Package ‘GenericML’

November 24, 2021

Title  Generic Machine Learning Inference

Version  0.1.0

Author  Max Welz [aut, cre] (<https://orcid.org/0000-0003-2945-1860>),
        Andreas Alfons [aut] (<https://orcid.org/0000-0002-2513-3788>),
        Mert Demirer [aut],
        Victor Chernozhukov [aut]

Maintainer  Max Welz <welz@ese.eur.nl>

Description  Generic Machine Learning Inference on heterogeneous treatment effects in randomized experiments as proposed in Chernozhukov, Demirer, Duflo and Fernández-Val (2020) <arXiv:1712.04802>. This package's workhorse is the 'mlr3' framework of Lang et al. (2019) <doi:10.21105/joss.01903>, which enables the specification of a wide variety of machine learners. The main functionality, GenericML(), runs Algorithm 1 in Chernozhukov, Demirer, Duflo and Fernández-Val (2020) <arXiv:1712.04802> for a suite of user-specified machine learners. All steps in the algorithm are customizable via setup functions. Methods for printing and plotting are available for objects returned by GenericML(). Parallel computing is supported.

License  GPL (>= 3)

Encoding  UTF-8

RoxygenNote  7.1.1

URL  https://github.com/mwelz/GenericML/

BugReports  https://github.com/mwelz/GenericML/issues/

Depends  ggplot2, mlr3, mlr3learners

Imports  sandwich, lmtest, splitstackshape, stats, parallel

Suggests  glmnet, ranger, rpart, e1071, testthat (>= 3.0.0)

Config/testthat/edition  3

NeedsCompilation  no

Repository  CRAN

Date/Publication  2021-11-24 08:10:18 UTC
## R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLP</td>
<td>2</td>
</tr>
<tr>
<td>CLAN</td>
<td>4</td>
</tr>
<tr>
<td>GATES</td>
<td>5</td>
</tr>
<tr>
<td>GenericML</td>
<td>7</td>
</tr>
<tr>
<td>GenericML_single</td>
<td>12</td>
</tr>
<tr>
<td>get_BLP</td>
<td>15</td>
</tr>
<tr>
<td>get_CLAN</td>
<td>16</td>
</tr>
<tr>
<td>get_GATES</td>
<td>17</td>
</tr>
<tr>
<td>lambda_parameters</td>
<td>19</td>
</tr>
<tr>
<td>Med</td>
<td>20</td>
</tr>
<tr>
<td>plot.GenericML</td>
<td>21</td>
</tr>
<tr>
<td>print.GenericML</td>
<td>23</td>
</tr>
<tr>
<td>propensity_score</td>
<td>24</td>
</tr>
<tr>
<td>proxy_BCA</td>
<td>25</td>
</tr>
<tr>
<td>proxy_CATE</td>
<td>27</td>
</tr>
<tr>
<td>quantile_group</td>
<td>28</td>
</tr>
<tr>
<td>setup_diff</td>
<td>29</td>
</tr>
<tr>
<td>setup_vcov</td>
<td>30</td>
</tr>
<tr>
<td>setup_X1</td>
<td>32</td>
</tr>
<tr>
<td>TrueIfUnix</td>
<td>33</td>
</tr>
</tbody>
</table>

## Index

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLP</td>
<td>2</td>
</tr>
<tr>
<td>CLAN</td>
<td>4</td>
</tr>
<tr>
<td>GATES</td>
<td>5</td>
</tr>
<tr>
<td>GenericML</td>
<td>7</td>
</tr>
<tr>
<td>GenericML_single</td>
<td>12</td>
</tr>
<tr>
<td>get_BLP</td>
<td>15</td>
</tr>
<tr>
<td>get_CLAN</td>
<td>16</td>
</tr>
<tr>
<td>get_GATES</td>
<td>17</td>
</tr>
<tr>
<td>lambda_parameters</td>
<td>19</td>
</tr>
<tr>
<td>Med</td>
<td>20</td>
</tr>
<tr>
<td>plot.GenericML</td>
<td>21</td>
</tr>
<tr>
<td>print.GenericML</td>
<td>23</td>
</tr>
<tr>
<td>propensity_score</td>
<td>24</td>
</tr>
<tr>
<td>proxy_BCA</td>
<td>25</td>
</tr>
<tr>
<td>proxy_CATE</td>
<td>27</td>
</tr>
<tr>
<td>quantile_group</td>
<td>28</td>
</tr>
<tr>
<td>setup_diff</td>
<td>29</td>
</tr>
<tr>
<td>setup_vcov</td>
<td>30</td>
</tr>
<tr>
<td>setup_X1</td>
<td>32</td>
</tr>
<tr>
<td>TrueIfUnix</td>
<td>33</td>
</tr>
</tbody>
</table>

## BLP

**Performs BLP regression**

### Description

Performs the linear regression for the Best Linear Predictor (BLP) procedure.

### Usage

```r
BLP(
    Y, 
    D, 
    propensity_scores, 
    proxy_BCA, 
    proxy_CATE, 
    HT = FALSE, 
    X1_control = setup_X1(), 
    vcov_control = setup_vcov(), 
    significance_level = 0.05 
)
```
Arguments

Y
- A numeric vector containing the response variable.

D
- A binary vector of treatment assignment. Value one denotes assignment to the treatment group and value zero assignment to the control group.

propensity_scores
- A numeric vector of propensity scores. We recommend to use the estimates of a propensity_score object.

proxy_BCA
- A numeric vector of proxy baseline conditional average (BCA) estimates. We recommend to use the estimates of a proxy_BCA object.

proxy_CATE
- A numeric vector of proxy conditional average treatment effect (CATE) estimates. We recommend to use the estimates of a proxy_CATE object.

HT
- Logical. If TRUE, a Horvitz-Thompson (HT) transformation is applied (BLP2 in the paper). Default is FALSE.

X1_control
- Specifies the design matrix \(X_1\) in the regression. Must be an instance of setup_X1. See the documentation of setup_X1 for details.

vcov_control
- Specifies the covariance matrix estimator. Must be an instance of setup_vcov. See the documentation of setup_vcov for details.

significance_level
- Significance level. Default is 0.05.

Value

An object of class BLP, consisting of the following components:

generic_targets
- A matrix of the inferential results on the BLP generic targets.

coefficients
- An object of class coeftest, contains the coefficients of the BLP regression.

lm
- An object of class lm used to fit the linear regression model.

References


See Also

setup_X1, setup_diff, setup_vcov, propensity_score, proxy_BCA, proxy_CATE

Examples

```r
## generate data
set.seed(1)
n <- 150 # number of observations
p <- 5  # number of covariates
D <- rbinom(n, 1, 0.5) # random treatment assignment
Y <- runif(n)        # outcome variable
propensity_scores <- rep(0.5, n) # propensity scores
```
proxy_BCA <- runif(n)  # proxy BCA estimates
proxy_CATE <- runif(n)  # proxy CATE estimates

## perform BLP
BLP(Y, D, propensity_scores, proxy_BCA, proxy_CATE)

---

**CLAN** *Performs CLAN*

---

**Description**

Performs Classification Analysis (CLAN) on all variables in a design matrix.

