Package: RCTrep (via r-universe)

August 22, 2024

Type Package

Title Validation of Estimates of Treatment Effects in Observational Data

Version 1.0.0

Author Lingjie Shen [aut, cre, cph], Gijs Geleijnse [aut], Maurits Kaptein [aut]

Maintainer Lingjie Shen <602249910@qq.com>

Description Validates 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 (G-computation, inverse propensity score weighting, etc.), 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. 'RCTrep' provides a simple dashboard to review the obtained results. The validation approach is introduced by Shen, L., Geleijnse, G. and Kaptein, M. (2023) <doi:10.21203/rs.3.rs-2559287/v1>.

License MIT + file LICENSE

URL https://github.com/duolajiang/RCTrep

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, caret

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.1.2

VignetteBuilder knitr

Depends R (>= 2.10), base

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

RemoteUrl https://github.com/duolajiang/rctrep

RemoteRef HEAD

RemoteSha 9b53d9ea2164afe630aca908d984e771043fcf05

Contents

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

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 effects 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`.

Value

an interactive interface visualizing results of four steps

Examples

```
source.data <- RCTrep::source.data[sample(dim(RCTrep::source.data)[1],500),]
target.data <- RCTrep::target.data[sample(dim(RCTrep::target.data)[1],500),]

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")

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
)

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)
```

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

Description

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

Usage

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

Arguments

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

Value

a data frame; column names are variables names, z, y

Examples

```
n_rct <- 500; n_rwd <- 500
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)
```

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

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`.

Value

an R6 object

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 subgroups individually or in combination depending on `stratification_joint`. Default value is `NULL`.

`stratification_joint` a logical indicating if subgroups are selected based on levels of individual variable in `stratification` or levels of combined variables in `stratifiation`. 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.

Examples

```
source.data <- RCTrep::source.data[sample(dim(RCTrep::source.data)[1],500),]
target.data <- RCTrep::target.data[sample(dim(RCTrep::target.data)[1],500),]

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")

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
)

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)

fusion <- Fusion$new(target.obj,
                     source.obj,
                     source.rep.obj)
fusion$plot()
fusion$evaluate()
```

GenerateSyntheticData *Generating the synthetic RCT data given marginal distribution of each covariate*

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 are `"bernoulli_categorical"` and `"bernoulli"`. If some variables have two categories and some have more than two categories, `"bernoulli_categorical"` should be specified; if all variables have two categories, `"bernoulli"` should be specified
`N`	a numeric value specifying 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("x2",...)); 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 of variables should be aligned with `margin`
`pw.cor`	a vector specifying the pairwise correlations of the variables, default is 0; when margin_dis="bernoulli", then pw.cor must be specified.

Value

a data frame with columns names x1, x2,....

`quasar.agg` *Aggregated data derived from paper of QUASAR trial*

Description

Aggregated data derived from paper of QUASAR trial

Usage

```
quasar.agg
```

Format

An object of class `list` of length 5.

`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 `TEstimator_Synthetic` (inherits from `TEstimator`, `R6`) of length 15.

`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 observational data*

Description

The function `RCTREP` is used to validate the estimates of treatment effects obtained from observational data by comparing to estimates from a target randomized control trial. The function currently implements the following types of estimators of treatment effects: G_computation, inverse propensity score weighting (IPW), and augmented propensity score weighting. The function implements the following three types of weighting estimators to compare the resulting estimates of treatment effects from RWD to the target RCT: exact matching weights, inverse selection probability weighting, and sub-classification. Since we regard the sample in the RCT as the target population, weights for each individual in observational data is $p/(1-p)$ so that the weighted population of observational data 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 conditional average treatment effects. 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"` models the potential outcomes.
`SEstimator`	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 `confounders_sampling_name` to balance the population covariates between `source.data` and `target.data`.
`source.data`	A data frame containing variables named in `vars_name` and possible other variables. `source.obj` is instantiated using `source.data`.
`target.data`	A data frame containing variables named in `vars_name` and possible other variables. `target.obj` is instantiated using `target.data`.
`source.name`	A character indicating the name of `source.obj`.
`target.name`	A character indicating the name of `target.obj`.
`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.
`confounders_sampling_name`	a character vector specifying variable names. The weights are estimated based on the variables.

`outcome_method`, `treatment_method`, `weighting_method`
: A character specifying model for outcome, treatment, and weight to use. Possible values are found using `names(getModelInfo())`. See `http://topepo.github.io/caret/train-models-by-tag.html`.

