

# Package ‘RCTrep’

March 23, 2023

**Type** Package

**Title** Validation of Estimates of Average Treatment Effects Derived from Observational Data

**Version** 1.0.0

**Author** person(“Lingjie”, “Shen”, email = “L.Shen@uvt.nl”, role = c(“aut”, “cre”))

**Maintainer** The package maintainer <yourself@somewhere.net>

**Description** The software compares and validate estimates of (conditional) average treatment effects obtained using observational data by a) making it easy to obtain and visualize estimates derived using a large variety of methods, and b) ensuring that estimates are easily compared to a gold standard (i.e., estimates derived from randomized controlled trials). RCTrep offers a generic protocol for treatment effect validation based on four simple steps, namely, set-selection, estimation, diagnosis, and validation. The package provides a simple dashboard to review the obtained results.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Imports** mvtnorm,  
MASS,  
MatchIt,  
ggplot2,  
ggpubr,  
PSweight,  
numDeriv,  
R6,  
dplyr,  
geex,  
optmatch,  
BART,  
fastDummies,  
tidyr,  
copula,  
shiny,  
shinydashboard,  
glue,  
stats,  
utils

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

**Config/testthat/edition** 3

**RoxygenNote** 7.1.2

**VignetteBuilder** knitr

**Depends** R (>= 2.10),  
base

**R topics documented:**

call_dashboard . . . . .	2
DGM . . . . .	3
DR . . . . .	5
Fusion . . . . .	6
GenerateSyntheticData . . . . .	7
IPW . . . . .	7
quasar.obj . . . . .	8
quasar.synthetic . . . . .	8
RCTREP . . . . .	9
SEstimator_wrapper . . . . .	12
source.data . . . . .	12
target.data . . . . .	13
TEstimator . . . . .	14
TEstimator_wrapper . . . . .	16
<b>Index</b>	<b>18</b>

---

call_dashboard	<i>Visualizing validation results according to four steps, namely, set-selection, estimation, diagnosis, and validation</i>
----------------	---

---

**Description**

Visualizing validation results according to four steps, namely, set-selection, estimation, diagnosis, and validation

**Usage**

call\_dashboard(source.obj = NULL, target.obj = NULL, source.obj.rep = NULL)

**Arguments**

- source.obj an instantiated object of class TEstimator. The estimates of conditional average treatment effects are compared to those from target.obj.
- target.obj an instantiated object of class TEstimator. The estimates of conditional average treatment effect are regarded as unbiased of truth.
- source.obj.rep an instantiated object of class SEstimator. The estimates of conditional average treatment effects are compared to those from target.obj.

## Examples

```
## Not run:
source.data <- RCTrep::source.data
target.data <- RCTrep::target.data

vars_name <- list(confounders_treatment_name = c("x1", "x2", "x3", "x4", "x5", "x6"),
                  treatment_name = c('z'),
                  outcome_name = c('y'))
)
confounders_sampling_name <- c("x2", "x6")

Estimate conditional average treatment effect
source.obj <- TEstimator_wrapper(
  Estimator = "G_computation",
  data = source.data,
  vars_name = vars_name,
  outcome_method = "glm",
  outcome_form = y ~ x1 + x2 + x3 + z + z:x1 + z:x2 + z:x3 + z:x6,
  name = "RWD",
  data.public = FALSE
)

target.obj <- TEstimator_wrapper(
  Estimator = "Crude",
  data = target.data,
  vars_name = vars_name,
  name = "RCT",
  data.public = FALSE,
  isTrial = TRUE
)

Estimate the weighted conditional average treatment effect of source.obj
strata <- c("x1", "x4")
source.rep.obj <- SEstimator_wrapper(Estimator = "Exact",
                                     target.obj = target.obj,
                                     source.obj = source.obj,
                                     confounders_sampling_name =
                                       confounders_sampling_name)
source.rep.obj$EstimateRep(stratification = strata, stratification_joint = TRUE)

call_dashboard(source.obj = source.obj,
               target.obj = target.obj,
               source.obj.rep = source.obj.rep)

## End(Not run)
```