**Usage**

```r
CLAN(
  Z_CLAN,
  membership,
  equal_variances = FALSE,
  diff = setup_diff(),
  significance_level = 0.05
)
```

**Arguments**

- **Z_CLAN** A numeric matrix holding variables on which classification analysis (CLAN) shall be performed. CLAN will be performed on each column of the matrix.
- **membership** A logical matrix that indicates the group membership of each observation in Z_CLAN. Needs to be of type `quantile_group`. Typically, the grouping is based on CATE estimates, which are for instance returned by `proxy_CATE`.
- **equal_variances** If TRUE, then all within-group variances of the CLAN groups are assumed to be equal. Default is FALSE. This specification is required for heteroskedasticity-robust variance estimation on the difference of two CLAN generic targets (i.e. variance of the difference of two means). If TRUE (corresponds to homoskedasticity assumption), the pooled variance is used. If FALSE (heteroskedasticity), the variance of Welch's t-test is used.
- **diff** Specifies the generic targets of CLAN. Must be an instance of `setup_diff`. See the documentation of `setup_diff` for details.
- **significance_level** Significance level. Default is 0.05.
**Value**

An object of the class CLAN, consisting of the following components:

- **generic_targets**: A list of result matrices for each variable in Z_CLAN. Each matrix contains inferential results on the CLAN generic targets.
- **coefficients**: A matrix of point estimates of each CLAN generic target parameter.

**References**


**See Also**

quantile_group, setup_diff

**Examples**

```r
## generate data
set.seed(1)
n <- 150 # number of observations
p <- 5 # number of covariates
Z_CLAN <- matrix(runif(n*p), n, p) # design matrix to perform CLAN on
membership <- quantile_group(rnorm(n)) # group membership

## perform CLAN
CLAN(Z_CLAN, membership)
```

---

**GATES**

*Performs GATES regression*

**Description**

Performs the linear regression for the Group Average Treatments Effects (GATES) procedure.

**Usage**

```r
GATES(  
  Y,  
  D,  
  propensity_scores,  
  proxy_BCA,  
  proxy_CATE,  
  membership,  
  HT = FALSE,  
  X1_control = setup_X1(),  
)```
vcov_control = setup_vcov(),
diff = setup_diff(),
significance_level = 0.05
)

Arguments

Y  A numeric vector containing the response variable.
D  A binary vector of treatment assignment. Value one denotes assignment to the
treatment group and value zero assignment to the control group.
propensity_scores  A numeric vector of propensity scores. We recommend to use the estimates of a
propensity_score object.
proxy_BCA  A numeric vector of proxy baseline conditional average (BCA) estimates. We
recommend to use the estimates of a proxy_BCA object.
proxy_CATE  A numeric vector of proxy conditional average treatment effect (CATE) esti-
mates. We recommend to use the estimates of a proxy_CATE object.
membership  A logical matrix that indicates the group membership of each observation in
Z_CLAN. Needs to be of type quantile_group. Typically, the grouping is based
on CATE estimates, which are for instance returned by proxy_CATE.
HT  Logical. If TRUE, a Horvitz-Thompson (HT) transformation is applied (BLP2 in
the paper). Default is FALSE.
X1_control  Specifies the design matrix $X_1$ in the regression. Must be an instance of setup_X1.
See the documentation of setup_X1 for details.
vcov_control  Specifies the covariance matrix estimator. Must be an instance of setup_vcov.
See the documentation of setup_vcov for details.
diff  Specifies the generic targets of GATES. Must be an instance of setup_diff.
See the documentation of setup_diff for details.
significance_level  Significance level. Default is 0.05.

Value

An object of class GATES, consisting of the following components:
generic_targets  A matrix of the inferential results on the GATES generic targets.
coefficients  An object of class coeftest, contains the coefficients of the GATES regression.
1m  An object of class 1m used to fit the linear regression model.

References

ing Inference on Heterogenous Treatment Effects in Randomized Experiments.” arXiv preprint
GenericML

See Also
setup_X1, setup_diff, setup_vcov, propensity_score, proxy_BCA, proxy_CATE

Examples

```r
## generate data
set.seed(1)
n <- 150 # number of observations
p <- 5 # number of covariates
D <- rbinom(n, 1, 0.5) # random treatment assignment
Y <- runif(n) # outcome variable
propensity_scores <- rep(0.5, n) # propensity scores
proxy_BCA <- runif(n) # proxy BCA estimates
proxy_CATE <- runif(n) # proxy CATE estimates
membership <- quantile_group(proxy_CATE) # group membership

## perform GATES
GATES(Y, D, propensity_scores, proxy_BCA, proxy_CATE, membership)
```

GenericML

Generic Machine Learning Inference

Description

Performs generic machine learning inference on heterogeneous treatment effects as in Chernozhukov, Demirer, Duflo and Fernández-Val (2020) with user-specified machine learning methods. Intended for randomized experiments.

Usage

```
GenericML(
  Z,
  D,
  Y,
  learners_GenericML,
  learner_propensity_score = "constant",
  num_splits = 100,
  Z_CLAN = NULL,
  HT = FALSE,
  quantile_cutoffs = c(0.25, 0.5, 0.75),
  X1_BLP = setup_X1(),
  X1_GATES = setup_X1(),
  diff_GATES = setup_diff(),
  diff_CLAN = setup_diff(),
  vcov_BLP = setup_vcov(),
  vcov_GATES = setup_vcov(),
  equal_variances_CLAN = FALSE,
)```

prop_aux = 0.5,
significance_level = 0.05,
min_variation = 1e-05,
parallel = TrueIfUnix(),
num_cores = parallel::detectCores(),
seed = NULL,
store_learners = FALSE,
store_splits = TRUE)

Arguments

Z A numeric design matrix that holds the covariates in its columns.
D A binary vector of treatment assignment. Value one denotes assignment to the
treatment group and value zero assignment to the control group.
Y A numeric vector containing the response variable.
learners_GenericML A character vector specifying the machine learners to be used for estimating
the baseline conditional average (BCA) and conditional average treatment ef-
effect (CATE). Either 'lasso', 'random_forest', 'tree', or a custom learner
specified with mlr3 syntax. In the latter case, do not specify in the mlr3 syntax
specification if the learner is a regression learner or classification learner. Ex-
ample: 'mlr3::lrn("ranger",num.trees = 100)' for a random forest learner
with 100 trees. Note that this is a string and the absence of the classif. or
regr. keywords. See https://mlr3learners.mlr-org.com for a list of mlr3
learners.
learner_propensity_score The estimator of the propensity scores. Either a numeric vector (which is then
taken as estimates of the propensity scores) or a string specifying the estimator.
In the latter case, the string must either be equal to 'constant' (estimates the
propensity scores by mean(D)), 'lasso', 'random_forest', 'tree', or mlr3
syntax. Note that in case of mlr3 syntax, do not specify if the learner is a regres-
sion learner or classification learner. Example: 'mlr3::lrn("ranger",num.trees
= 100)' for a random forest learner with 100 trees. Note that this is a string and
the absence of the classif. or regr. keywords. See https://mlr3learners.
mlr-org.com for a list of mlr3 learners.
num_splits Number of sample splits. Default is 100. Must be larger than one. If you want
to run GenericML on a single split, please use GenericML_single.
Z_CLAN A numeric matrix holding variables on which classification analysis (CLAN)
shall be performed. CLAN will be performed on each column of the matrix. If
NULL (default), then Z_CLAN = Z, i.e. CLAN is performed for all variables in Z.
HT Logical. If TRUE, a Horvitz-Thompson (HT) transformation is applied in the
BLP and GATES regressions. Default is FALSE.
quantile_cutoffs The cutoff points of the quantiles that shall be used for GATES grouping. De-
default is c(0.25, 0.5, 0.75), which corresponds to the four quartiles.
X1_BLP: Specifies the design matrix $X_1$ in the BLP regression. Must be an instance of \texttt{setup_X1}. See the documentation of \texttt{setup_X1} for details.

