Package ‘dmlalg’

October 13, 2022

Title Double Machine Learning Algorithms

Version 1.0.2

Description Implementation of double machine learning (DML) algorithms in R, based on Emmenegger and Buehlmann (2021) ``Regularizing Double Machine Learning in Partially Linear Endogenous Models'' <arXiv:2101.12525> and Emmenegger and Buehlmann (2021) <arXiv:2108.13657> ``Double Machine Learning for Partially Linear Mixed-Effects Models with Repeated Measurements''.
First part: our goal is to perform inference for the linear parameter in partially linear models with confounding variables.
The standard DML estimator of the linear parameter has a two-stage least squares interpretation, which can lead to a large variance and overwide confidence intervals.
We apply regularization to reduce the variance of the estimator, which produces narrower confidence intervals that are approximately valid.
Nuisance terms can be flexibly estimated with machine learning algorithms.
Second part: our goal is to estimate and perform inference for the linear coefficient in a partially linear mixed-effects model with DML. Machine learning algorithms allows us to incorporate more complex interaction structures and high-dimensional variables.

License GPL (>= 3)

URL https://gitlab.math.ethz.ch/ecorinne/dmlalg.git

Encoding UTF-8

Depends R (>= 4.0.0), stats

Suggests testthat (>= 3.0.0)

Imports glmnet, lme4, matrixcalc, methods, splines, randomForest

Config/testthat/edition 3

NeedsCompilation no

Author Corinne Emmenegger [aut, cre] (<https://orcid.org/0000-0003-0353-8888>), Peter Buehlmann [ths] (<https://orcid.org/0000-0002-1782-6015>)

Maintainer Corinne Emmenegger <emmenegger@stat.math.ethz.ch>

Repository CRAN

Date/Publication 2022-02-03 12:40:02 UTC

R topics documented:

`coef.regsdml` *Accessing the coefficients of regsdml fits*

Description

This is a method for the class `regsdml`. It returns the estimated coefficients from objects of class `regsdml`, which typically result from a function call to `regsdml`.

Usage

```
## S3 method for class 'regsdml'
coef(object,
    print_regsDML = NULL,
    print_safety = NULL,
    print_DML = NULL,
    print_regDML = NULL,
    print_regDML_all_gamma = !is.null(parm),
    parm = NULL,
    print_gamma = FALSE, ...)
```

Arguments

`object`	An object of class `regsdml`. This object usually results from a function call to `regsdml`.
`print_regsDML`	A boolean. If `TRUE`, the results of the `regsDML` method are returned.
`print_safety`	A boolean. If `TRUE`, the results of the safety device are returned.
`print_DML`	A boolean. If `TRUE`, the results of the `DML` method are returned.

print_regDML — A boolean. If TRUE, the results of the regDML method with the optimal choice of gamma (including the factor a_N) are returned.

print_regDML_all_gamma — A boolean. If TRUE, the results specified by parm below are returned.

parm — A vector containing the indices for which gamma-values the results of the regularized DML estimator, whose results are stored in the list regDML_all_gamma_statistics of object, should be included in the output. If parm is specified, it is not necessary to specify print_regDML_all_gamma.

print_gamma — A boolean. If TRUE, the gamma-values are printed in an extra row where the respective regularization methods achieved their optimum.

... — Further arguments passed to or from other methods.

Value

Coefficients of the methods regsDML, the safety device, DML, regDML with the optimal choice of γ (including the factor a_N), and regDML with prespecified γ-values are returned by setting the respective arguments. It is possible to return the respective gamma-values.

If none of the printing arguments are set, only the results of regsDML are returned if they are available. If they are not available and none of the printing arguments are set, the results from all available methods are returned. If print_regsDML = FALSE, only the results from those methods are returned that are explicitly specified by the printing arguments.

See Also

regsdml, summary.regsdml, confint.regsdml, vcov.regsdml print.regsdml

Examples

```
## See example(regsdml) for examples
```

---

confint.mmdml — *Confidence Intervals for coefficient estimates of mmdml fits*

Description

This is a method for the class mmdml. It computes two-sided confidence intervals for testing the two-sided component-wise null hypotheses $H_0 : \beta_j = 0$ with the (approximate) asymptotic Gaussian distribution of the coefficient estimator. The method can be applied to objects of class mmdml that typically result from a function call to mmdml.

Usage

```
## S3 method for class 'mmdml'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

- `object` — An object of class `mmdml`. This object usually results from a function call to `mmdml`.
- `parm` — A vector containing the indices for which β_0-entries confidence intervals should be computed. By default, it is set to `NULL`, in which case confidence intervals for all entries of β_0 are computed.
- `level` — A number between 0 and 1 representing the confidence level. The default is `level = 0.95`.
- `...` — Further arguments passed to or from other methods.

Value

A matrix with columns giving the lower and upper confidence limits for each entry of β_0. The columns are labelled as These will be labelled as `(1-level)/2%` and `1 - (1-level)/2%`, by default 2.5% and 97.5%.

See Also

`mmdml`

Examples

```
## See example(mmdml) for examples
```

`confint.regsdml` *Confidence Intervals for coefficient estimates of regsdml fits*

Description

This is a method for the class `regsdml`. It computes two-sided confidence intervals for testing the two-sided component-wise null hypotheses that tests if a component equals zero with the (approximate) asymptotic Gaussian distribution of the coefficient estimator. The method can be applied to objects of class `regsdml`, which typically result from a function call to `regsdml`.

Usage