## Description

Generating RCT data or observational data for examples used in the package

**Usage**

```
DGM(
  trial,
  n,
  var_name,
  p_success,
  tau,
  y0,
  log.ps = NULL,
  binary = FALSE,
  noise = 1,
  ...
)
```

**Arguments**

<code>trial</code>	Logical indicating whether the treatment is randomly assigned in the generated data. If TRUE, RCT data is generated. Otherwise, observational data is generated.
<code>n</code>	A numeric value indicating the number of observations in the generated data
<code>var_name</code>	A character vector indicating the names of covariates
<code>p_success</code>	the success probability of binary variables of covariates
<code>tau</code>	a character indicating the generation of the true treatment effect of each individual
<code>y0</code>	a character indicating the generation of the potential outcome under control
<code>log.ps</code>	a numeric value indicating the logit of propensity score
<code>binary</code>	logical indicating whether the outcome is binary or continuous variable
<code>noise</code>	a numeric value indicating the standard error of normal distribution for noise term of continuous outcome generation
<code>...</code>	an optional argument indicating pairwise correlation between covariates.

**Value**

a data frame

**Examples**

```
## Not run:
n_rct <- 5000; n_rwd <- 5000
var_name <- c("x1", "x2", "x3", "x4", "x5", "x6")
p_success_rct <- c(0.7, 0.9, 0.2, 0.3, 0.2, 0.3)
p_success_rwd <- c(0.2, 0.2, 0.8, 0.8, 0.7, 0.8)
tau <- "6*x2+x6+2"
y0 <- "x1"
log.ps <- "x1*x2+x3*x4+5*x5+x6"
rho1 <- c("x1", "x2", 0)
rho2 <- c("x2", "x3", 0)
target.data <- RCTrep::DGM(trial=TRUE, n_rct, var_name,
                           p_success_rct, tau, y0, log.ps=0,
                           binary = FALSE, noise=1, rho1, rho2)
source.data <- RCTrep::DGM(trial=FALSE, n_rwd, var_name,
```

```

p_success_rwd, tau, y0, log.ps,
binary = FALSE, noise=1, rho1, rho2)

## End(Not run)

```

DR

*R6 class: Doubly robust estimator base class***Description**

A base R6 class for doubly robust estimator of average treatment effect that implements comment methods.

**Super class**

`RCTrep:TEstimator` -> DR

**Methods****Public methods:**

- `DR$new()`
- `DR$clone()`

**Method new():**

*Usage:*

```

DR$new(
  df,
  vars_name,
  name,
  treatment_method,
  treatment_formula,
  outcome_method,
  outcome_formula,
  two_models,
  isTrial,
  ...
)

```

**Method clone():** The objects of this class are cloneable with this method.

*Usage:*

```
DR$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

---

Fusion	<i>Validation of estimates of conditional average treatment effects in objects of class TEstimator and SEstimator.</i>
--------	--

---

## Description

Validation of estimates of conditional average treatment effects in objects of class TEstimator and SEstimator.

Validation of estimates of conditional average treatment effects in objects of class TEstimator and SEstimator.

## Methods

### Public methods:

- `Fusion$new()`
- `Fusion$plot()`
- `Fusion$print()`
- `Fusion$evaluate()`
- `Fusion$clone()`

### Method `new()`:

*Usage:*

```
Fusion$new(..., stratification = NULL, stratification_joint = NULL)
```

*Arguments:*

... objects of class TEstimator and SEstimator.

`stratification` a character vector specifying variables. The variables are used to select subpopulations individually or in combination depending on `stratification_joint`. Default value is `NULL`.

`stratification_joint` a logical indicating if subpopulations are selected based on levels of individual variable in `stratification` or levels of combined variables in `stratification`. Default value is `NULL`.

### Method `plot()`:

*Usage:*