X1_GATES: Same as X1_BLP, just for the GATES regression.

diff_GATES: Specifies the generic targets of GATES. Must be an instance of \texttt{setup_diff}. See the documentation of \texttt{setup_diff} for details.

diff_CLAN: Same as diff_GATES, just for the CLAN generic targets.

vcov_BLP: Specifies the covariance matrix estimator in the BLP regression. Must be an instance of \texttt{setup_vcov}. See the documentation of \texttt{setup_vcov} for details.

vcov_GATES: Same as vcov_BLP, just for the GATES regression.

equal_variances_CLAN: Logical. If TRUE, then all within-group variances of the CLAN groups are assumed to be equal. Default is FALSE. This specification is required for heteroskedasticity-robust variance estimation on the difference of two CLAN generic targets (i.e. variance of the difference of two means). If TRUE (corresponds to homoskedasticity assumption), the pooled variance is used. If FALSE (heteroskedasticity), the variance of Welch’s t-test is used.

prop_aux: Proportion of samples that shall be in the auxiliary set. Default is 0.5. The number of samples in the auxiliary set will be equal to \lfloor prop_aux \times \text{length}(Y) \rfloor. If the data set is large, you can save computing time by choosing prop_aux to be smaller than 0.5.

significance_level: Significance level for VEIN. Default is 0.05.

min_variation: Specifies a threshold for the minimum variation of the BCA/CATE predictions. If the variation of a BCA/CATE prediction falls below this threshold, random noise with distribution $N(0, \text{var}(Y)/20)$ is added to it. Default is $1e^{-05}$.

parallel: Logical. If TRUE, parallel computing will be used. Currently only supported for Unix systems.

num_cores: Number of cores to be used in parallelization (if applicable). Default is the number of cores of the user’s machine.

seed: Random seed. Default is NULL for no random seeding.

store_learners: Logical. If TRUE, all intermediate results of the learners will be stored. That is, for each learner and each split, all BCA and CATE predictions as well as all BLP, GATES, CLAN, and $\Lambda$ estimates will be stored. Default is FALSE.

store_splits: Logical. If TRUE (default), the sample splits will be stored.

Details

The specifications lasso, random_forest, and tree in learners_GenericML and learner_propensity_score correspond to the following mlr3 specifications (we omit the keywords \texttt{classif.} and \texttt{regr.}). lasso is a cross-validated Lasso estimator, which corresponds to \texttt{mlr3::lrn("cv_glmnet",s = "lambda.min",alpha = 1')}.
random_forest is a random forest with 500 trees, which corresponds to \texttt{mlr3::lrn("ranger",num.trees = 500')}.
tree is a tree learner, which corresponds to \texttt{mlr3::lrn("rpart")}. 
Value

An object of class `GenericML`. On this object, we recommend to use the accessor functions `get_BLP`, `get_GATES`, and `get_CLAN` to extract the results of the analyses of BLP, GATES, and CLAN, respectively. An object of class `GenericML` contains the following components:

- **VEIN** A list containing two sub-lists called `best_learners` and `all_learners`, respectively. Each of these two sub-lists contains the inferential VEIN results on the generic targets of the BLP, GATES, and CLAN analyses. `all_learners` does this for all learners specified in the argument `learners_GenericML`, `best_learners` only for the corresponding best learners. Which learner is best for which analysis is assessed by the $\Lambda$ criteria discussed in Sections 5.2 and 5.3 of the paper.

- **best** A list containing information on the evaluation of which learner is the best for which analysis. Contains four components. The first three contain the name of the best learner for BLP, GATES, and CLAN, respectively. The fourth component, `overview`, contains the two $\Lambda$ criteria used to determine the best learners (discussed in Sections 5.2 and 5.3 of the paper).

- **propensity_scores** The propensity score estimates as well as the mlr3 objects used to estimate them (if mlr3 was used for estimation).

- **GenericML_single** Only nonempty if `store_learners = TRUE`. Contains all intermediate results of each learner for each split. That is, for a given learner (first level of the list) and split (second level), objects of classes `BLP`, `GATES`, `CLAN`, `proxy_BCA`, `proxy_CATE` as well as the $\Lambda$ criteria ("best") are listed, which were computed with the given learner and split.

- **splits** Only nonempty if `store_splits = TRUE`. Contains a character matrix of dimension `length(Y)` by `num_splits`. Contains the group membership (main or auxiliary) of each observation (rows) in each split (columns). "M" denotes the main set, "A" the auxiliary set.

- **arguments** A list of arguments used in the function call.

Note

In an earlier development version, Lucas Kitzmueller alerted us to several minor bugs and proposed fixes. Many thanks to him!

References


See Also

`plot`, `print`, `get_BLP`, `get_GATES`, `get_CLAN`, `setup_X1`, `setup_diff`, `setup_vcov`, `GenericML_single`
Examples

```r
if (require("glmnet") && require("ranger")) {

## generate data
set.seed(1)
n <- 150 \text{ # number of observations}
p <- 5 \text{ # number of covariates}
D <- rbinom(n, 1, 0.5) \text{ # random treatment assignment}
Z <- matrix(runif(n*p), n, p) \text{ # design matrix}
Y0 <- as.numeric(Z %*% rexp(p) + rnorm(n)) \text{ # potential outcome without treatment}
Y1 <- 2 + Y0 \text{ # potential outcome under treatment}
Y <- ifelse(D == 1, Y1, Y0) \text{ # observed outcome}

## column names of Z
colnames(Z) <- paste0("V", 1:p)

## specify learners
learners <- c("lasso", "mlr3::lrn(ranger', num.trees = 10")

## specify quantile cutoffs (the 4 quartile groups here)
quantile_cutoffs <- c(0.25, 0.5, 0.75)

## specify the differenced generic targets of GATES and CLAN
# use G4-G1, G4-G2, G4-G3 as differenced generic targets in GATES
diff_GATES <- setup_diff(subtract_from = "most",
                    subtracted = c(1,2,3))
# use G1-G3, G1-G2 as differenced generic targets in CLAN
diff_CLAN <- setup_diff(subtract_from = "least",
                    subtracted = c(3,2))

## perform generic ML inference
# small number of splits to keep computation time low
x <- GenericML(Z, D, Y, learners, num_splits = 2,
            quantile_cutoffs = quantile_cutoffs,
            diff_GATES = diff_GATES,
            diff_CLAN = diff_CLAN,
            parallel = FALSE)

## access BLP generic targets for best learner and make plot
get_BLP(x, plot = TRUE)

## access GATES generic targets for best learner and make plot
get_GATES(x, plot = TRUE)

## access CLAN generic targets for "V1" & best learner and make plot
get_CLAN(x, variable = "V1", plot = TRUE)
}
```
GenericML_single  

Single iteration of the GenericML algorithm

Description

Performs generic ML inference for a single learning technique and a given split of the data. Can be seen as a single iteration of Algorithm 1 in the paper.