```
## S3 method for class 'regsdml'
confint(object,
    parm = NULL,
    level = 0.95,
    print_regsDML = NULL,
    print_safety = NULL,
    print_DML = NULL,
    print_regDML = NULL,
    print_regDML_all_gamma = !is.null(parm),
    print_gamma = FALSE, ...)
```

Arguments

- `object` — An object of class `regsdml`. This object usually results from a function call to `regsdml`.
- `parm` — A vector containing the indices for which `gamma`-values the results of the regularized DML estimator, whose results are stored in the list `regDML_all_gamma_statistics` of `object`, should be included in the output. If `parm` is specified, it is not necessary to specify `print_regDML_all_gamma`.
- `level` — A number between 0 and 1 representing the confidence level. The default is `level = 0.95`.
- `print_regsDML` — A boolean. If `TRUE`, the results of the `regsDML` method are returned.
- `print_safety` — A boolean. If `TRUE`, the results of the safety device are returned.
- `print_DML` — A boolean. If `TRUE`, the results of the `DML` method are returned.
- `print_regDML` — A boolean. If `TRUE`, the results of the `regDML` method with the optimal choice of `gamma` (including the factor `a_N`) are returned.
- `print_regDML_all_gamma` — A boolean. If `TRUE`, the results specified by `parm` below are returned.
- `print_gamma` — A boolean. If `TRUE`, the `gamma`-values are printed in brackets where the respective regularization methods achieved their optimum.
- `...` — Further arguments passed to or from other methods.

Value

Confidence intervals for the methods `regsDML`, the `safety` device, `DML`, `regDML` with the optimal choice of γ (including the factor `a_N`), and `regDML` with prespecified γ-values are returned by setting the respective arguments.

If none of the printing arguments are set, only the results of regsDML are returned if they are available. If they are not available and none of the printing arguments are set, the results from all available methods are returned. If `print_regsDML = FALSE`, only the results from those methods are returned that are explicitly specified by the printing arguments.

See Also

`regsdml`, `summary.regsdml`, `coef.regsdml`, `vcov.regsdml` `print.regsdml`

Examples

```
## See example(regsdml) for examples.
```

dmlalg *dmlalg: double machine learning algorithms*

Description

The dmlalg package contains implementations of double machine learning (DML) algorithms in R.

Partially linear models with confounding variables

Our goal is to perform inference for the linear parameter in partially linear models with confounding variables. The standard DML estimator of the linear parameter has a two-stage least squares interpretation, which can lead to a large variance and overwide confidence intervals. We apply regularization to reduce the variance of the estimator, which produces narrower confidence intervals that are approximately valid. Nuisance terms can be flexibly estimated with machine learning algorithms.

regsdml Estimates the linear parameter in a partially linear model with confounding variables with regularized and standard DML methods.

summary.regsdml A summary method for objects fitted with regsdml.

confint.regsdml A confint method for objects fitted with regsdml.

coef.regsdml A coef method for objects fitted with regsdml.

vcov.regsdml A vcov method for objects fitted with regsdml.

print.regsdml A print method for objects fitted with regsdml.

Partially linear mixed-effects models with repeated measurements

Our goal is to estimate and perform inference for the linear coefficient in a partially linear mixed-effects model with DML. Machine learning algorithms allows us to incorporate more complex interaction structures and high-dimensional variables.

mmdml Estimates the linear parameter in a PLMM with repeated measurements using double machine learning.

confint.mmdml A confint method for objects fitted with mmdml.

fixef.mmdml A fixef method for objects fitted with mmdml.

print.mmdml A print method for objects fitted with mmdml.

ranef.mmdml A ranef method for objects fitted with mmdml.

residuals.mmdml A residuals method for objects fitted with mmdml.

sigma.mmdml A sigma method for objects fitted with mmdml.

summary.mmdml A summary method for objects fitted with mmdml.

vcov.mmdml A vcov method for objects fitted with mmdml.

VarCorr.mmdml A VarCorr method for objects fitted with mmdml.

References

C. Emmenegger and P. Bühlmann. Regularizing Double Machine Learning in Partially Linear Endogenous Models, 2021. Preprint arXiv:2101.12525.

C. Emmenegger and P. Bühlmann. Double Machine Learning for Partially Linear Mixed-Effects Models with Repeated Measurements. Preprint arXiv:2108.13657.

---

example_data_mmdml — *Generate data from partially linear mixed-effects model*

Description

Generate data from a partially linear mixed-effects model with one or two fixed effects, 2 random effects, and 3 nonparametric variables. The true underlying function of the nonparametric variables are step functions. The random effects and error terms are from a Gaussian distribution.

Usage

```
example_data_mmdml(beta0, N = 10L, n = 5L)
```

Arguments

`beta0`	Numeric vector of length 1 or 2 representing the linear coefficient/fixed effects of the model.
`N`	Number of experimental units. Equals 10 per default.
`n`	Expected number of observations per experimental unit, needs to be at least 5. Equals 5 per default.

Value

A data frame with the columns `resp` (the response), `id` and `cask` (random effects), `w1`, `w2`, and `w3` (nonparametric confounders), and `x1` if `beta0` is of length 1 and `x1` and `x2` if `beta0` is of length 2. The random effects are modelled with `"(1|id) + (1|cask:id)"`.

See Also

`mmdml`

Examples

```
## See example(mmdml) for examples
```

lme4-extractors *Extract Components from 'mmdml' Fits Imported from 'lme4'*

Description

Methods for the class mmdml for generics from **lme4**.

Usage

```
fixef(object, ...)
## S3 method for class 'mmdml'
fixef(object, ...)

ranef(object, ...)
## S3 method for class 'mmdml'
ranef(object, ...)

VarCorr(x, sigma = 1, ...)
## S3 method for class 'mmdml'
VarCorr(x, ...)

vcov(object, ...)
## S3 method for class 'mmdml'
vcov(object, ...)
```

Arguments

object, x	An object of class mmdml. This object usually results from a function call to mmdml.
sigma	See lmer from package **lme4**.
...	Further arguments passed to or from other methods.

Details

fixef.mmdml: Extracts the estimator of the linear coefficient β_0, which is a named and numeric vector.

ranef.mmdml: Extracts the random_eff entry from object.

VarCorr.mmdml: The variance and correlation components are computed with the sigma and the theta entries of x as in lmer. For each of the S repetitions, sigma and theta computed on the K sample splits are aggregated by taking the mean. Then, the S mean-aggregated estimates are aggregated by the median. The variance and correlation components are computed with these median-aggregated estimates.

vcov.mmdml: It returns the variance-covariance matrix of the estimator of the linear coefficient is extracted. It is computed based on the asymptotic Gaussian distribution of the estimator. First, for each of the S repetitions, the variance-covariance matrices computed on the K sample splits are aggregated by taking the mean. Second, the S mean-aggregated estimates are aggregated by adding a term correcting for the randomness in the sample splits and by taking the median of these corrected terms. This final corrected and median-aggregated matrix is returned.

Value

See lmer from package **lme4**.

See Also

mmdml

Examples