`outcome_formula`, `treatment_formula`, `selection_formula`
: An optional object of class `formula` describing the outcome model specification, treatment model specification, and selection model specification.

`stratification`
: An optional character vector containing variables to select subgroups. `source.obj` will compute both weighted and unweighted average treatment effects of the subgroups, `targe.obj` will calculate the average treatment effects of the subgroups.

`stratification_joint`
: An optional logical indicating if the subgroups are selected based on levels of combined variables in `stratification` or levels of individual variable in `stratification`.

`strata_cut_source`
: An optional list containing lists. Each component is a list with tag named by a variable in `source.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 `TEstimator` object are discretized, and the weight is calculated based on the discretized variables.

`strata_cut_target`
: An optional list containing lists. Each component is a list with tag named by a variable in `target.data` to discretize.

`two_models`
: An optional logical indicating whether potential outcomes should be modeled separately when `TEstimator="DR"`. Default is `FALSE`.

`data.public`
: An optional logical indicating whether the `data` in the output objects are public. Default is `TRUE`.

`...`
: An optional argument passed to `fit()` of each estimator object for model training and tuning. See `https://topepo.github.io/caret/model-training-and-tuning.html` 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`. `SEstimator_wrapper()` weights the estimates of `source.obj` via the class method `RCTrep()`. The weights are computed using data in the source object `source.obj`, target object `target.obj`, and estimator of weights `SEstimator`.

Value

A list of length three with three R6 class objects, `source.obj`, `target.obj` and `source.rep.obj`

Examples

```
output <- RCTREP(TEstimator = "G_computation", SEstimator = "Exact",
                 outcome_method = "BART",
             source.data = RCTrep::source.data[sample(dim(RCTrep::source.data)[1],500),],
             target.data = RCTrep::target.data[sample(dim(RCTrep::target.data)[1],500),],
                 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"),
                 stratification = c("x1","x3","x4","x5"),
                 stratification_joint = TRUE)
output$target.obj
output$source.obj
output$source.rep.obj
```

`SEstimator_wrapper` *Estimating the weighted conditional average treatment effects in* `source.obj` *based on input objects* `source.obj` *and* `target.obj` *of class* `TEstimator`.

Description

Estimating 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

`Estimator`	a character specifying an estimator for weight. The allowed estimators are `"Exact"`, `"ISW"`, and `"Subclass"`.

`target.obj`, `source.obj`
: an instantiated object of class `TEstimator`.

`confounders_sampling_name`
: a character vector specifying the names of variables in `data` of `source.obj` and `target.obj`. Weights are estimated based on the variables.

`method`
: an optional character specifying a model for estimating sampling probability when `Estimator='ISW'` or `Estimator='Subclass'`.

`sampling_formula`
: an object of class `formula` specifying a model specification for sampling probability. Default value is `NULL`.

`...`
: an optional argument specifying training and tuning for a model of sampling probability. See `https://topepo.github.io/caret/model-training-and-tuning.html` for details.

Value

An object of class `SEstimator`

Examples

```
source.data <- RCTrep::source.data[sample(dim(RCTrep::source.data)[1],500),]
target.data <- RCTrep::target.data[sample(dim(RCTrep::target.data)[1],500),]

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

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

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 = TRUE)

source.rep.obj <- SEstimator_wrapper(Estimator="Exact",
                                     target.obj=target.obj,
                                     source.obj=source.obj,
                                     confounders_sampling_name=c("x2","x6"))
source.rep.obj$EstimateRep(stratification = c("x1","x3","x4","x5"),
                           stratification_joint = TRUE)
```

`source.binary.data` *A dataset of simulated observational data, where outcome is binary variable. The data is filtered after compared to target.binary.data*

Description

A dataset of simulated observational data, where outcome is binary variable. The data is filtered after compared to target.binary.data

Usage

```
source.binary.data
```

Format

A data frame with 2624 rows and 9 variables.