Usage

GenericML_single(
  Z,
  D,
  Y,
  learner,
  propensity_scores,
  M_set,
  A_set = setdiff(1:length(Y), M_set),
  Z_CLAN = NULL,
  HT = FALSE,
  quantile_cutoffs = c(0.25, 0.5, 0.75),
  X1_BLP = setup_X1(),
  X1_GATES = setup_X1(),
  diff_GATES = setup_diff(),
  diff_CLAN = setup_diff(),
  vcov_BLP = setup_vcov(),
  vcov_GATES = setup_vcov(),
  equal_variances_CLAN = FALSE,
  significance_level = 0.05,
  min_variation = 1e-05
)

Arguments

Z  A numeric design matrix that holds the covariates in its columns.
D  A binary vector of treatment assignment. Value one denotes assignment to the treatment group and value zero assignment to the control group.
Y  A numeric vector containing the response variable.
learner A character specifying the machine learner to be used for estimating the baseline conditional average (BCA) and conditional average treatment effect (CATE). Either 'lasso', 'random_forest', 'tree', or a custom learner specified with mlr3 syntax. In the latter case, do not specify in the mlr3 syntax specification if the learner is a regression learner or classification learner. Example: 'mlr3::lrn("ranger", num.trees = 100)' for a random forest learner with 100 trees. Note that this is a string and the absence of the classif. or regr. keywords. See https://mlr3learners.mlr-org.com for a list of mlr3 learners.
GenericML_single

propensity_scores
A numeric vector of propensity score estimates.

M_set
A numerical vector of indices of observations in the main sample.

A_set
A numerical vector of indices of observations in the auxiliary sample. Default is complementary set to M_set.

Z_CLAN
A numeric matrix holding variables on which classification analysis (CLAN) shall be performed. CLAN will be performed on each column of the matrix. If NULL (default), then Z_CLAN = Z, i.e. CLAN is performed for all variables in Z.

HT
Logical. If TRUE, a Horvitz-Thompson (HT) transformation is applied in the BLP and GATES regressions. Default is FALSE.

quantile_cutoffs
The cutoff points of the quantiles that shall be used for GATES grouping. Default is c(0.25, 0.5, 0.75), which corresponds to the four quartiles.

X1_BLP
Specifies the design matrix \( X_1 \) in the BLP regression. Must be an instance of setup_X1. See the documentation of setup_X1 for details.

X1_GATES
Same as X1_BLP, just for the GATES regression.

diff_GATES
Specifies the generic targets of GATES. Must be an instance of setup_diff. See the documentation of setup_diff for details.

diff_CLAN
Same as diff_GATES, just for the CLAN generic targets.

vcov_BLP
Specifies the covariance matrix estimator in the BLP regression. Must be an instance of setup_vcov. See the documentation of setup_vcov for details.

vcov_GATES
Same as vcov_BLP, just for the GATES regression.

equal_variances_CLAN
Logical. If TRUE, then all within-group variances of the CLAN groups are assumed to be equal. Default is FALSE. This specification is required for heteroskedasticity-robust variance estimation on the difference of two CLAN generic targets (i.e. variance of the difference of two means). If TRUE (corresponds to homoskedasticity assumption), the pooled variance is used. If FALSE (heteroskedasticity), the variance of Welch’s t-test is used.

significance_level
Significance level for VEIN. Default is 0.05.

min_variation
Specifies a threshold for the minimum variation of the BCA/CATE predictions. If the variation of a BCA/CATE prediction falls below this threshold, random noise with distribution \( N(0, \text{var}(Y)/20) \) is added to it. Default is 1e-05.

Details
The specifications lasso, random_forest, and tree in learner correspond to the following mlr3 specifications (we omit the keywords classif. and regr.). lasso is a cross-validated Lasso estimator, which corresponds to `mlr3::lrn("cv_glmnet", s = "lambda.min", alpha = 1)`.
random_forest is a random forest with 500 trees, which corresponds to `mlr3::lrn("ranger", num.trees = 500)`.
tree is a tree learner, which corresponds to `mlr3::lrn("rpart")`. 
Value

A list with the following components:

- **BLP** An instance of **BLP**.
- **GATES** An instance of **GATES**.
- **CLAN** An instance of **CLAN**.
- **proxy_BCA** An instance of **proxy_BCA**.
- **proxy_CATE** An instance of **proxy_CATE**.
- **best** Estimates of the $\Lambda$ parameters for finding the best learner, returned by **lambda_parameters**.

References


See Also

**GenericML**

Examples

```r
if(require("ranger")){
  ## generate data
  set.seed(1)
  n <- 150 # number of observations
  p <- 5 # number of covariates
  Z <- matrix(runif(n*p), n, p) # design matrix
  D <- rbinom(n, 1, 0.5) # random treatment assignment
  Y <- runif(n) # outcome variable
  propensity_scores <- rep(0.5, n) # propensity scores
  M_set <- sample(1:n, size = n/2) # main set

  ## specify learner
  learner <- "mlr3::lrn('ranger', num.trees = 10)"

  ## run single GenericML iteration
  GenericML_single(Z, D, Y, learner, propensity_scores, M_set)
}
```
Accessor function for the BLP generic target estimates

Usage

get_BLP(x, learner = "best", plot = TRUE)

Arguments

x
An object of the class GenericML.

learner
A character string of the learner whose BLP generic target estimates shall be accessed. Default is "best" for the best learner for BLP.

plot
Logical. If TRUE (default), a plot is printed.

Value

A numeric matrix of BLP generic target estimates which contains information on point estimates, confidence bounds, and (adjusted) p-values. Furthermore, prints a plot if plot = TRUE.