```
## See example(mmdml) for examples
```

---

mmdml — *Estimating linear coefficients in partially linear mixed-effects models with repeated measurements using double machine learning.*

Description

Our goal is to perform inference for the linear parameter in partially linear mixed-effects models (PLMMs) with repeated measurements using double machine learning (DML).

The function mmdml estimates the linear parameter β_0 in the PLMM

$$Y_i = X_i\beta_0 + g(W_i) + Z_i b_i + \epsilon_{Y_i}, (i = 1, \dots, N)$$

for the continuous response Y_i with linear covariates X_i, nonlinear covariates W_i, unobserved random effects b_i, and the error term ϵ_{Y_i}. For each i, there are n_i repeated observations available. That is, Y_i is an n_i-dimensional vector. The matrix Z_i is fixed. The random effects b_i and the error terms ϵ_{Y_i} are Gaussian distributed, independent, and independent of b_j and ϵ_{Y_j}, respectively, for $i \neq j$. The linear and nonlineare covariates X_i and W_i are random and independent of all random effects and error terms.

The linear parameter β_0 can be estimated with a linear mixed-effects modeling approach with maximum likelihood after regressing out W_i nonparametrically from Y_i and X_i using machine learning algorithms. A linear mixed-effects model is estimated on the partialled-out data

$$Y_i - E[Y_i|W_i] = (X_i - E[X_i|W_i])\beta_0 + Z_i b_i + \epsilon_{Y_i}.$$

The conditional expectations are estimated with machine learning algorithms and sample splitting, and cross-fitting is used to regain full efficiency of the estimator of $beta_0$. This estimator is asymptotically Gaussian distributed and efficient.

Usage

