Package: dmlrs (via r-universe)

November 1, 2026

Type Package

Title Fast Double / Debiased Machine Learning Implemented in Rust

Version 0.1.0

Description Double / debiased machine learning (Chernozhukov et al., 2018, <doi:10.1111/ectj.12097>) for causal inference: partially linear, interactive, instrumental variable and difference-in-differences models with cross-fitted nuisance functions, multiplier bootstrap and sensitivity analysis for omitted confounding. Estimation, including the nuisance learners (lasso, logistic regression, random forests, gradient boosting), runs in Rust through the 'dml-core' crate, in parallel. Results match the Python package 'dml-rs', which is validated against 'DoubleML'.

License MIT + file LICENSE

URL https://asadgudzevic94.github.io/dml-rs/, https://github.com/AsadGudzevic94/dml-rs

BugReports https://github.com/AsadGudzevic94/dml-rs/issues

Encoding UTF-8

Depends R (>= 4.2)

Imports stats

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

SystemRequirements Cargo (Rust's package manager), rustc >= 1.80

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libclang-dev

Repository https://asadgudzevic94.r-universe.dev

Date/Publication 2026-11-01 05:02:56 UTC

RemoteUrl https://github.com/AsadGudzevic94/dml-rs

RemoteRef HEAD

RemoteSha f96392695708d7bfe63d4666825598409ff8573f

RemoteSubdir r/dmlrs

Contents

confint.dmlrs_fit 2
dml_did 3
dml_iivm 4
dml_irm 5
dml_pliv 6
dml_plr 7
learners 8
sensitivity_analysis 11

Index **12**

`confint.dmlrs_fit` *Confidence intervals for DML estimates*

Description

Pointwise intervals, or simultaneous intervals for all treatments from the multiplier bootstrap (`joint = TRUE`), as DoubleML's `confint()`.

Usage

```
## S3 method for class 'dmlrs_fit'
confint(
  object,
  parm,
  level = 0.95,
  joint = FALSE,
  method = c("normal", "Bayes", "wild"),
  n_boot = 500,
  seed = 0,
  ...
)
```

Arguments

`object`	A fitted model.
`parm`	Treatments (names or indices); all by default.
`level`	Confidence level.
`joint`	Simultaneous intervals from the multiplier bootstrap.

method	Bootstrap weights: "normal", "Bayes" or "wild".
n_boot	Number of bootstrap draws.
seed	Seed of the bootstrap.
...	Not used.

Value

A matrix with one row per treatment.

Examples

```
set.seed(3)
n <- 400
x <- matrix(rnorm(n * 3), n)
d <- cbind(d1 = x[, 1] + rnorm(n), d2 = x[, 2] + rnorm(n))
y <- 0.5 * d[, 1] - 0.2 * d[, 2] + x[, 3] + rnorm(n)
fit <- dml_plr(y, d, x, ml_l = lrn_ols())
confint(fit, joint = TRUE, n_boot = 1000)
```

---

dml_did *Difference in differences*

Description

The average treatment effect on the treated with two periods (Sant'Anna and Zhao, 2020), as DoubleMLDID and DoubleMLDIDCS. With panel data, y is the change of the outcome between the periods; with repeated cross-sections, y is the outcome and t the period (0/1).

Usage

```
dml_did(
  y,
  d,
  x,
  t = NULL,
  ml_g = lrn_lasso_cv(),
  ml_m = lrn_logistic_cv(),
  score = c("observational", "experimental"),
  in_sample_normalization = TRUE,
  n_folds = 5,
  n_rep = 1,
  seed = 0
)
```

Arguments

y	Outcome change (panel) or outcome (repeated cross-sections).
d	Binary treatment group indicator (0/1).
x	Covariates: a numeric matrix or data frame.
t	NULL for panel data, or the period indicator (0/1) of repeated cross-sections.
ml_g	Learner for the outcome regressions.
ml_m	Classifier for the propensity score (not used by the "experimental" score).
score	"observational" or "experimental".
in_sample_normalization	Normalise the weights in sample, as DoubleML.
n_folds	Number of cross-fitting folds.
n_rep	Number of repetitions of the sample split; estimates are aggregated with the median, as in DoubleML.
seed	Seed of the sample splits.