See Also

GenericML, get_GATES, get_CLAN, plot

Examples

if(require("glmnet") && require("ranger")){
  ## generate data
  set.seed(1)
  n <- 150 # number of observations
  p <- 5  # number of covariates
  D <- rbinom(n, 1, 0.5) # random treatment assignment
  Z <- matrix(runif(n*p), n, p) # design matrix
  Y0 <- as.numeric(Z %*% rexp(p) + rnorm(n)) # potential outcome without treatment
  Y1 <- 2 + Y0 # potential outcome under treatment
  Y <- ifelse(D == 1, Y1, Y0) # observed outcome

  ## column names of Z
  colnames(Z) <- paste0("V", 1:p)

  ## specify learners
  learners <- c("tree", "mlr3::lrn('ranger', num.trees = 10)"

  ## perform generic ML inference
  # small number of splits to keep computation time low
  x <- GenericML(Z, D, Y, learners, num_splits = 2,
get_CLAN

Accessor function for the CLAN generic target estimates

Description

Accessor function for the CLAN generic target estimates

Usage

get_CLAN(x, variable, learner = "best", plot = TRUE)

Arguments

x
variable
learner
plot

An object of the class GenericML.
The (character) name of a variable on which CLAN was performed.
A character string of the learner whose CLAN generic target estimates shall be accessed. Default is "best" for the best learner for CLAN
Logical. If TRUE (default), a plot is printed.

Value

A numeric matrix of CLAN generic target estimates which contains information on point estimates, confidence bounds, and (adjusted) p-values. Furthermore, prints a plot if plot = TRUE.
get\_GATES

Accessor function for the GATES generic target estimates

See Also

GenericML, get\_BLP, get\_GATES, plot

Examples

if(require("glmnet") && require("ranger")){
  ## generate data
  set.seed(1)
  n <- 200 # number of observations
  p <- 5 # number of covariates
  D <- rbinom(n, 1, 0.5) # random treatment assignment
  Z <- matrix(runif(n*p), n, p) # design matrix
  Y0 <- as.numeric(Z %*% rexp(p) + rnorm(n)) # potential outcome without treatment
  Y1 <- 2 + Y0 # potential outcome under treatment
  Y <- ifelse(D == 1, Y1, Y0) # observed outcome

  ## column names of Z
  colnames(Z) <- paste0("V", 1:p)

  ## specify learners
  learners <- c("lasso", "mlr3::lrn('ranger', num.trees = 30)")

  ## perform generic ML inference
  # small number of splits to keep computation time low
  x <- GenericML(Z, D, Y, learners, num_splits = 10,
                  parallel = FALSE)

  ## access BLP generic targets for best learner w/o plot
  get\_BLP(x, learner = "best", plot = FALSE)

  ## access BLP generic targets for ranger learner w/o plot
  get\_BLP(x, learner = "mlr3::lrn('ranger', num.trees = 30)", plot = FALSE)

  ## access GATES generic targets for best learner w/o plot
  get\_GATES(x, learner = "best", plot = FALSE)

  ## access GATES generic targets for ranger learner w/o plot
  get\_GATES(x, learner = "mlr3::lrn('ranger', num.trees = 30)", plot = FALSE)

  ## access CLAN generic targets for "V1" & best learner, w/o plot
  get\_CLAN(x, learner = "best", variable = "V1", plot = FALSE)

  ## access CLAN generic targets for "V1" & ranger learner, w/o plot
  get\_CLAN(x, learner = "mlr3::lrn('ranger', num.trees = 30)",
            variable = "V1", plot = FALSE)
}
Description

Accessor function for the GATES generic target estimates

Usage

get_GATES(x, learner = "best", plot = TRUE)

Arguments

  x  
  An object of the class GenericML.

  learner  
  A character string of the learner whose GATES generic target estimates shall be 
  accessed. Default is "best" for the best learner for GATES.

  plot  
  Logical. If TRUE (default), a plot is printed.

Value

A numeric matrix of GATES generic target estimates which contains information on point estimates, 
confidence bounds, and (adjusted) p-values. Furthermore, prints a plot if plot = TRUE.

See Also

GenericML, get_BLP, get_CLAN, plot

Examples

if(require("glmnet") && require("ranger")){
  ## generate data
  set.seed(1)
  n <- 200 # number of observations
  p <- 5  # number of covariates
  D <- rbinom(n, 1, 0.5) # random treatment assignment
  Z <- matrix(runif(n*p), n, p) # design matrix
  Y0 <- as.numeric(Z %*% rexp(p) + rnorm(n)) # potential outcome without treatment
  Y1 <- 2 + Y0 # potential outcome under treatment
  Y <- ifelse(D == 1, Y1, Y0) # observed outcome

  ## column names of Z
  colnames(Z) <- paste0("V", 1:p)

  ## specify learners
  learners <- c("lasso", "mlr3::lrn(' ranger', num.trees = 30")

  ## perform generic ML inference
  # small number of splits to keep computation time low
  x <- GenericML(Z, D, Y, learners, num_splits = 10,
                  parallel = FALSE)

  ## access BLP generic targets for best learner w/o plot
  get_BLP(x, learner = "best", plot = FALSE)

  ## access BLP generic targets for ranger learner w/o plot
get_BLP(x, learner = "mlr3::lrn('ranger', num.trees = 30)", plot = FALSE)

## access GATES generic targets for best learner w/o plot
get_GATES(x, learner = "best", plot = FALSE)

## access GATES generic targets for ranger learner w/o plot
get_GATES(x, learner = "mlr3::lrn('ranger', num.trees = 30)", plot = FALSE)

## access CLAN generic targets for "V1" & best learner, w/o plot
get_CLAN(x, learner = "best", variable = "V1", plot = FALSE)

## access CLAN generic targets for "V1" & ranger learner, w/o plot
get_CLAN(x, learner = "mlr3::lrn('ranger', num.trees = 30)",
          variable = "V1", plot = FALSE)
}

lambda_parameters

Estimate the two lambda parameters

Description

Estimates the lambda parameters $\Lambda$ and $\overline{\Lambda}$ whose medians are used to find the best ML method.

Usage

lambda_parameters(BLP, GATES, proxy_CATE, membership)

Arguments

- BLP: An instance of BLP.
- GATES: An instance of GATES.
- proxy_CATE: Proxy estimates of the CATE.
- membership: A logical matrix that indicates the group membership of each observation in Z_CLAN. Needs to be of type quantile_group. Typically, the grouping is based on CATE estimates, which are for instance returned by proxy_CATE.

Value

A list containing the estimates of $\Lambda$ and $\overline{\Lambda}$, denoted lambda and lambda.bar, respectively.

References

Examples

```r
## generate data
set.seed(1)
n <- 200  # number of observations
p <- 5   # number of covariates
D <- rbinom(n, 1, 0.5)  # random treatment assignment
Y <- runif(n)  # outcome variable
propensity_scores <- rep(0.5, n)  # propensity scores
proxy_BCA <- runif(n)  # proxy BCA estimates
proxy_CATE <- runif(n)  # proxy CATE estimates
membership <- quantile_group(proxy_CATE)  # group membership

## perform BLP
BLP <- BLP(Y, D, propensity_scores, proxy_BCA, proxy_CATE)

## perform GATES
GATES <- GATES(Y, D, propensity_scores, proxy_BCA, proxy_CATE, membership)

## get estimates of the lambda parameters
lambda_parameters(BLP, GATES, proxy_CATE, membership)
```

---

**Med**

*Calculate lower and upper median*

**Description**

Calculates the lower and median of a vector as proposed in Comment 4.2 in the paper.

**Usage**

```r
Med(x)
```

**Arguments**

- `x`: A numeric vector.