```
Fusion$plot()
```

### Method `print()`:

*Usage:*

```
Fusion$print()
```

### Method `evaluate()`:

*Usage:*

```
Fusion$evaluate()
```

### Method `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
Fusion$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

---

GenerateSyntheticData	<i>Generating the synthetic RCT data given marginal distribution of each covariate</i>
-----------------------	--

---

### Description

Generating the synthetic RCT data given marginal distribution of each covariate

### Usage

```
GenerateSyntheticData(margin_dis, N, margin, var_name, pw.cor = 0)
```

### Arguments

margin_dis	a character indicating the distribution of each variable, allowable options is "bernoulli_categorical" and "bernoulli"
N	a numeric value indicating the sample size for the simulated data
margin	a list containing the marginal distribution of variables; if margin_dis="bernoulli_categorical", then margin should be list(x1=c("x1",nlevels(x1),level1, level2,...,leveln, plevel1, plevel2,...,plevel3), x2=c(...)); if margin_dis="bernoulli", margin=list(p(x1=1),p(x2=1),...,p(xn=1))
var_name	a vector indicating the name of variables, the order should be in line with margin
pw.cor	a vector containing the pairwise correlation of these variables, default is NULL; when margin_dis="bernoulli", then pw.cor must be specified

---

IPW	<i>R6 class: Inverse propensity score weighting estimator base class</i>
-----	--

---

### Description

A base R6 class for inverse propensity score weighting estimator of average treatment effect that implements comment methods.

### Super class

```
RCTrep::TEstimator -> IPW
```

### Methods

#### Public methods:

- `IPW$new()`
- `IPW$diagnosis_t_ignorability()`
- `IPW$clone()`

#### Method new():

*Usage:*

```
IPW$new(df, vars_name, name, treatment_method, treatment_formula, isTrial, ...)
```

#### Method diagnosis\_t\_ignorability():

*Usage:*

```
IPW$diagnosis_t_ignorability(stratification, stratification_joint = TRUE)
```

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
IPW$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

---

quasar.obj

*An object of class TEstimator\_Synthetic using quasar.synthetic*

---

## Description

An object of class TEstimator\_Synthetic using quasar.synthetic

## Usage

```
quasar.obj
```

## Format

An object of class Synthetic\_TEstimator (inherits from TEstimator, R6) of length 14.

---

quasar.synthetic

*A synthetic QUASAR trial dataset, where outcome is a binary variable, treatment is a binary variable.*

---

## Description

A synthetic QUASAR trial dataset, where outcome is a binary variable, treatment is a binary variable.

## Usage

```
quasar.synthetic
```

## Format

## 'quasar.synthetic' A data frame with 5934 rows and 3 variables:

**Stage2** binary variable, 1 indicating stage 2 and 0 indicating stage 3

**male** binary variable, 1 indicating male and 0 indicating female

**age** categorical variable, 1 indicating [23,50], 2 indicating [50,59], 3 indicating [60,69], 4 indicating [70,86]



RCTREP

*Replicate treatment effect estimates obtained from a randomized control trial using real world data*

## Description

The function RCTREP is used to replicate the estimate of treatment effect from a target randomized control trial based on real-world data (RWD). This function estimate the treatment effect of RWD to ensure the internal validity of the estimates within the study population (namely, the observational data) and weight the resulting estimates to the target population (namely, the RCT) to enable external validity. The function currently implement the following types of estimators of treatment effect: G\_computation, inverse propensity score weighting (IPW), and augmented propensity score weighting. The function implement the following two types of weighting estimators to generalize the resulting estimates of treatment effect from RWD to the target RCT: exact matching weights, and selection score weights. Since we regard the sample in the RCT as the target population, weights for each individual in RWD is  $p/(1 - p)$  so that the weighted population of RWD is representative to the target population.

## Usage