Value

A dmlrs_fit object; see stats::coef(), confint.dmlrs_fit(), base::summary() and sensitivity_analysis().

---

dml_iivm *Interactive IV model*

Description

Local average treatment effect of a binary treatment with a binary instrument, as DoubleMLIIVM.

Usage

```
dml_iivm(
  y,
  d,
  z,
  x,
  ml_g = lrn_lasso_cv(),
  ml_m = lrn_logistic_cv(),
  ml_r = ml_m,
  subgroups = c(always_takers = TRUE, never_takers = TRUE),
  n_folds = 5,
  n_rep = 1,
  seed = 0
)
```

Arguments

y	Outcome (numeric vector).
d	Binary treatment (0/1).
z	Binary instrument (0/1).
x	Covariates: a numeric matrix or data frame.
ml_g	Learner for $E[Y \mid Z, X]$.
ml_m	Classifier for $P(Z = 1 \mid X)$.
ml_r	Classifier for $P(D = 1 \mid Z, X)$.
subgroups	Named logical vector: whether there can be `always_takers` and `never_takers` (as DoubleML's `subgroups`).
n_folds	Number of cross-fitting folds.
n_rep	Number of repetitions of the sample split; estimates are aggregated with the median, as in DoubleML.
seed	Seed of the sample splits.

Value

A `dmlrs_fit` object; see `stats::coef()`, `confint.dmlrs_fit()`, `base::summary()` and `sensitivity_analysis()`.

`dml_irm` *Interactive regression model*

Description

Average treatment effect (`"ATE"`) or average treatment effect on the treated (`"ATTE"`) of a binary treatment, as `DoubleMLIRM`.

Usage

```
dml_irm(
  y,
  d,
  x,
  ml_g = lrn_lasso_cv(),
  ml_m = lrn_logistic_cv(),
  score = c("ATE", "ATTE"),
  clipping_threshold = 0.01,
  normalize_ipw = FALSE,
  n_folds = 5,
  n_rep = 1,
  seed = 0
)
```

Arguments

y	Outcome (numeric vector).
d	Binary treatment (0/1).
x	Covariates: a numeric matrix or data frame.
ml_g	Learner for $E[Y \mid D, X]$ (fitted on each treatment arm).
ml_m	Classifier for the propensity score $P(D = 1 \mid X)$.
score	"ATE" or "ATTE".
clipping_threshold	Propensity scores are clipped to $[c, 1 - c]$.
normalize_ipw	Normalise the inverse-probability weights.
n_folds	Number of cross-fitting folds.
n_rep	Number of repetitions of the sample split; estimates are aggregated with the median, as in DoubleML.
seed	Seed of the sample splits.

Value

A dmlrs_fit object; see stats::coef(), confint.dmlrs_fit(), base::summary() and sensitivity_analysis().

Examples

```
set.seed(2)
n <- 1000
x <- matrix(rnorm(n * 4), n)
d <- rbinom(n, 1, plogis(x[, 1]))
y <- d + x[, 1] + rnorm(n)
dml_irm(y, d, x, ml_g = lrn_ols(), ml_m = lrn_logistic())
```

---

dml_pliv *Partially linear IV regression*

Description

The partially linear model with an endogenous treatment and one instrument Z, as DoubleMLPLIV ("partialling out" score).

Usage

```
dml_pliv(
  y,
  d,
  z,
  x,
  ml_l = lrn_lasso_cv(),
```

```
  ml_m = ml_l,
  ml_r = ml_l,
  n_folds = 5,
  n_rep = 1,
  seed = 0
)
```

Arguments

y	Outcome (numeric vector).
d	Treatment: a numeric vector, or a matrix with one column per treatment (column names become the treatment names).
z	Instrument (numeric vector).
x	Covariates: a numeric matrix or data frame.
ml_l	Learner for $E[Y \mid X]$.
ml_m	Learner for $E[Z \mid X]$.
ml_r	Learner for $E[D \mid X]$.
n_folds	Number of cross-fitting folds.
n_rep	Number of repetitions of the sample split; estimates are aggregated with the median, as in DoubleML.
seed	Seed of the sample splits.