**Value**

A list with the upper and lower median and the Med statistic (which is their mean).

**References**


Examples

set.seed(1)
x <- runif(100)
Med(x)

Description

Visualizes the estimates of the generic targets of interest: plots the point estimates as well as the corresponding confidence bounds. The generic targets of interest can be (subsets of) the parameters of the BLP, GATES, or CLAN analysis.

Usage

```r
## S3 method for class 'GenericML'
plot(
x, 
type = "GATES",
learner = "best",
CLAN_variable = NULL,
groups = "all",
limits = NULL,
title = NULL,
...
)
```

Arguments

- **x**: An instance of `GenericML`.
- **type**: The analysis whose parameters shall be plotted. Either "GATES", "BLP", or "CLAN". Default is "GATES".
- **learner**: The learner whose results are to be returned. Default is "best" for the best learner as measured by the Lambda parameters.
- **CLAN_variable**: Name of the CLAN variable to be plotted. Only applicable if type = "CLAN".
- **groups**: Character vector indicating the per-group parameter estimates that shall be plotted in GATES and CLAN analyses. Default is "all" for all parameters. If there are K groups, this variable is a subset of c("G1","G2",...,"GK","G1-G2","G1-G2",...,"G1-GK","GK-G1","GK-G2",...), where Gk denotes the k-th group. Note that this set depends on the choices of the arguments "diff_GATES" and "diff_CLAN" of the `GenericML` object.
- **limits**: The limits of the y-axis of the plot.
- **title**: The title of the plot.
- **...**: Additional arguments to be passed down.
Value

An object of class "ggplot" (see ggplot).

See Also

GenericML, get_BLP, get_GATES, get_CLAN, setup_diff

Examples

if(require("glmnet")) {

## generate data
set.seed(1)
n <- 150 # number of observations
p <- 5  # number of covariates
D <- rbinom(n, 1, 0.5) # random treatment assignment
Z <- matrix(runif(n*p), n, p) # design matrix
Y0 <- as.numeric(Z %*% rexp(p) + rnorm(n)) # potential outcome without treatment
Y1 <- 2 + Y0 # potential outcome under treatment
Y <- ifelse(D == 1, Y1, Y0) # observed outcome

## name the columns of Z
colnames(Z) <- paste0("V", 1:p)

## specify learners
learners <- c("lasso")

## specify quantile cutoffs (the 4 quartile groups here)
quantile_cutoffs <- c(0.25, 0.5, 0.75)

diff_GATES <- setup_diff(subtract_from = "most", subtracted = c(1,2,3))
diff_CLAN <- setup_diff(subtract_from = "least", subtracted = c(3,2))

## perform generic ML inference
# small number of splits to keep computation time low
x <- GenericML(Z, D, Y, learners, num_splits = 2,
                 quantile_cutoffs = quantile_cutoffs,
                 diff_GATES = diff_GATES,
                 diff_CLAN = diff_CLAN,
                 parallel = FALSE)

## plot BLP parameters
plot(x, type = "BLP")

## plot GATES parameters "G1", "G4", "G4-G1"
plot(x, type = "GATES", groups = c("G1", "G4", "G4-G1"))

## plot CLAN parameters "G1", "G2", "G2-G1" of variable "V1":
plot(x, type = "CLAN", CLAN_variable = "V1",}

groups = c("G1", "G2", "G1-G3")
}

print.G genericML

Description
Prints key results of the analyses conducted in GenericML.

Usage
## S3 method for class 'GenericML'
print(x, ...)

Arguments
x
An instance of GenericML.
...
Additional arguments to be passed down.

Value
A print to the console.

Examples
if(require("glmnet")){
  
  ## generate data
  set.seed(1)
  n <- 150 # number of observations
  p <- 5 # number of covariates
  D <- rbinom(n, 1, 0.5) # random treatment assignment
  Z <- matrix(runif(n*p), n, p) # design matrix
  Y0 <- as.numeric(Z %*% rexp(p) + rnorm(n)) # potential outcome without treatment
  Y1 <- 2 + Y0 # potential outcome under treatment
  Y <- ifelse(D == 1, Y1, Y0) # observed outcome

  ## specify learners
  learners <- c("lasso")

  ## perform generic ML inference
  # small number of splits to keep computation time low
  x <- GenericML(Z, D, Y, learners, num_splits = 2,
                 parallel = FALSE)

  ## print
  print(x)
}
propensity_score

Propensity score estimation

Description

Estimates the propensity scores \( Pr[D = 1|Z] \) for binary treatment assignment \( D \) and covariates \( Z \). Either done by taking the empirical mean of \( D \) (which should equal roughly 0.5, since we assume a randomized experiment), or by direct machine learning estimation.

Usage

```r
propensity_score(Z, D, estimator = "constant")
```

Arguments

- **Z** A numeric design matrix that holds the covariates in its columns.
- **D** A binary vector of treatment assignment. Value one denotes assignment to the treatment group and value zero assignment to the control group.
- **estimator** Character specifying the estimator. Must either be equal to 'constant' (estimates the propensity scores by mean(D)), 'lasso', 'random_forest', 'tree', or mlr3 syntax. Note that in case of mlr3 syntax, do not specify if the learner is a regression learner or classification learner. Example: 'mlr3::lrn("ranger", num.trees = 500)' for a random forest learner. Note that this is a string and the absence of the classif. or regr. keywords. See https://mlr3learners.mlr-org.com for a list of mlr3 learners.

Details

The specifications lasso, random_forest, and tree in estimator correspond to the following mlr3 specifications (we omit the keywords classif. and regr.). lasso is a cross-validated Lasso estimator, which corresponds to 'mlr3::lrn("cv_glmnet", s = "lambda.min", alpha = 1)'. random_forest is a random forest with 500 trees, which corresponds to 'mlr3::lrn("ranger",num.trees = 500)'. tree is a tree learner, which corresponds to 'mlr3::lrn("rpart")'.

Value

An object of class propensity_score, consisting of the following components:

- **estimates** A numeric vector of propensity score estimates.
- **mlr3_objects** mlr3 objects used for estimation. Only non-empty if mlr3 was used.

References


Examples

```r
## generate data
set.seed(1)
n <- 100 # number of observations
p <- 5 # number of covariates
D <- rbinom(n, 1, 0.5) # random treatment assignment
Z <- matrix(runif(n*p), n, p) # design matrix

## estimate propensity scores via mean(D)...
propensity_score(Z, D, estimator = "constant")

## ... and via SVM with cache size 40
if(require("e1071")){
  propensity_score(Z, D,
  estimator = 'mlr3::lrn("svm", cachesize = 40')
}
```

---

**proxy_BCA**

*Baseline Conditional Average*

**Description**

Proxy estimation of the Baseline Conditional Average (BCA), defined by $E[Y|D=0, Z]$. Estimation is done on the auxiliary sample, but BCA predictions are made for all observations.