```
mmdml(
  w, x, y, z, data = NULL,
  z_formula = NULL, group = "group",
  K = 2L, S = 100L,
  cond_method = rep("forest", 2),
  params = NULL,
  parallel = c("no", "multicore", "snow"),
  ncpus = 1L, cl = NULL,
  nr_random_eff = if (S > 5) 1L else S,
  nr_res = nr_random_eff
)
```

Arguments

`w`	A vector, matrix, or data frame. Its columns contain observations of the nonlinear predictors. Alternatively, if the data is provided in the data frame `data`, `w` is a character vector whose entries specify the columns of `data` acting as W.
`x`	A vector, matrix, or data frame. This is the linear predictor. Alternatively, if the data is provided in the data frame `data`, `x` is a character vector whose entries specify the columns of `data` acting as X.
`y`	A vector, matrix, or data frame. This is the response. Alternatively, if the data is provided in the data frame `data`, `y` is a character vector whose entries specify the columns of `data` acting as Y.
`z`	A vector, matrix, or data frame. It acts as the fixed matrix assigning the random effects. Alternatively, if the data is provided in the data frame `data`, `z` is a character vector whose entries specify the columns of `data` acting as Z.
`z_formula`	A string specifying the structure of the random effect using the notation as in `lmer` from package **lme4**, e.g., `(1|id) + (1|cask:id)`.
`group`	A string containing the name of the grouping variable in `zz`.
`data`	An optional data frame. If it is specified, its column names need to coincide with the character vectors specified in `a`, `w`, `x`, and `y`.
`K`	The number of sample splits used for cross-fitting.
`S`	Number of replications to correct for the random splitting of the sample. It is set to `100L` by default.
`cond_method`	A character vector of length 2 specifying the estimation methods used to fit the conditional expectations $E[X\|W]$ and $E[Y\|W]$. Its components are from from `"spline"`, `"forest"`, `"ols"`, `"lasso"`, `"ridge"`, and `"elasticnet"`, or it is a list of length 2 with components from `"spline"`, `"forest"`, `"ols"`, `"lasso"`, `"ridge"`, and `"elasticnet"`, and where some components of the list are functions to estimate the conditional expectations. These functions have the input arguments `(yy_fit, ww_fit, ww_predict, params = NULL)` and output the conditional expectation of $E[Y\|W]$ estimated with `yy_fit` and `ww_fit` and predicted with `ww_predict`. The argument `params` is described below. The functions return a matrix where the columns correspond to the component-wise estimated conditional expectations. Here, `yy` symbolically stands for either `x` or `y`.

Please see below for the default arguments of the "spline", "forest", "ols", "lasso", "ridge", and "elasticnet" methods.

params
: An optional list of length 2. The 2 elements of this list are lists themselves. These lists specify additional input arguments for estimating the conditional expectations $E[X|W]$ and $E[Y|W]$, respectively.

parallel
: One out of "no", "multicore", or "snow" specifying the parallelization method used to compute the S replications. The default is "no".

ncpus
: An integer specifying the number of cores used if parallel is not set to "no".

cl
: An optional parallel or snow cluster if parallel = "snow". The argument ncpus does not have to be specified if the argument cl is specified.

nr_random_eff
: An integer specifying the number of unaggregated sets of random effect estimates among the S repetitions that should be returned.

nr_res
: An integer specifying the number of unaggregated sets of residual estimates among the S repetitions that should be returned.

Details

The estimator of β_0 is computed using sample splitting and cross-fitting. The subject-specific data (over $i = 1, \ldots, N$) is split into K sets that are equally large if possible. For each such set, the nuisance parameters (that is, the conditional expectations $E[Y_i|W_i]$ and $E[X_i|W_i]$) are estimated on its complement and evaluated on the set itself. Estimators of β_0 and the variance parameters are computed for each of the K data sets and are then averaged. If K = 1, no sample splitting is performed. In this case, the nuisance parameters are estimated and predicted on the full sample.

The whole estimation procedure is repeated S times to account for the randomness introduced by the random sample splits. The S repetitions can be run in parallel by specifying the arguments parallel and ncpus. The S estimators of β_0 and the variance components are aggregated by taking the median of them. The S variance-covariance matrices of the estimator of β_0 are aggregated by first adding a correction term to them that accounts for the random splitting and by afterwards taking the median of the corrected variance-covariance matrices. If $d > 1$, it can happen that this final matrix is not positive definite anymore, in which case the mean is considered instead. Estimates of the conditional random effects and the residuals are computed in each of the S repetitions. A total number of nr_random_eff and nr_res of them, respectively, is returned. Additionally, the random effects estimates from all S repetitions are aggregated using the mean and returned.

If the design in at least 0.5 * S of the S repetitions is singular, an error message is displayed. If the designs in some but less than 0.5 * S of the S repetitions are singular, another S repetitions are performed. If, in total, at least S repetitions result in a nonsingular design, the results are returned together with a warning message.

The default options of the "spline", "forest", "ols", "lasso", "ridge", and "elasticnet" methods are as follows. With the "spline" method, the function bs from the package splines is employed with degree = 3 and df = ceiling(N ^ (1 / 5)) + 2 if N satisfies (df + 1) * v + 1 > N, where v denotes the number of columns of w and N denotes the sample size. Otherwise, df is consecutively reduced by 1 until this condition is satisfied. The splines are fitted and predicted on different data sets. If they are extrapolated, a warning message is displayed. With the "forest" method, the function randomForest from the package randomForest is employed with nodesize = 5, ntree = 500, na.action = na.omit, and replace = TRUE. With the "ols" method, the default arguments are used and no additional arguments are specified. With the "lasso" and "ridge" methods, the function cv.glmnet from the package glmnet performs 10-fold cross validation by default (argument nfolds) to find the one-standard-error-rule λ-parameter. With the "elasticnet" method, the function cv.glmnet from the package glmnet performs 10-fold cross validation (argument nfolds) with alpha = 0.5 by default to find the one-standard-error-rule λ-parameter. All default values of the mentioned parameters can be adapted by specifying the argument params.

There are three possibilities to set the argument parallel, namely "no" for serial evaluation (default), "multicore" for parallel evaluation using forking, and "snow" for parallel evaluation using a parallel socket cluster. It is recommended to select RNGkind ("L'Ecuyer-CMRG") and to set a seed to ensure that the parallel computing of the package dmlalg is reproducible. This ensures that each processor receives a different substream of the pseudo random number generator stream. Thus, the results reproducible if the arguments remain unchanged. There is an optional argument cl to specify a custom cluster if parallel = "snow".

The response y needs to be continuous. The covariate w may contain factor variables in its columns. If the variable x contains factor variables, the factors should not be included as factor columns of x. Instead, dummy encoding should be used for all individual levels of the factor. That is, a factor with 4 levels should be encoded with 4 columns where each column consists of 1 and 0 entries indicating the presence of the respective level of the factor.

There are confint, fixef, print, ranef, residuals, sigma, summary, vcov, and VarCorr methods available for objects fitted with mmdml. They are called confint.mmdml, fixef.mmdml, print.mmdml, ranef.mmdml, residuals.mmdml, sigma.mmdml, summary.mmdml, vcov.mmdml, and VarCorr.mmdml, respectively.

Value

A list similar to the output of lmer from package **lme4** containing the following entries.

beta	Estimator of the linear coefficient β_0.
vcov	Variance-covariance matrix of beta. Also see lmer. The S individual variance-covariance matrices are aggregated by first adding a correction term to them correcting for the randomness of the sample splits and by subsequently taking the median of the corrected variance-covariance matrices.
sigma	Please see lmer for its meaning. It is computed by averaging over the K sample splits and by aggregating the S repetitions using the median.
theta	Please see lmer for its meaning. It is computed by averaging over the K sample splits and by aggregating the S repetitions using the median.
varcor	Variance correlation components computed with theta. Please also see lmer.
random_eff	Conditional estimates of the random effects similarly to lmer. The individual sets of S random effects estimates are aggregated using the mean.
random_eff_all	The first nr_random_eff sets of the S sets of random effects estimates.
residuals	The first nr_res sets of the S sets of residuals. Each set of residuals is computed with parameters and data that is aggregated over the K sample splits.

The other elements ngrps, nobs, fitMsgs, cnms, nc, nms, useSc, optinfo, and methTitle are as in lmer. The gradient and Hessian information of optinfo is computed by aggregating the respective information over the S repetitions with the median.

References

C. Emmenegger and P. Bühlmann. Double Machine Learning for Partially Linear Mixed-Effects Models with Repeated Measurements. Preprint arXiv:2108.13657.

See Also

confint, fixef, print, ranef, residuals, sigma, summary, vcov, VarCorr

Examples

```
## generate data
RNGkind("L'Ecuyer-CMRG")
set.seed(19)
data1 <- example_data_mmdml(beta0 = 0.2)
data2 <- example_data_mmdml(beta0 = c(0.2, 0.2))

## fit models
## Caveat: Warning messages are displayed because the small number of
## observations results in a singular random effects model
fit1 <-
  mmdml(w = c("w1", "w2", "w3"), x = "x1", y = "resp", z = c("id", "cask"),
        data = data1, z_formula = "(1|id) + (1|cask:id)", group = "id", S = 3)

fit2 <-
  mmdml(w = c("w1", "w2", "w3"), x = c("x1", "x2"), y = "resp", z = c("id", "cask"),
        data = data2, z_formula = "(1|id) + (1|cask:id)", group = "id", S = 3)

## apply methods
confint(fit2)
fixef(fit2)
print(fit2)
ranef(fit2)
residuals(fit2)
sigma(fit2)
summary(fit2)
vcov(fit2)
VarCorr(fit2)
```

---

print.mmdml *Printing mmdml fits*

Description

This is a method for the class mmdml. It prints objects of class mmdml that typically result from a function call to mmdml.

Usage

```
## S3 method for class 'mmdml'
print(x, digits = max(3, getOption("digits") - 3),
    ranef.comp = "Std.Dev.", ...)
```

Arguments

`x`	An object of class `mmdml`. This object usually results from a function call to `mmdml`.
`digits`	Number of significant digits for printing; also see `lmer` from package **lme4**.
`ranef.comp`	A character vector of length one or two indicating if random-effects parameters should be reported on the variance and/or standard deviation scale; also see `lmer`.
`...`	Further arguments passed to or from other methods.

Value

See `lmer`.

See Also

`mmdml`

Examples