```
RCTREP(
  TEstimator = "G_computation",
  SEstimator = "Exact",
  source.data = source.data,
  target.data = target.data,
  source.name = "RWD",
  target.name = "RCT",
  vars_name,
  confounders_sampling_name,
  outcome_method = "glm",
  treatment_method = "glm",
  weighting_method = "glm",
  outcome_formula = NULL,
  treatment_formula = NULL,
  selection_formula = NULL,
  stratification = NULL,
  stratification_joint = FALSE,
  strata_cut_source = NULL,
  strata_cut_target = NULL,
  two_models = FALSE,
  data.public = TRUE,
  ...
)
```

## Arguments

TEstimator	A character specifying an estimator for average treatment effect. The allowed estimators for TEstimator are: "G_computation", "IPW", and "DR". The corresponding object will be created by the wrapper function TEstimator_wrapper(). The default is "G_computation", which, along with outcome_method="glm" model the potential outcomes.
------------	--

<code>SEstimator</code>	A character specifying an estimator for weight. The allowed estimators are: "Exact", "Subclass", "ISW". The default is "Exact", which, implements the exact matching on variables in <code>confounders_sampling_name</code> to balance the population covariates between <code>source.data</code> and <code>target.data</code> .
<code>source.data</code>	A data frame containing variables named in <code>vars_name</code> and possible other variables. <code>source.obj</code> is instantiated using <code>source.data</code> .
<code>target.data</code>	A data frame containing variables named in <code>vars_name</code> and possible other variables. <code>target.obj</code> is instantiated using <code>target.data</code> .
<code>source.name</code>	A character indicating the name of <code>source.obj</code> .
<code>target.name</code>	A character indicating the name of <code>target.obj</code> .
<code>vars_name</code>	A list containing four vectors <code>confounders_treatment_name</code> , <code>treatment_name</code> , and <code>outcome_name</code> . <code>confounders_treatment_name</code> is a character vector containing the adjustment variables, which, along with <code>TEstimator</code> and the corresponding <code>outcome_method</code> or <code>treatment_method</code> to correct for confounding; <code>outcome_name</code> is a character vector of length one containing the variable name of outcome; <code>treatment_name</code> is a character vector of length one containing the variable name of treatment.
<code>confounders_sampling_name</code>	a character vector specifying variable names. The weights are estimated using the estimator specified in <code>SEstimator</code> based on the variables.
<code>outcome_method, treatment_method, weighting_method</code>	A character specifying which model for outcome, treatment, and weight to use. Possible values are found using <code>names(getModelInfo())</code> . See <a href="http://topepo.github.io/caret/train-models-by-tag.html">http://topepo.github.io/caret/train-models-by-tag.html</a> .
<code>outcome_formula, treatment_formula, selection_formula</code>	An optional object of class <code>formula</code> describing the outcome model specification, treatment model specification, and selection model specification.
<code>stratification</code>	An optional character vector containing variables to select subgroups. <code>source.obj</code> will compute both weighted and unweighted average treatment effects of the subgroups, <code>target.obj</code> will calculate the average treatment effects of the subgroups.
<code>stratification_joint</code>	An optional logical indicating if the subgroups are selected based on levels of combined variables in <code>stratification</code> or levels of individual variable in <code>stratification</code> .
<code>strata_cut_source</code>	An optional list containing lists. Each component is a list with tag named by a variable in <code>source.data</code> to discretize, containing <code>break</code> which is a vector specifying the interval of range of the variable to divide, <code>label</code> which is a character vector specifying how to code value in the variable according to which interval they fall. The leftmost interval corresponds to level one, the next leftmost to level two and so on. This parameter is useful in the case we concern the integrated treatment effect conditioning on variables with multiple levels (for instance, continuous variable or ordinal variable with multiple levels). Note that we first model based on these continuous variables, then we discretize these variables according to <code>strata_cut</code> . The variables in data of <code>TEstimator</code> object are discretized, and the weight is calculated based on the discretized variables.
<code>strata_cut_target</code>	An optional list containing lists. Each component is a list with tag named by a variable in <code>target.data</code> to discretize. The

<code>two_models</code>	An optional logical indicating whether potential outcomes should be modeled separately when <code>TEstimator="DR"</code> . Default is <code>FALSE</code> .
<code>data.public</code>	An optional logical indicating whether the data in the output objects are public. Default is <code>TRUE</code> .
<code>...</code>	An optional argument passed to <code>fit()</code> of each estimator object for model training and tuning. See <a href="https://topepo.github.io/caret/model-training-and-tuning.html">https://topepo.github.io/caret/model-training-and-tuning.html</a> for details.

## Details

An R6 object is constructed by a wrapper function `TEstimator_wrapper` and `SEstimator_wrapper` with user's input of data and estimators for treatment effect and weight. `TEstimator_wrapper()` returns initialized objects `source.obj` and `target.obj`. `TEstimator_wrapper()` replicates the target RCT estimates of `target.obj` via the class method `RCTrep()` with input of source object `source.obj`, target object `target.obj`, and estimator of weights `SEstimator`.

## Value

A list of length two with two R6 class objects `source.obj=source.obj` and `target.obj=target.obj`.

## Examples