**Usage**

```r
proxy_BCA(Z, D, Y, A_set, learner, min_variation = 1e-05)
```

**Arguments**

- **Z**: A numeric design matrix that holds the covariates in its columns.
- **D**: A binary vector of treatment assignment. Value one denotes assignment to the treatment group and value zero assignment to the control group.
- **Y**: A numeric vector containing the response variable.
- **A_set**: A numerical vector of the indices of the observations in the auxiliary sample.
- **learner**: A string specifying the machine learner for the estimation. Either `'lasso'`, `'random_forest'`, `'tree'`, or a custom learner specified with mlr3 syntax. In the latter case, do not specify in the mlr3 syntax specification if the learner is a regression learner or classification learner. Example: `'mlr3::lrn("ranger", num.trees = 100)’ for a random forest learner with 100 trees. Note that this is a string and the absence of the classif. or regr. keywords. See [https://mlr3learners.mlr-org.com](https://mlr3learners.mlr-org.com) for a list of mlr3 learners.
- **min_variation**: Specifies a threshold for the minimum variation of the predictions. If the variation of a BCA prediction falls below this threshold, random noise with distribution $N(0, var(Y)/20)$ is added to it. Default is $1e^{-05}$. 
proxy_BCA

Details

The specifications lasso, random_forest, and tree in learner correspond to the following mlr3 specifications (we omit the keywords classif. and regr.). lasso is a cross-validated Lasso estimator, which corresponds to `mlr3::lrn("cv_glmnet", s = "lambda.min", alpha = 1)`. random_forest is a random forest with 500 trees, which corresponds to `mlr3::lrn("ranger", num.trees = 500)`. tree is a tree learner, which corresponds to `mlr3::lrn("rpart")`.

Value

An object of class proxy_BCA, consisting of the following components:

estimates A numeric vector of BCA estimates of each observation.
mlr3_objects mlr3 objects used for estimation.

References


See Also

proxy_CATE

Examples

```r
if(require("ranger")){
  ## generate data
  set.seed(1)
  n <- 150 # number of observations
  p <- 5 # number of covariates
  D <- rbinom(n, 1, 0.5) # random treatment assignment
  Z <- matrix(runif(n*p), n, p) # design matrix
  Y0 <- as.numeric(Z %*% rexp(p) + rnorm(n)) # potential outcome without treatment
  Y1 <- 2 + Y0 # potential outcome under treatment
  Y <- ifelse(D == 1, Y1, Y0) # observed outcome
  A_set <- sample(1:n, size = n/2) # auxiliary set

  ## BCA predictions via random forest
  proxy_BCA(Z, D, Y, A_set, learner = "mlr3::lrn('ranger', num.trees = 10)")
}
```
Description
Proxy estimation of the Conditional Average Treatment Effect (CATE), defined by $E[Y|D = 1, Z] - E[Y|D = 0, Z]$. Estimation is done on the auxiliary sample, but CATE predictions are made for all observations.

Usage
```
proxy_CATE(Z, D, Y, A_set, learner, proxy_BCA = NULL, min_variation = 1e-05)
```

Arguments
- **Z**: A numeric design matrix that holds the covariates in its columns.
- **D**: A binary vector of treatment assignment. Value one denotes assignment to the treatment group and value zero assignment to the control group.
- **Y**: A numeric vector containing the response variable.
- **A_set**: A numerical vector of the indices of the observations in the auxiliary sample.
- **learner**: A string specifying the machine learner for the estimation. Either 'lasso', 'random_forest', 'tree', or a custom learner specified with mlr3 syntax. In the latter case, do not specify in the mlr3 syntax specification if the learner is a regression learner or classification learner. Example: 'mlr3::lrn("ranger",num.trees = 100)' for a random forest learner with 100 trees. Note that this is a string and the absence of the classif. or regr. keywords. See https://mlr3learners.mlr-org.com for a list of mlr3 learners.
- **proxy_BCA**: A vector of proxy estimates of the baseline conditional average, BCA, $E[Y|D = 0, Z]$. If NULL, these will be estimated separately.
- **min_variation**: Minimum variation of the predictions before random noise with distribution $N(0, var(Y)/20)$ is added. Default is $1e-05$.

Details
The specifications lasso, random_forest, and tree in learner correspond to the following mlr3 specifications (we omit the keywords classif. and regr.). lasso is a cross-validated Lasso estimator, which corresponds to 'mlr3::lrn("cv_glmnet",s = "lambda.min",alpha = 1)'. random_forest is a random forest with 500 trees, which corresponds to 'mlr3::lrn("ranger",num.trees = 500)'. tree is a tree learner, which corresponds to 'mlr3::lrn("rpart")'.

Value
An object of class proxy_CATE, consisting of the following components:
- **estimates**: A numeric vector of CATE estimates of each observation.
- **mlr3_objects**: mlr3 objects used for estimation of $E[Y|D = 1, Z]$ (Y1_learner) and $E[Y|D = 0, Z]$ (Y0_learner). The latter is not available if proxy_BCA = NULL.
References


See Also

proxy_BCA

Examples

```r
if(require("ranger")){
  ## generate data
  set.seed(1)
  n <- 150  # number of observations
  p <- 5   # number of covariates
  D <- rbinom(n, 1, 0.5)  # random treatment assignment
  Z <- matrix(runif(n*p), n, p)  # design matrix
  Y0 <- as.numeric(Z %*% rexp(p) + rnorm(n))  # potential outcome without treatment
  Y1 <- 2 + Y0  # potential outcome under treatment
  Y <- ifelse(D == 1, Y1, Y0)  # observed outcome
  A_set <- sample(1:n, size = n/2)  # auxiliary set

  ## CATE predictions via random forest
  proxy_CATE(Z, D, Y, A_set, learner = "mlr3::lrn('ranger', num.trees = 10)")
}
```

quantile_group

**Partition a vector into quantile groups**

**Description**

Partitions a vector into quantile groups and returns a logical matrix indicating group membership.

**Usage**

```r
quantile_group(x, cutoffs = c(0.25, 0.5, 0.75))
```

**Arguments**

- `x`: A numeric vector to be partitioned.
- `cutoffs`: A numeric vector denoting the quantile cutoffs for the partition. Default are the quartiles: c(0.25, 0.5, 0.75).
Value

An object of type `quantile_group`, which is a logical matrix indicating group membership.

Examples

```r
set.seed(1)
x <- runif(100)
cutoffs <- c(0.25, 0.5, 0.75)
quantile_group(x, cutoffs)
```

Description

This setup function controls how differences of generic target parameters are taken. Returns a list with two components, called `subtract_from` and `subtracted`. The first element (`subtract_from`) denotes what shall be the base group to subtract from in the generic targets of interest (GATES or CLAN); either "most" or "least". The second element (`subtracted`) are the groups to be subtracted from `subtract_from`, which is a subset of 1, 2, ..., K, where K equals the number of groups. The number of groups should be consistent with the number of groups induced by the argument `quantile_cutoffs`, which is the cardinality of `quantile_cutoffs`, plus one.