```
## See example(mmdml) for examples
```

---

print.regsdml *Printing regsdml fits*

Description

This is a method for the class `regsdml`. It prints objects of class `regsdml`, which typically result from a function call to `regsdml`.

Usage

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

Arguments

`x`	An object of class `regsdml`. This object usually results from a function call to `regsdml`.
`...`	Further arguments passed to or from other methods.

Value

By default, `summary(x)` is called. Please see `summary.regsdml` for further details.

See Also

`regsdml`, `summary.regsdml`, `confint.regsdml`, `coef.regsdml`, `vcov.regsdml`

Examples

```
## Generate some data:
set.seed(19)
# true linear parameter
beta0 <- 1
n <- 40
# observed confounder
w <- pi * runif(n, -1, 1)
# instrument
a <- 3 * tanh(2 * w) + rnorm(n, 0, 1)
# unobserved confounder
h <- 2 * sin(w) + rnorm(n, 0, 1)
# linear covariate
x <- -1 * abs(a) - h - 2 * tanh(w) + rnorm(n, 0, 1)
# response
y <- beta0 * x - 3 * cos(pi * 0.25 * h) + 0.5 * w ^ 2 + rnorm(n, 0, 1)

## Estimate the linear coefficient from x to y
## (The parameters are chosen small enough to make estimation fast):
## Caveat: A spline estimator is extrapolated, which raises a warning message.
## Extrapolation lies in the nature of our method. To omit the warning message
## resulting from the spline estimator, another estimator may be used.
fit <- regsdml(a, w, x, y,
               gamma = exp(seq(-4, 1, length.out = 4)),
               S = 3,
               do_regDML_all_gamma = TRUE,
               cond_method = c("forest",  # for E[A|W]
                               "spline",  # for E[X|W]
                               "spline"), # for E[Y|W]
               params = list(list(ntree = 1), NULL, NULL))
print(fit)
```

`regsdml` *Estimating linear coefficients with double machine learning (DML)*

Description

Our goal is to perform inference for the linear parameter in partially linear models with confounding variables. The standard double machine learning (DML) estimator of the linear parameter has a two-stage least squares interpretation, which can lead to a large variance and overwide confidence intervals. We apply regularization to reduce the variance of the estimator, which produces narrower confidence intervals that remain approximately valid.

The function `regsdml` estimates the linear parameter β_0 in the partially linear model

$$Y = X^T\beta_0 + g(W) + h(H) + \epsilon_Y$$

of the continuous response Y with linear covariates X, nonlinear covariates W, unobserved confounding variables H, and the error term ϵ_Y. An additional variable A is required that is not part of the right-hand side defining Y. The variable A acts as an instrument after W is regressed out of it.

The linear parameter β_0 can be estimated with a two-stage least squares (TSLS) approach ("standard" DML) or with regularized approaches (regDML, regsDML). All approaches use double machine learning. The TSLS approach regresses the residual $Y - E[Y|W]$ on $X - E[X|W]$ using the instrument $A - E[A|W]$. The regularized approaches minimize an objective function that equals γ times the objective function of TSLS plus an objective function that partials out $A - E[A|W]$ from the residual quantity $Y - E[Y|W] - (X - E[X|W])^T\beta$. The different regularization approaches choose different regularization parameters γ. The conditional expectations act as nuisance parameters and are estimated with machine learning algorithms. All approaches use sample splitting and cross-fitting to estimate β_0.

Usage