```
## Not run:
library(RCTrep)
source.data <- RCTrep::source.data
target.data <- RCTrep::target.data
Estimator <- "IPW"
strata <- c("Stage2", "pT")
strata_joint <- TRUE
vars_name <- list(
  confounders_internal = c("Stage2", "age", "pT"),
  confounders_external = c("Stage2", "age", "pT"),
  treatment_name = c("combined_chemo"),
  outcome_name = c("vitstat")
)
outcome_form <- vitstat ~ Stage2 + age + combined_chemo + pT +
  Stage2:combined_chemo + age:combined_chemo + pT:combined_chemo + pT:Stage2:combined_chemo
strata_cut <- list(age = list(
  breaks = c(
    min(data$age),
    50, 60, 70, max(data$age)
  ),
  labels = c(1, 2, 3, 4)
))
output <- RCTREP(
  Estimator = "G_computation", two_models = FALSE,
  source.data = source.data, target.data = target.data,
  vars_name = vars_name,
  outcome_formula = outcome_form,
  stratification = strata, stratification_joint = TRUE, strata_cut = strata_cut
)
output$source.obj
output$target.obj

## End(Not run)
```

---

<code>SEstimator_wrapper</code>	<i>R6 class which estimates the weighted conditional average treatment effects in <code>source.obj</code> based on input objects <code>source.obj</code> and <code>target.obj</code> of class <code>TEstimator</code>.</i>
---------------------------------	--

---

### Description

R6 class which estimates the weighted conditional average treatment effects in `source.obj` based on input objects `source.obj` and `target.obj` of class `TEstimator`.

### Usage

```
SEstimator_wrapper(
  Estimator,
  target.obj,
  source.obj,
  confounders_sampling_name,
  method = "glm",
  sampling_formula = NULL,
  ...
)
```

### Arguments

<code>Estimator</code>	a character specifying an estimator for weight. The allowed estimators are Exact, ISW, and Subclass.
<code>target.obj</code> , <code>source.obj</code>	an instantiated object of class <code>TEstimator</code> .
<code>confounders_sampling_name</code>	a character vector specifying the names of variables in data of <code>source.obj</code> and <code>target.obj</code> . Weights are estimated based on the variables.
<code>method</code>	an optional character specifying a model for estimating sampling probability once <code>Estimator='ISW'</code> or <code>Estimator='Subclass'</code> .
<code>sampling_formula</code>	an object of class <code>formula</code> specifying a model specification for sampling probability. Default value is <code>NULL</code> .
<code>...</code>	an optional argument specifying training and tuning for a model of sampling probability. See <a href="https://topepo.github.io/caret/model-training-and-tuning.html">https://topepo.github.io/caret/model-training-and-tuning.html</a> for details.

---

<code>source.data</code>	<i>A data set of simulated observational data, where outcome is continuous variable, treatment is a binary variable.</i>
--------------------------	--

---

### Description

A data set of simulated observational data, where outcome is continuous variable, treatment is a binary variable.

**Usage**