Usage

```r
setup_diff(subtract_from = "most", subtracted = 1)
```

Arguments

- `subtract_from` String indicating the base group to subtract from, either "most" (default) or "least". The most affected group corresponds to the K-th group in the paper (there are K groups). The least affected group corresponds to the first group.
- `subtracted` Vector indicating the groups to be subtracted from the group specified in `subtract_from`. If there are K groups, `subtracted` should be a subset of 1, 2, ..., K. Be careful to not specify a zero difference: If `subtract_from = "most"`, subtracting group K results in a zero difference. Same if `subtract_from = "least"` and we subtract group 1.

Details

The output of this setup function is intended to be used as argument in the functions `GenericML` and `GenericML_single` (arguments `diff_GATES`, `diff_CLAN`), as well as `GATES` and `CLAN` (argument `diff`).
**Value**

An object of class `setup_diff`, consisting of the following components:

- `subtract_from` A character equal to "most" or "least".
- `subtracted` A numeric vector of group indices.

See the description above for details.

**References**


**See Also**

`GenericML`, `GenericML_single`, `CLAN`, `GATES`, `setup_X1`, `setup_vcov`

**Examples**

```r
## specify quantile cutoffs (the 4 quartile groups here)
quantile_cutoffs <- c(0.25, 0.5, 0.75)

## Use group difference GK-G1 as generic targets in GATES and CLAN
## Gx is the x-th group
setup_diff(subtract_from = "most", subtracted = 1)

## Use GK-G1, GK-G2, GK-G3 as differenced generic targets
setup_diff(subtract_from = "most", subtracted = c(1,2,3))

## Use G1-G2, G1-G3 as differenced generic targets
setup_diff(subtract_from = "least", subtracted = c(3,2))
```

**setup_vcov**

*Setup function for vcov_control arguments*

**Description**

Returns a list with two elements called `estimator` and `arguments`. The element `estimator` is a string specifying the covariance matrix estimator to be used in the linear regression regression of interest and needs to be a covariance estimator function in the `sandwich` package. The second element, `arguments`, is a list of arguments that shall be passed to the function specified in the first element, `estimator`.

**Usage**

```r
setup_vcov(estimator = "vcovHC", arguments = list(type = "const"))
```
Arguments

estimator  Character specifying a covariance matrix estimator in the sandwich package. Default is "vcovHC". Supported estimators are "vcovBS", "vcovCL", "vcovHAC", and "vcovHC".

arguments  A list of arguments that are to be passed to the function in the sandwich package that is specified in estimator. Default is list(type = "const"), which specifies the homoskedastic ordinary least squares covariance matrix estimator.

Details

The output of this setup function is intended to be used as argument in the functions GenericML and GenericML_single (arguments vcov_BLP, vcov_GATES), as well as BLP and GATES (argument vcov_control).

Value

An object of class setup_vcov, consisting of the following components:

estimator  A character equal to covariance estimation function names in the sandwich package.

arguments  A list of arguments that shall be passed to the function specified in the estimator argument.

See the description above for details.

References


See Also

GenericML, GenericML_single, BLP, GATES, setup_X1, setup_diff

Examples

# use standard homoskedastic OLS covariance matrix estimate
setup_vcov(estimator = "vcovHC", arguments = list(type = "const"))

# use White's heteroskedasticity-robust estimator
setup_vcov(estimator = "vcovHC", arguments = list(type = "H0"))

if (require("sandwich")){
  # use HAC-robust estimator with prewhitening and Andrews' (Econometrica, 1991) weights
  # since weightsAndrews() is a function in 'sandwich', require this package
  setup_vcov(estimator = "vcovHAC", arguments = list(prewhite = TRUE, weights = weightsAndrews))
}
**Description**

Returns a list with three elements. The first element of the list, `funs_Z`, controls which functions of matrix `Z` are used as regressors in \( X_1 \). The second element, `covariates`, is an optional matrix of custom covariates that shall be included in \( X_1 \). The third element, `fixed_effects`, controls the inclusion of fixed effects.

**Usage**

```r
setup_X1(funs_Z = c("B"), covariates = NULL, fixed_effects = NULL)
```

**Arguments**

- **funs_Z**: Character vector controlling the functions of `Z` to be included in \( X_1 \). Subset of `c("S","B","p")`, where "p" corresponds to the propensity scores, "B" to the proxy baseline estimates, and "S" to the proxy CATE estimates. Default is "B".
- **covariates**: Optional numeric matrix containing additional covariates to be included in \( X_1 \). Default is NULL.
- **fixed_effects**: Numeric vector of integers that indicates cluster membership of the observations: For each cluster, a fixed effect will be added. Default is NULL for no fixed effects.

**Details**

The output of this setup function is intended to be used as argument in the functions `GenericML` and `GenericML_single` (arguments `X1_BLP`, `X1_GATES`), as well as `BLP` and `GATES` (argument `X1_control`).

**Value**

An object of class `setup_X1`, consisting of the following components:

- **funs_Z**: A character vector, being a subset of `c("S","B","p")`.
- **covariates**: Either NULL or a numeric matrix.
- **fixed_effects**: Either NULL or an integer vector indicating cluster membership.

See the description above for details.

**References**

See Also

GenericML, GenericML_single, BLP, GATES, setup_vcov, setup_diff

Examples

```r
set.seed(1)
n <- 100 # sample size
p <- 5  # number of covariates
covariates <- matrix(runif(n*p), n, p) # sample matrix of covariates

# let there be three clusters; assign membership randomly
fixed_effects <- sample(c(1,2,3), size = n, replace = TRUE)

# use BCA estimates in matrix X1
setup_X1(funs_Z = "B", covariates = NULL, fixed_effects = NULL)

# use BCA and propensity score estimates in matrix X1
# uses uniform covariates and fixed effects
setup_X1(funs_Z = c("B", "p"), covariates = covariates, fixed_effects = NULL)
```

**TrueIfUnix**

*Check if user’s OS is a Unix system*

**Description**

Check if user’s OS is a Unix system

**Usage**

`TrueIfUnix()`

**Value**

A Boolean that is TRUE if the user’s operating system is a Unix system and FALSE otherwise.
Index

BLP, 2, 10, 14, 19, 31–33
CLAN, 4, 10, 14, 29, 30
coefftest, 3, 6
GATES, 5, 10, 14, 19, 29–33
GenericML, 7, 14–18, 21–23, 29–33
GenericML_single, 8, 10, 12, 29–33
get_BLP, 10, 15, 17, 18, 22
get_CLAN, 10, 15, 16, 18, 22
get_GATES, 10, 15, 17, 17, 22
ggplot, 22

lambda_parameters, 14, 19
lm, 3, 6

Med, 20

plot, 10, 15, 17, 18
plot.GenericML, 21
print, 10
print.GenericML, 23
propensity_score, 3, 6, 7, 24
proxy_BCA, 3, 6, 7, 10, 14, 25, 28
proxy_CATE, 3, 6, 7, 10, 14, 26, 27

quantile_group, 4–6, 19, 28

setup_diff, 3–7, 9, 10, 13, 22, 29, 31, 33
setup_vcov, 3, 6, 7, 9, 10, 13, 30, 30, 33
setup_X1, 3, 6, 7, 9, 10, 13, 30, 31, 32

TrueIfUnix, 33