```
regsdml(
  a, w, x, y, data = NULL,
  DML = c("DML2", "DML1"),
  K = 2L,
  gamma = exp(seq(-4, 10, length.out = 100)),
  aN = NULL,
  do_regsDML = TRUE,
  do_safety = FALSE,
  do_DML = do_regDML || do_regsDML || do_safety,
  do_regDML = FALSE,
  do_regDML_all_gamma = FALSE,
  safety_factor = 0.7,
  cond_method = rep("spline", 3),
  params = NULL,
  level = 0.95,
  S = 100L,
  parallel = c("no", "multicore", "snow"),
  ncpus = 1L,
  cl = NULL
)
```

Arguments

`a` A vector, matrix, or data frame. It acts as an instrument after regressing out `w` of it. Alternatively, if the data is provided in the data frame `data`, `a` is a character vector whose entries specify the columns of `data` acting as "instrument" A.

w	A vector, matrix, or data frame. Its columns contain observations of the nonlinear predictors. Alternatively, if the data is provided in the data frame `data`, `w` is a character vector whose entries specify the columns of `data` acting as W.
x	A vector, matrix, or data frame. This is the linear predictor. Alternatively, if the data is provided in the data frame `data`, `x` is a character vector whose entries specify the columns of `data` acting as X.
y	A vector, matrix, or data frame. This is the response. Alternatively, if the data is provided in the data frame `data`, `y` is a character vector whose entries specify the columns of `data` acting as Y.
data	An optional data frame. If it is specified, its column names need to coincide with the character vectors specified in `a`, `w`, `x`, and `y`.
DML	Either `"DML2"` or `"DML1"` depending on which DML method should be used. The default is `"DML2"`.
K	The number of sample splits used for cross-fitting.
gamma	A vector specifying the grid of regularization parameters over which to optimize.
aN	The Nth element of a sequence of non-negative real numbers diverging to $+\infty$ as the sample size N tends to $+\infty$. By default, it equals `max(log(sqrt(N)), 1)`, where `N` denotes the sample size.
do_regsDML	A boolean that specifies whether the regsDML estimator is computed. It is set to `TRUE` by default.
do_safety	A boolean that specifies whether a safety device is employed. The safety device chooses the regularization parameter γ such that the variance of the regularized estimator is at least (100 * `safety_factor`)% of the variance of standard DML.
do_DML	A boolean that specifies whether the standard DML estimator is computed. It is set to `TRUE` by default if at least one of `do_regsDML`, `do_safety`, or `do_regDML` is set to `TRUE`.
do_regDML	A boolean that specifies whether the regularized DML estimator regDML with the regularization parameter equal to `a_N` times the γ leading to the lowest mean squared error is computed. It is set to `FALSE` by default.
do_regDML_all_gamma	A boolean that specifies whether the regularized estimators for all values γ of the grid `gamma` are returned. It is set to `FALSE` by default.
safety_factor	The factor of the safety method. It is set to `0.7` by default.
cond_method	A character vector of length 3 specifying the estimation methods used to fit the conditional expectations $E[A\|W]$, $E[X\|W]$, and $E[Y\|W]$. Its components are from from `"spline"`, `"forest"`, `"ols"`, `"lasso"`, `"ridge"`, and `"elasticnet"`, or it is a list of length 3 with components from `"spline"`, `"forest"`, `"ols"`, `"lasso"`, `"ridge"`, and `"elasticnet"`, and where some components of the list are functions to estimate the conditional expectations. These functions have the input arguments `(yy_fit, ww_fit, ww_predict, params = NULL)` and output the conditional expectation of $E[Y\|W]$ estimated with `yy_fit` and `ww_fit` and predicted with `ww_predict`. The argument `params` is described below. The functions return a matrix where the columns correspond to the component-wise estimated conditional expectations. Here, `yy` symbolically stands for either `a`, `x`, or `y`. Please see below for the default arguments of the `"spline"`, `"forest"`, `"ols"`, `"lasso"`, `"ridge"`, and `"elasticnet"` methods.

params	An optional list of length 3. All 3 elements of this list are lists themselves. These lists specify additional input arguments for estimating the conditional expectations $E[A\|W]$, $E[X\|W]$, and $E[Y\|W]$, respectively.
level	Level for computing confidence intervals for testing the two-sided component-wise null hypotheses that test if a component equals zero with the (approximate) asymptotic Gaussian distribution. The default is 0.95.
S	Number of replications to correct for the random splitting of the sample. It is set to 100L by default.
parallel	One out of "no", "multicore", or "snow" specifying the parallelization method used to compute the S replications. The default is "no".
ncpus	An integer specifying the number of cores used if parallel is not set to "no".
cl	An optional parallel or snow cluster if parallel = "snow". The argument ncpus does not have to be specified if the argument cl is specified.

Details

The estimator of β_0 is computed using sample splitting and cross-fitting. Irrespective of which methods are performed, the data is split into K sets that are equally large if possible. For each such set, the nuisance parameters (that is, the conditional expectations $E[A|W]$, $E[X|W]$, and $E[Y|W]$) are estimated on its complement and evaluated on the set itself. If DML = "DML1", then K individual estimators are computed for each of the K data sets and are then averaged. If DML = "DML2", the nuisance parameter matrices are first assembled before the estimator of β_0 is computed. This enhances stability of the coefficient estimator compared to "DML1". If K = 1, no sample splitting is performed. In this case, the nuisance parameters are estimated and predicted on the full sample.

The whole estimation procedure can be repeated S times to account for the randomness introduced by the random sample splits. The S repetitions can be run in parallel by specifying the arguments parallel and ncpus. The S estimators of β_0 are aggregated by taking the median of them. The S variance-covariance matrices are aggregated by first adding a correction term to them that accounts for the random splitting and by afterwards taking the median of the corrected variance-covariance matrices. If $d > 1$, it can happen that this final matrix is not positive definite anymore, in which case the mean is considered instead.

If the design in at least 0.5 * S of the S repetitions is singular, an error message is displayed. If the designs in some but less than 0.5 * S of the S repetitions are singular, another S repetitions are performed. If, in total, at least S repetitions result in a nonsingular design, the results are returned together with a warning message.

The regularized estimators and their associated mean squared errors (MSEs) are computed for the regularization parameters γ of the grid gamma. These estimators are returned if the argument do_regDML_all_gamma is set to TRUE. The γ-value whose corresponding regularized estimator from the do_regDML_all_gamma method achieves the smallest MSE is multiplied by aN, leading to γ'. The do_regDML_all_gamma estimator with regularization parameter γ' is called regDML. The regsDML estimator equals regDML or DML depending on whose variance is smaller. If β_0 is of larger dimension than 1, the MSE computations and the variance comparison step are performed with the sum of the diagonal entries of the respective variance-covariance matrices.

If do_safety = TRUE, a γ value is chosen such that the regularized estimator among do_regDML_all_gamma with this value of γ has a variance that is just not smaller than safety_factor times the variance of DML. If β_0 is of larger dimension than 1, the sum of the diagonal entries of the respective variance-covariance matrices is taken as a measure of variance. If the regularization scheme leads to considerable variance reductions, it is possible that this safety device cannot be applied. In this case, a respective message is returned.

The default options of the `"spline"`, `"forest"`, `"ols"`, `"lasso"`, `"ridge"`, and `"elasticnet"` methods are as follows. With the `"spline"` method, the function `bs` from the package `splines` is employed with `degree = 3` and `df = ceiling(N ^ (1 / 5)) + 2` if `N` satisfies `(df + 1) * v + 1 > N`, where `v` denotes the number of columns of `w` and `N` denotes the sample size. Otherwise, `df` is consecutively reduced by `1` until this condition is satisfied. The splines are fitted and predicted on different data sets. If they are extrapolated, a warning message is displayed. With the `"forest"` method, the function `randomForest` from the package `randomForest` is employed with `nodesize = 5`, `ntree = 500`, `na.action = na.omit`, and `replace = TRUE`. With the `"ols"` method, the default arguments are used and no additional arguments are specified. With the `"lasso"` and `"ridge"` methods, the function `cv.glmnet` from the package `glmnet` performs 10-fold cross validation by default (argument `nfolds`) to find the one-standard-error-rule λ-parameter. With the `"elasticnet"` method, the function `cv.glmnet` from the package `glmnet` performs 10-fold cross validation (argument `nfolds`) with `alpha = 0.5` by default to find the one-standard-error-rule λ-parameter. All default values of the mentioned parameters can be adapted by specifying the argument `params`.

There are three possibilities to set the argument `parallel`, namely `"no"` for serial evaluation (default), `"multicore"` for parallel evaluation using forking, and `"snow"` for parallel evaluation using a parallel socket cluster. It is recommended to select RNGkind (`"L'Ecuyer-CMRG"`) and to set a seed to ensure that the parallel computing of the package `dmlalg` is reproducible. This ensures that each processor receives a different substream of the pseudo random number generator stream. Thus, the results reproducible if the arguments remain unchanged. There is an optional argument `cl` to specify a custom cluster if `parallel = "snow"`.

The response `y` needs to be continuous. The covariate `w` may contain factor variables in its columns. If the variables `a` and `x` contain factor variables, the factors should not be included as factor columns of `a` or `x`. Instead, dummy encoding should be used for all individual levels of the factor. That is, a factor with 4 levels should be encoded with 4 columns where each column consists of 1 and 0 entries indicating the presence of the respective level of the factor.

There are `summary`, `confint`, `coef`, `vcov`, and `print` methods available for objects fitted with `regsdml`. They are called `summary.regsdml`, `confint.regsdml`, `coef.regsdml`, `vcov.regsdml`, and `print.regsdml`, respectively.

Value

A list containing some of the lists `regsDML_statistics`, `regDML_safety_statistics`, `DML_statistics`, `regDML_statistics`, and `regDML_all_gamma_statistics` is returned. The individual sublists contain the following arguments supplemented by an additional suffix specifying the method they correspond to.

`beta`	Estimator of the linear coefficient β_0.
`sd`	Standard error estimates of the respective entries of `beta`.
`var`	Variance-covariance matrix of `beta`.
`pval`	p-values for the respective entries of `beta`.
`CI`	Two-sided confidence intervals for β_0 where the jth row of `CI` corresponds to the two-sided testing of $H_0 : (\beta_0)_j = 0$ at level `level`. They are computed with the (approximate) asymptotic Gaussian distribution of the coefficient estimates.

The list `regsDML_statistics` contains the following additional entries:

`message_regsDML`
: Specifies if regsDML selects the regularized estimator or DML.

`gamma_aN`
: Chosen optimal regularization parameter if regsDML equals the regularized estimator. This entry is not present if DML is selected.

If the safety device is applicable, the list `regDML_safety_statistics` contains the following additional entries:

`message_safety`
: Specifies whether the safety device was applicable.

`gamma_safety`
: Chosen regularization parameter of the safety device.

If the safety device is not applicable, the list `regDML_safety_statistics` contains `message_safety` as its only entry.

The list `regDML_statistics` contains the following additional entry:

`gamma_opt`
: Chosen optimal regularization parameter.

The list `regDML_all_gamma_statistics` is a list of the same length as the grid `gamma`, where each individual list is of the structure just described.

References

C. Emmenegger and P. Bühlmann. Regularizing Double Machine Learning in Partially Linear Endogenous Models, 2021. Preprint arXiv:2101.12525.

See Also

`summary.regsdml`, `confint.regsdml`, `coef.regsdml`, `vcov.regsdml` `print.regsdml`

Examples