Value

A `dmlrs_fit` object; see `stats::coef()`, `confint.dmlrs_fit()`, `base::summary()` and `sensitivity_analysis()`.

`dml_plr` *Partially linear regression*

Description

Double / debiased machine learning for the partially linear model $Y = D\theta + g(X) + \varepsilon$, as `DoubleMLPLR`. The nuisance functions are cross-fitted in Rust.

Usage

```
dml_plr(
  y,
  d,
  x,
  ml_l = lrn_lasso_cv(),
  ml_m = ml_l,
  ml_g = NULL,
  score = c("partialling out", "IV-type"),
  n_folds = 5,
```

```
  n_rep = 1,
  seed = 0
)
```

Arguments

y	Outcome (numeric vector).
d	Treatment: a numeric vector, or a matrix with one column per treatment (column names become the treatment names).
x	Covariates: a numeric matrix or data frame.
ml_l	Learner for $E[Y \mid X]$ (see learners).
ml_m	Learner for $E[D \mid X]$.
ml_g	Learner for $g(X)$, used by the "IV-type" score (default: ml_l).
score	"partialling out" or "IV-type".
n_folds	Number of cross-fitting folds.
n_rep	Number of repetitions of the sample split; estimates are aggregated with the median, as in DoubleML.
seed	Seed of the sample splits.

Value

A dmlrs_fit object; see stats::coef(), confint.dmlrs_fit(), base::summary() and sensitivity_analysis().

Examples

```
set.seed(1)
n <- 500
x <- matrix(rnorm(n * 5), n)
d <- x[, 1] + rnorm(n)
y <- 0.5 * d + x[, 2] + rnorm(n)
fit <- dml_plr(y, d, x, ml_l = lrn_lasso_cv())
fit
confint(fit)
```

---

learners *Built-in nuisance learners*

Description

Learners for the nuisance functions of the DML models, implemented in Rust. Parameters follow scikit-learn, and results match the learners of the Python package dml-rs. Regressors learn conditional means; classifiers (lrn_logistic(), lrn_logistic_cv(), and the tree ensembles) learn probabilities of a binary variable.

Usage

```
lrn_ols(ridge = 0)

lrn_lasso(alpha = 1, tol = 1e-04, max_iter = 1e+05)

lrn_lasso_cv(n_alphas = 100, eps = 0.001, cv = 5)

lrn_logistic(C = 1)

lrn_logistic_cv(n_cs = 10, cv = 5)

lrn_random_forest(
  n_estimators = 100,
  max_depth = NULL,
  min_samples_leaf = 1,
  max_features = NULL,
  random_state = 0
)

lrn_gradient_boosting(
  n_estimators = 100,
  learning_rate = 0.1,
  max_depth = 3,
  min_samples_leaf = 1,
  subsample = 1,
  random_state = 0
)

lrn_hist_gradient_boosting(
  max_iter = 100,
  learning_rate = 0.1,
  max_leaf_nodes = 31,
  min_samples_leaf = 20,
  l2_regularization = 0,
  random_state = 0
)

lrn_regression_forest(n_estimators = 2000, min_node_size = 5, random_state = 0)
```

Arguments

`ridge`, `alpha`, `C`	Penalties (see above).
`tol`, `max_iter`	Convergence tolerance and maximum iterations (sweeps of coordinate descent, or boosting iterations for `lrn_hist_gradient_boosting()`).
`n_alphas`, `eps`, `n_cs`, `cv`	Grid size, smallest-to-largest penalty ratio and number of folds of the cross-validated learners.
`n_estimators`	Number of trees.

`max_depth` Maximum tree depth (`NULL`: unlimited).

`min_samples_leaf` Minimum observations per leaf.

