estimators. Simple wrapper for `sandwich::vcovHC()` and `sandwich::vcovCL()`. Default standard errors are heteroskedasiticty-robust. If the `ddml` estimator was computed using a `cluster_variable`, the standard errors are also cluster-robust by default.

Usage

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

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

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

Arguments

`object`	An object of class `ddml_plm`, `ddml_pliv`, or `ddml_fpliv` as fitted by `ddml_plm()`, `ddml_pliv()`, and `ddml_fpliv()`, respectively.
`...`	Additional arguments passed to `vcovHC` and `vcovCL`. See `sandwich::vcovHC()` and `sandwich::vcovCL()` for a complete list of arguments.

Value

An array with inference results for each `ensemble_type`.

References

Zeileis A (2004). "Econometric Computing with HC and HAC Covariance Matrix Estimators." Journal of Statistical Software, 11(10), 1-17.

Zeileis A (2006). "Object-Oriented Computation of Sandwich Estimators." Journal of Statistical Software, 16(9), 1-16.

Zeileis A, Köll S, Graham N (2020). "Various Versatile Variances: An Object-Oriented Implementation of Clustered Covariances in R." Journal of Statistical Software, 95(1), 1-36.

See Also

`sandwich::vcovHC()`, `sandwich::vcovCL()`

Examples

```
# Construct variables from the included Angrist & Evans (1998) data
y = AE98[, "worked"]
D = AE98[, "morekids"]
X = AE98[, c("age","agefst","black","hisp","othrace","educ")]

# Estimate the partially linear model using a single base learner, ridge.
plm_fit <- ddml_plm(y, D, X,
                    learners = list(what = mdl_glmnet,
                                    args = list(alpha = 0)),
                    sample_folds = 2,
                    silent = TRUE)
summary(plm_fit)
```

Index