```
## Generate some data:
RNGkind("L'Ecuyer-CMRG")
set.seed(19)
# true linear parameter
beta0 <- 1
n <- 40
# observed confounder
w <- pi * runif(n, -1, 1)
# instrument
a <- 3 * tanh(2 * w) + rnorm(n, 0, 1)
# unobserved confounder
h <- 2 * sin(w) + rnorm(n, 0, 1)
# linear covariate
x <- -1 * abs(a) - h - 2 * tanh(w) + rnorm(n, 0, 1)
# response
y <- beta0 * x - 3 * cos(pi * 0.25 * h) + 0.5 * w ^ 2 + rnorm(n, 0, 1)

## Estimate the linear coefficient from x to y
```

```
## (The parameters are chosen small enough to make estimation fast):
## Caveat: A spline estimator is extrapolated, which raises a warning message.
## Extrapolation lies in the nature of our method. To omit the warning message
## resulting from the spline estimator, another estimator may be used.
fit <- regsdml(a, w, x, y,
               gamma = exp(seq(-4, 1, length.out = 4)),
               S = 3,
               do_regDML_all_gamma = TRUE,
               cond_method = c("forest",  # for E[A|W]
                               "spline",  # for E[X|W]
                               "spline"), # for E[Y|W]
               params = list(list(ntree = 1), NULL, NULL))
## parm = c(2, 3) prints an additional summary for the 2nd and 3rd gamma-values
summary(fit, parm = c(2, 3),
        correlation = TRUE,
        print_gamma = TRUE)
confint(fit, parm = c(2, 3),
        print_gamma = TRUE)
coef(fit) # coefficients
vcov(fit) # variance-covariance matrices

## Alternatively, provide the data in a single data frame
## (see also caveat above):
data <- data.frame(a = a, w = w, x = x, y = y)
fit <- regsdml(a = "a", w = "w", x = "x", y = "y", data = data,
               gamma = exp(seq(-4, 1, length.out = 4)),
               S = 3)

## With more realistic parameter choices:
if (FALSE) {
  fit <- regsdml(a, w, x, y,
                 cond_method = c("forest",  # for E[A|W]
                                 "spline",  # for E[X|W]
                                 "spline")) # for E[Y|W]
  summary(fit)
  confint(fit)

  ## Alternatively, provide the data in a single data frame:
  ## (see also caveat above):
  data <- data.frame(a = a, w = w, x = x, y = y)
  fit <- regsdml(a = "a", w = "w", x = "x", y = "y", data = data)
}
```

`residuals.mmdml` *Confidence Intervals for coefficient estimates of regsDML fits*

Description

A list whose elements correspond to the potentially scaled first `nr_res` sets of residuals of the `S` residuals.

Usage

```
## S3 method for class 'mmdml'
residuals(object, scaled = FALSE, ...)
```

Arguments

`object`	An object of class `mmdml`. This object usually results from a function call to `mmdml`.
`scaled`	A boolean specifying whether scaled residuals should be returned. It is set to FALSE by default.
`...`	Further arguments passed to or from other methods.

Value

A list whose elements correspond to the first `nr_res` sets of residuals of the `S` residuals.

See Also

`mmdml`

Examples