```
source.data
```

**Format**

```
## 'source.data' A data frame with 5000 rows and 8 variables:
```

```
x1 binary variable,  $x1 \sim \text{rbinom}(5000, 1, 0.2)$ 
```

```
x2 binary variable,  $x2 \sim \text{rbinom}(5000, 1, 0.2)$ 
```

```
x3 binary variable,  $x3 \sim \text{rbinom}(5000, 1, 0.8)$ 
```

```
x4 binary variable,  $x4 \sim \text{rbinom}(5000, 1, 0.8)$ 
```

```
x5 binary variable,  $x5 \sim \text{rbinom}(5000, 1, 0.7)$ 
```

```
x6 binary variable,  $x6 \sim \text{rbinom}(5000, 1, 0.8)$ 
```

```
z binary variable indicating treatment and control.  $pp = x1*x2+x3*x4+5*x5+x6$ ,  $p(z=1) = p = 1/(1+e^{-(pp-\text{mean}(pp))/\text{sd}(pp)*\sqrt{3}}/\pi)$ ,  $z \sim \text{rbinom}(5000, 1, p)$ 
```

```
y continuous variable indicating outcome,  $y \sim x1 + 6*x2+x6+2*z + \text{rnorm}(5000, 0, 1)$ 
```

---

```
target.data
```

```
A data set of simulated RCT data, where outcome is continuous variable, treatment is a binary variable.
```

---

**Description**

A data set of simulated RCT data, where outcome is continuous variable, treatment is a binary variable.

**Usage**

```
target.data
```

**Format**

```
## 'target.data' A data frame with 5000 rows and 8 variables:
```

```
x1 binary variable,  $x1 \sim \text{rbinom}(5000, 1, 0.7)$ 
```

```
x2 binary variable,  $x2 \sim \text{rbinom}(5000, 1, 0.9)$ 
```

```
x3 binary variable,  $x3 \sim \text{rbinom}(5000, 1, 0.2)$ 
```

```
x4 binary variable,  $x4 \sim \text{rbinom}(5000, 1, 0.3)$ 
```

```
x5 binary variable,  $x5 \sim \text{rbinom}(5000, 1, 0.2)$ 
```

```
x6 binary variable,  $x6 \sim \text{rbinom}(5000, 1, 0.3)$ 
```

```
z binary variable indicating treatment and control,  $z \sim \text{rbinom}(5000, 1, 0.5)$ 
```

```
y continuous variable indicating outcome,  $y \sim x1 + 6*x2+x6+2*z + \text{rnorm}(5000, 0, 1)$ 
```

TEstimator

*R6 class: Estimator base class*

## Description

A base R6 class for estimator of average treatment effect that implements the common methods, such as RCTrep, get\_CATE(), plot\_CATE(), inherited by G\_computation, IPW, and DR class.

## Methods

### Public methods:

- `TEstimator$new()`
- `TEstimator$get_CATE()`
- `TEstimator$plot_CATE()`
- `TEstimator$plot_y1_y0()`
- `TEstimator$diagnosis_t_overlap()`
- `TEstimator$diagnosis_y_overlap()`
- `TEstimator$diagnosis_t_ignorability()`
- `TEstimator$clone()`

**Method** `new()`: Create a new Estimator object

*Usage:*