`max_features` Features tried per split: `NULL` (all for regression, `"sqrt"` for classification), `"sqrt"`, `"log2"`, a fraction or a number.

`random_state` Seed.

`learning_rate` Shrinkage of each tree.

`subsample` Share of observations used for each tree.

`max_leaf_nodes` Maximum leaves per tree (`NULL`: unlimited).

`l2_regularization` L2 penalty on leaf values.

`min_node_size` Minimum observations per leaf (grf's `min.node.size`).

Details

- `lrn_ols()`: least squares with intercept, optionally ridge-penalised (`ridge` is scikit-learn's `Ridge(alpha)`).
- `lrn_lasso()`: lasso with penalty `alpha` (scikit-learn's `Lasso`).
- `lrn_lasso_cv()`: lasso with the penalty chosen by `cv`-fold cross-validation over `n_alphas` values (scikit-learn's `LassoCV`).
- `lrn_logistic()`: L2-penalised logistic regression with inverse penalty `C`.
- `lrn_logistic_cv()`: logistic regression with `C` chosen by stratified cross-validation (log loss) over `n_cs` values from 1e-4 to 1e4.
- `lrn_random_forest()`: random forest (regressor or classifier).
- `lrn_gradient_boosting()`: gradient boosting with the trees of scikit-learn's `GradientBoosting*`.
- `lrn_hist_gradient_boosting()`: histogram gradient boosting, identical to scikit-learn's `HistGradientBoosting*`; the learner for large data.
- `lrn_regression_forest()`: honest regression forest (R's `grf::regression_forest`), regression only.

Value

A learner specification (class `dmlrs_learner`).

Examples

```
lrn_lasso_cv()
lrn_random_forest(n_estimators = 200, min_samples_leaf = 5)
```

`sensitivity_analysis` *Sensitivity analysis for omitted confounding*

Description

Bounds on the effect and robustness values when the confounding is as strong as `cf_y` (share of the residual outcome variance) and `cf_d` (share of the residual Riesz representer variance), with correlation `rho` (Chernozhukov et al., 2022), as DoubleML's `sensitivity_analysis()`. Available for PLR, IRM and DID fits.

Usage

```
sensitivity_analysis(
  object,
  cf_y = 0.03,
  cf_d = 0.03,
  rho = 1,
  level = 0.95,
  null_hypothesis = 0
)
```

Arguments

`object`	A fitted model.
`cf_y`, `cf_d`	Confounding strengths.
`rho`	Correlation of the confounding in the outcome and treatment.
`level`	Confidence level of the bounds.
`null_hypothesis`	Value of the effect under the null hypothesis.

Value

A data frame with one row per treatment: the bounds on the effect (`theta_lower`, `theta_upper`), their confidence bounds (`ci_lower`, `ci_upper`) and the robustness values `rv` and `rva` (in percent).

Examples

```
set.seed(4)
n <- 500
x <- matrix(rnorm(n * 3), n)
d <- x[, 1] + rnorm(n)
y <- 0.5 * d + x[, 2] + rnorm(n)
sensitivity_analysis(dml_plr(y, d, x, ml_l = lrn_ols()))
```

Index

base::summary(), *4–8*

confint.dmlrs_fit, 2
confint.dmlrs_fit(), *4–8*

dml_did, 3
dml_iivm, 4
dml_irm, 5
dml_pliv, 6
dml_plr, 7

learners, 8, *8*
lrn_gradient_boosting *(*learners*)*, 8
lrn_hist_gradient_boosting *(*learners*)*, 8
lrn_lasso *(*learners*)*, 8
lrn_lasso_cv *(*learners*)*, 8
lrn_logistic *(*learners*)*, 8
lrn_logistic_cv *(*learners*)*, 8
lrn_ols *(*learners*)*, 8
lrn_random_forest *(*learners*)*, 8
lrn_regression_forest *(*learners*)*, 8

sensitivity_analysis, 11
sensitivity_analysis(), *4–8*
stats::coef(), *4–8*