```
## See example(mmdml) for examples
```

`sigma.mmdml` *Extract Residual Standard Deviation 'Sigma' from mmdml Fits*

Description

Extract the estimated standard deviation of the errors, the "residual standard deviation", from a fitted `mmdml` model.

Usage

```
## S3 method for class 'mmdml'
sigma(object, ...)
```

Arguments

`object`	An object of class `mmdml`. This object usually results from a function call to `mmdml`.
`...`	Further arguments passed to or from other methods.

Value

A number representing the estimated standard deviation. First, for each of the `S` repetitions, the standard deviations computed on the `K` sample splits are aggregated by taking the mean. Second, the `S` mean-aggregated estimates are aggregated by the median. This final value is returned.

See Also

mmdml

Examples

```
## See example(mmdml) for examples
```

---

summary.mmdml *Summarizing mmdml fits*

Description

This is a method for the class mmdml. It summarizes objects of class mmdml that typically result from a function call to mmdml.

Usage

```
## S3 method for class 'mmdml'
summary(object,
    correlation = (p <= getOption("lme4.summary.cor.max")),
    nr_res = NULL, ...)
```

Arguments

object	An object of class mmdml. This object usually results from a function call to mmdml.
correlation	Boolean indicating if the variance and correlation components (vcov, VarCorr) should be printed.
nr_res	Boolean indicating how many sets of residuals among the S should be used to compute the residual information. By default, all available sets, that is, nr_res many that are saved in object resulting from mmdml are used.
...	Further arguments passed to or from other methods.

Value

Prints a summary output similar to lmer from package **lme4**.

See Also

mmdml

Examples

```
## See example(mmdml) for examples
```

`summary.regsdml` *Summarizing regsdml fits*

Description

This is a method for the class `regsdml`. It summarizes objects of class `regsdml`, which typically result from a function call to `regsdml`.

Usage

```
## S3 method for class 'regsdml'
summary(object,
    print_regsDML = NULL,
    print_safety = NULL,
    print_DML = NULL,
    print_regDML = NULL,
    print_regDML_all_gamma = !is.null(parm),
    parm = NULL,
    correlation = FALSE,
    print_gamma = FALSE, ...)
```

Arguments

`object`	An object of class `regsdml`. This object usually results from a function call to `regsdml`.
`print_regsDML`	A boolean. If `TRUE`, the results of the `regsDML` method are returned.
`print_safety`	A boolean. If `TRUE`, the results of the safety device are returned.
`print_DML`	A boolean. If `TRUE`, the results of the `DML` method are returned.
`print_regDML`	A boolean. If `TRUE`, the results of the `regDML` method with the optimal choice of `gamma` (including the factor `a_N`) are returned.
`print_regDML_all_gamma`	A boolean. If `TRUE`, the results specified by `parm` below are returned.
`parm`	A vector containing the indices for which `gamma`-values the results of the regularized DML estimator, whose results are stored in the list `regDML_all_gamma_statistics` of `object`, should be included in the summary output. If `parm` is specified, it is not necessary to specify `print_regDML_all_gamma`.
`correlation`	A boolean. If `TRUE`, the variance-covariance matrices of the coefficient estimates are displayed.
`print_gamma`	A boolean. If `TRUE`, the `gamma`-values are printed in brackets where the respective regularization methods achieved their optimum.
`...`	Further arguments passed to or from other methods.

Value

Summary statistics of the methods regsDML, the safety device, DML, regDML with the optimal choice of γ (including the factor a_N), and regDML with prespecified γ-values are returned by setting the respective arguments. It is possible to return the respective gamma-values and variance-covariance matrices.

If none of the printing arguments are set, only the results of regsDML are returned if they are available. If they are not available and none of the printing arguments are set, the results from all available methods are returned. If print_regsDML = FALSE, only the results from those methods are returned that are explicitly specified by the printing arguments.

See Also

regsdml, confint.regsdml, coef.regsdml, vcov.regsdml print.regsdml

Examples

```
## See example(regsdml) for examples
```

---

vcov.regsdml   *Accessing the variance-covariance matrices of regsdml fits*

Description

This is a method for the class regsdml. It returns the variance-covariance matrices of the coefficients from objects of class regsdml, which typically result from a function call to regsdml.

Usage

```
## S3 method for class 'regsdml'
vcov(object,
    print_regsDML = NULL,
    print_safety = NULL,
    print_DML = NULL,
    print_regDML = NULL,
    print_regDML_all_gamma = !is.null(parm),
    parm = NULL,
    print_gamma = FALSE, ...)
```

Arguments

object	An object of class regsdml. This object usually results from a function call to regsdml.
print_regsDML	A boolean. If TRUE, the results of the regsDML method are returned.
print_safety	A boolean. If TRUE, the results of the safety device are returned.
print_DML	A boolean. If TRUE, the results of the DML method are returned.

print_regDML — A boolean. If TRUE, the results of the regDML method with the optimal choice of gamma (including the factor a_N) are returned.

print_regDML_all_gamma — A boolean. If TRUE, the results specified by parm below are returned.

parm — A vector containing the indices for which gamma-values the results of the regularized DML estimator, whose results are stored in the list regDML_all_gamma_statistics of object, should be included in the output. If parm is specified, it is not necessary to specify print_regDML_all_gamma.

print_gamma — A boolean. If TRUE, the gamma-values are printed in an extra row where the respective regularization methods achieved their optimum.

... — Further arguments passed to or from other methods.

Value

Variance-covariance matrices of the methods regsDML, the safety device, DML, regDML with the optimal choice of γ (including the factor a_N), and regDML with prespecified γ-values are returned by setting the respective arguments. It is possible to return the respective gamma-values.

If none of the printing arguments are set, only the results of regsDML are returned if they are available. If they are not available and none of the printing arguments are set, the results from all available methods are returned. If print_regsDML = FALSE, only the results from those methods are returned that are explicitly specified by the printing arguments.

See Also

regsdml, summary.regsdml, confint.regsdml, coef.regsdml print.regsdml

Examples

```
## See example(regsdml) for examples
```

Index