```
TEstimator$new(df, vars_name, name)
```

*Arguments:*

`df` A data frame containing variables in `vars_name`

`vars_name` `vars_name` A list containing four vectors `confounders_internal`, `treatment_name`, and `outcome_name`. `confounders_internal` is a character vector containing the adjustment variables, which, along with `Estimator` and the corresponding `outcome_method` or `treatment_method` to correct for confounding; `outcome_name` is a character vector of length one containing the variable name of outcome; `treatment_name` is a character vector of length one containing the variable name of treatment.

**Method** `get_CATE()`: Replicating the average treatment effect of `target.obj`. If stratification is specified, then replicating the conditional average treatment effect stratified by `stratification` and `stratification_joint` by weighting based on the residual variables, namely, variables that are specified in `confounders_treatment_name` while not in `stratification`.

Get conditional average treatment effect of subgroups defined by `stratification` and `stratification_joint`. If `stratification_joint=FALSE`, then the method return conditional average treatment effect of subgroups stratified by each of variables in `stratification`.

*Usage:*

```
TEstimator$get_CATE(stratification, stratification_joint = TRUE)
```

*Arguments:*

`stratification` An optional string vector containing variables to define subgroup. If `!is.NULL(stratification)` `source.obj` will compute both weighted and unweighted conditional average treatment effect based on these variables, `target.obj` will calculate the conditional average treatment effect based on these variables.

`stratification` An string vector containing variables to define subgroup.

`stratification_joint` An optional logical defining the subgroup based on joint distribution of variables or univariate distribution in stratification when stratification is specified.

`stratification_joint` An logical defining the subgroup based on joint distribution of variables or univariate distribution in stratification.

`target.obj` An object of class Estimator or list.

`weighting_estimator` A string specifying a weighting estimator for generalizing/transporting the estimates to `target.obj`. The allowed estimators are: "balancing", and "modeling".

`weighting_method` A string specifying which model for selection to use. Possible values are found using `names(getModelInfo())`. See <http://topepo.github.io/caret/train-models-by-tag.html>.

**Returns:** A data frame. If `stratification_joint=TRUE`, then the method returns a data frame with `N` rows and `J` columns, where `N` represents the number of subgroups, and `J` is equal to the sum of number of variables in stratification and 3 (three additional columns with name `cate`, `se`, and `size`, representing the estimated conditional average treatment effect of this subgroup, standard error of the estimate, and the sample size of the subgroup). If `stratification_joint=FALSE`, then the method returns a data frame with `N` rows and 5 columns, where `N` represents the number of subgroups stratified by each variable in stratification and 5 columns with name `name`, `value`, `cate`, `se`, and `size`, representing the name of a variable used to stratify the population, a level of the variable, the estimated conditional average treatment effect of this subgroup, standard error of the estimate, and the sample size of the subgroup).

**Method** `plot_CATE()`: Plot the forest plot of conditional average treatment effect of subgroups defined by stratification and `stratification_joint`. The method first call public method `get_CATE(stratification, stratification_joint)`, then plot the results.

*Usage:*

```
TEstimator$plot_CATE(
  stratification = private$confounders_treatment_name,
  stratification_joint = TRUE
)
```

*Arguments:*

`stratification` An string vector containing variables to define subgroup.

`stratification_joint` An logical defining the subgroup based on joint distribution of variables or univariate distribution in stratification.

**Method** `plot_y1_y0()`:

*Usage:*

```
TEstimator$plot_y1_y0(
  stratification,
  stratification_joint = TRUE,
  seperate = FALSE
)
```

**Method** `diagnosis_t_overlap()`:

*Usage:*

```
TEstimator$diagnosis_t_overlap(stratification, stratification_joint = TRUE)
```

**Method** `diagnosis_y_overlap()`:

*Usage:*

```
TEstimator$diagnosis_y_overlap(stratification, stratification_joint = TRUE)
```

**Method** diagnosis\_t\_ignorability():

*Usage:*