x1 binary variable, x1 ~ rbinom(5000,1,0.2)

x2 binary variable, x2 ~ rbinom(5000,1,0.2)

x3 binary variable, x3 ~ rbinom(5000,1,0.8)

x4 binary variable, x4 ~ rbinom(5000,1,0.8)

x5 binary variable, x5 ~ rbinom(5000,1,0.7)

x6 binary variable, x6 ~ rbinom(5000,1,0.8)

z binary variable. pp = x1*x2+x3*x4+5*x5+x6, p(z=1) = p = 1/1+e^-(pp-mean(pp))/sd(pp)*sqrt(3)/pi, z ~ rbinom(5000,1,p)

y binary variable. pp = x1 + (6*x2+x6+2)*z, p(y=1) = p = 1/1+e^-(pp-mean(pp))/sd(pp)*sqrt(3)/pi, y ~ rbinom(5000,1,p)

pt a continuous variable within 0 and 1, specifying the probability of p(z=1) given x1,x2,x3,x4,x5,x6

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

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 ~ rbinom(5000,1,0.2)

x2 binary variable, x2 ~ rbinom(5000,1,0.2)

x3 binary variable, x3 ~ rbinom(5000,1,0.8)

x4 binary variable, x4 ~ rbinom(5000,1,0.8)

x5 binary variable, x5 ~ rbinom(5000,1,0.7)

x6 binary variable, x6 ~ 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-mean(pp))/sd(pp)*sqrt(3)/pi, z ~ rbinom(5000,1,p)

y continuous variable indicating outcome, y ~ x1 + 6*x2+x6+2*z + rnorm(5000,0,1)

`target.binary.data` *A dataset of simulated RCT data, where outcome is binary variable. The data is filtered after compared to source.binary.data*

Description

A dataset of simulated RCT data, where outcome is binary variable. The data is filtered after compared to source.binary.data

Usage

```
target.binary.data
```

Format

A data frame with 3194 rows and 9 variables.

x1 binary variable, x1 ~ rbinom(5000,1,0.7)

x2 binary variable, x2 ~ rbinom(5000,1,0.9)

x3 binary variable, x3 ~ rbinom(5000,1,0.2)

x4 binary variable, x4 ~ rbinom(5000,1,0.3)

x5 binary variable, x5 ~ rbinom(5000,1,0.2)

x6 binary variable, x6 ~ rbinom(5000,1,0.3)

z binary variable. pp = x1*x2+x3*x4+5*x5+x6, p(z=1) = p = 1/1+exp^-(pp-mean(pp))/sd(pp)*sqrt(3)/pi, z ~ rbinom(5000,1,p)

y binary variable. pp = x1 + (6*x2+x6+2)*z, p(y=1) = p = 1/1+exp^-(pp-mean(pp))/sd(pp)*sqrt(3)/pi, y ~ rbinom(5000,1,p)

pt a continuous variable within 0 and 1, specifying the probability of p(z=1) given x1,x2,x3,x4,x5,x6

`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 ~ rbinom(5000,1,0.7)

x2 binary variable, x2 ~ rbinom(5000,1,0.9)

x3 binary variable, x3 ~ rbinom(5000,1,0.2)

x4 binary variable, x4 ~ rbinom(5000,1,0.3)

x5 binary variable, x5 ~ rbinom(5000,1,0.2)

x6 binary variable, x6 ~ rbinom(5000,1,0.3)

z binary variable indicating treatment and control, z ~ rbinom(5000,1,0.5)

y continuous variable indicating outcome, y ~ x1 + 6*x2+x6+2*z + rnorm(5000,0,1)

`TEstimator_wrapper` *Estimating conditional average treatment effects*

Description

Estimating conditional average treatment effects

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 effects. The allowed estimators are: `"G_computation"`, `"IPW"`, and `"DR"`.i The corresponding object wll be created by the function `TEstimator_wrapper()`. The default is `"G_computation"`, which, along with `outcome_method="glm"` models the potential outcomes.
`data`	A data frame containing variables named in `vars_name` and possible other variables.
`vars_name`	A list containing four character 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 name of outcome; `treatment_name` is a character vector of length one containing the 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 `http://topepo.github.io/caret/train-models-by-tag.html`. Default is "glm".
`treatment_method`	A character specifying a model for treatment. Possible values are found using `names(getModelInfo())`. See `http://topepo.github.io/caret/train-models-by-tag.html`. 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 https://topepo.github.io/caret/model-training-and-tuning.html for details.

Value

An object of class `TEstimator`.

Examples

```
data <- RCTrep::source.data[sample(dim(RCTrep::source.data)[1],500),]
vars_name <- list(confounders_treatment_name = c("x1","x2","x3","x4","x5","x6"),
                  treatment_name = c('z'),
                  outcome_name = c('y'))

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

Index