```
TEstimator$diagnosis_t_ignorability()
```

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
TEstimator$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

---

TEstimator_wrapper	<i>Estimator for average treatment effect wrapper function</i>
--------------------	--

---

## Description

The function Estimate is used to estimate the average treatment effect obtained from data. If TEstimator is G\_computation and outcome\_method is BART, then make sure that treatment and outcome column is integer/numeric, no matter the outcome is binary or continuous.

## Usage

```
TEstimator_wrapper(
  Estimator,
  data,
  vars_name,
  name = "",
  outcome_method = "glm",
  treatment_method = "glm",
  two_models = FALSE,
  outcome_formula = NULL,
  treatment_formula = NULL,
  data.public = TRUE,
  isTrial = FALSE,
  strata_cut = NULL,
  ...
)
```

## Arguments

Estimator	A character specifying an estimator for conditional average treatment effect. The allowed estimators for TEstimator are: "G_computation", "IPW", and "DR". The corresponding object will be created by the wrapper function TEstimator_wrapper(). The default is "G_computation", which, along with outcome_method="glm" model the potential outcomes.
data	A data frame containing variables named in vars_name and possible other variables.



vars_name	A list containing four vectors confounders_treatment_name, treatment_name, and outcome_name. confounders_treatment_name is a character vector containing the adjustment variables, which, along with TEstimator and the corresponding outcome_method or treatment_method to correct for confounding; outcome_name is a character vector of length one containing the variable name of outcome; treatment_name is a character vector of length one containing the variable name of treatment.
name	A character indicating the name of the output object
outcome_method	A character specifying a model for outcome. Possible values are found using names(getModelInfo()). See <a href="http://topepo.github.io/caret/train-models-by-tag.html">http://topepo.github.io/caret/train-models-by-tag.html</a> . Default is "glm".
treatment_method	A character specifying a model for treatment. Possible values are found using names(getModelInfo()). See <a href="http://topepo.github.io/caret/train-models-by-tag.html">http://topepo.github.io/caret/train-models-by-tag.html</a> . Default is "glm".
two_models	An optional logical indicating whether potential outcomes should be modeled separately when TEstimator="DR". Default is FALSE.
outcome_formula	An optional object of class formula describing the outcome model specification when Estimator="G_computation" or Estimator="DR".
treatment_formula	An optional object of class formula describing the treatment model specification when Estimator="IPW" or Estimator="DR"
data.public	An optional logical indicating whether individual-level data is public in the output object. Default is TRUE.
isTrial	An optional logical indicating whether the treatment assignment of data is random or unknown.
strata_cut	An optional list containing lists. Each component is a list with tag named by a variable in data to discretize, containing break which is a vector specifying the interval of range of the variable to divide, lable which is a character vector specifying how to code value in the variable according to which interval they fall. The leftmost interval corresponds to level one, the next leftmost to level two and so on. This parameter is useful in the case we concern the integrated treatment effect conditioning on variables with multiple levels (for instance, continuous variable or ordinal variable with multiple levels). Note that we first model based on these continuous variables, then we discretize these variables according to strata_cut. The variables in data of the output object are discretized.
...	An optional argument passed to the private function fit() of each class for model training and tuning. See <a href="https://topepo.github.io/caret/model-training-and-tuning.html">https://topepo.github.io/caret/model-training-and-tuning.html</a> for details.

## Value

An object of class TEstimator.

# Index

## \*Topic **datasets**

- quasar.obj, [8](#)
- quasar.synthetic, [8](#)
- source.data, [12](#)
- target.data, [13](#)

call\_dashboard, [2](#)

DGM, [3](#)

DR, [5](#)

Fusion, [6](#)

GenerateSyntheticData, [7](#)

IPW, [7](#)

quasar.obj, [8](#)

quasar.synthetic, [8](#)

RCTREP, [9](#)

RCTrep::TEstimator, [5](#), [7](#)

SEstimator\_wrapper, [11](#), [12](#)

source.data, [12](#)

target.data, [13](#)

TEstimator, [12](#), [14](#)

TEstimator\_wrapper, [11](#), [16](#)