Package 'RCTrep'

December 15, 2021

Type Package

Title What the Package Does (Title Case)

Version 0.1.0

Author Who wrote it

Maintainer The package maintainer <yourself@somewhere.net>

Description More about what it does (maybe more than one line)
Use four spaces when indenting paragraphs within the Description.

License What license is it under?

Encoding UTF-8

LazyData true

Imports mvtnorm, MASS, MatchIt, ggplot2, ggpubr, dplyr, PSweight, numDeriv, R6

Suggests rmarkdown,
knitr,
testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.1.2

VignetteBuilder knitr

Depends R (>= 2.10)

R topics documented:

DR *R6 class: Doubly robust estimator class*

Description

A R6 class for doubly robust estimator that implements its own fit method.

Super classes

`RCTrep::Estimator` -> `RCTrep::DR_base` -> `DR`

Methods

Public methods:

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

Method `new()`:

Usage:

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

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.

DR_base *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::Estimator` -> `DR_base`

Methods

Public methods:

- DR_base$new()
- DR_base$clone()

Method new():

Usage:

```
DR_base$new(df, vars_name)
```

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Estimate *Estimate average treatment effect*

Description

The function Estimate is used to estimate the average treatment effect obtained from data.

Usage

```
Estimate(
  Estimator,
  data,
  vars_name,
  outcome_method = "glm",
  treatment_method = "glm",
  two_models = NULL,
  outcome_formula,
  treatment_formula,
  stratification,
  stratification_joint,
  strata_cut,
  ...
)
```

Arguments

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

vars_name
: A list containing four vectors confounders_internal, confounders_external, 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; confounders_external is a character vector containing variables for weighting as to generalize estimates from source.data to target.data; 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.

outcome_method
: A string specifying which model for outcome, treatment, and selection to use. Possible values are found using names(getModelInfo()). See http://topepo.github.io/caret/train-models-by-tag.html. A list of functions can also be passed for a custom model function. See http://topepo.github.io/caret/using-your-own-model-in-train.html for details.

treatment_method
: A string specifying which model for outcome, treatment, and selection to use. Possible values are found using names(getModelInfo()). See http://topepo.github.io/caret/train-models-by-tag.html. A list of functions can also be passed for a custom model function. See http://topepo.github.io/caret/using-your-own-model-in-train.html for details.

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

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

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

stratification
: An optional string vector containing variables to define subgroups. source.obj will compute both weighted and unweighted conditional average treatment effect, targe.obj will calculate the conditional average treatment effect based on these variables.

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

strata_cut
: An optional list containing lists. Each component is a list with tag named by a variable to discretize, containing break which is a vector specifying the interval of range of the variable to divide, lable which is a string 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 by modifying the public variable data of Estimator object in wrap function Estimate, and calculate the weight for generalization based on the discretized variables.

...
: An optional arguments 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.

Value

An object of class Estimator.

Estimator *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, inheritted by G_computation_base, IPW_base, and DR_base class.

Methods

Public methods:

- Estimator$new()
- Estimator$RCTrep()
- Estimator$get_CATE()
- Estimator$plot_CATE()
- Estimator$clone()

Method new(): Create a new Estimator object

Usage:

```
Estimator$new(df, vars_name)
```

Arguments:

df A data frame containing variables in vars_name

vars_name vars_name A list containing four vectors confounders_internal, confounders_external, 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; confounders_external is a character vector containing variables for weighting as to generalize estimates from source.data to target.data; 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 RCTrep(): 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_external_name while not in stratification.

Usage:

```
Estimator$RCTrep(
  target.obj,
  weighting_estimator,
  weighting_method,
  stratification,
  stratification_joint
)
```

Arguments:

`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.

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

Method `get_CATE()`**:** 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:

```
Estimator$get_CATE(stratification, stratification_joint)
```

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`.

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:

```
Estimator$plot_CATE(stratification, stratification_joint = FALSE)
```

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`.

Returns: A plot containing a forest plot and a table with numeric results.

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

G_computation *R6 class: G_computation class*

Description

A R6 class for G_computation estimator for average treatment effect

Super classes

RCTrep::Estimator -> RCTrep::G_computation_base -> G_computation

Methods

Public methods:

- G_computation$new()
- G_computation$clone()

Method new():

Usage:

```
G_computation$new(
  df,
  vars_name,
  gc.method,
  gc.formula,
  var_approach = "Bias_adjusted",
  ...
)
```

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

G_computation_base *R6 class: G_computation base class*

Description

A base R6 class for G_computation estimator for average treatment effect

Super class

`RCTrep::Estimator` -> `G_computation_base`

Methods

Public methods:

- `G_computation_base$new()`
- `G_computation_base$residual_check()`
- `G_computation_base$clone()`

Method `new()`:

Usage:

```
G_computation_base$new(df, vars_name)
```

Method `residual_check()`:

Usage:

```
G_computation_base$residual_check(stratification)
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

IPW *R6 class: IPW class*

Description

A R6 class for IPW estimator for average treatment effect that implements its own fit method.

Super classes

`RCTrep::Estimator` -> `RCTrep::IPW_base` -> `IPW`

Methods

Public methods:

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

Method `new()`:

Usage:

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

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.

`IPW_base` *R6 class: Inverse propensity score weighting estimator base class*

Description

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

Super class

`RCTrep::Estimator` -> `IPW_base`

Public fields

`model:` a fitted model for treatment conditioning on covariates

`method:` a string specifying the method for fitting the propensity score

`formula:` an optional `formula` object. The formula is specified as `Z ~ X1+X2...`, where `Z` represents treatment variable and `X1` and `X2` are covariates.

`ps:` a numeric vector of length n where n is row number of `self$data`.

Active bindings

`model:` a fitted model for treatment conditioning on covariates

`method:` a string specifying the method for fitting the propensity score

`formula:` an optional `formula` object. The formula is specified as `Z ~ X1+X2...`, where `Z` represents treatment variable and `X1` and `X2` are covariates.

`ps:` a numeric vector of length n where n is row number of `self$data`.

Methods

Public methods:

- IPW_base$new()
- IPW_base$clone()

Method new():

Usage:

```
IPW_base$new(df, vars_name)
```

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Plot_estimates *plot average treatment effect and possible conditional average treatment*

Description

plot average treatment effect and possible conditional average treatment

Usage

```
Plot_estimates(source.obj, target.obj)
```

Arguments

source.obj	An object of class Estimator resulting from source data
target.obj	An object of class Estimator resulting from target data

Value

a plot with a forest plot and a table with numeric results

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(
  Estimator = "G_computation",
  weighting_estimator = "Balancing",
  source.data = source.data,
  target.data = target.data,
  vars_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 = NULL,
  two_models = NULL,
  ...
)
```

Arguments

Estimator	A character specifying an estimator for average treatment effect. The allowed estimators for Estimator are: "G_computation", "IPW", and "DR". The corresponding object will be created by the wrapper function Estimate(). The default is "G_computation", which, along with outcome_method="glm" model the potential outcomes.
weighting_estimator	A character specifying a weighting estimator for generalizing/transporting the estimates of source.obj (initiated using RWD) to target.obj (initiated using RCT) as to enable replication/comparison between source study (RWD) and target study(RCT). The allowed estimators are: "balancing", and "modeling".

"balancing" estimator use exact matching to compute the weight; "modeling" estimator model the probability of being selected to the target RCT (assuming combing target RCT and the source RWD as a population). The default is "Balancing", which, implements the exact matching on variables in vars_name$external_confoun 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. If not found in source.data, the function will stop and throw error; vectors of binary treatment and binary outcome should be factor.

target.data A data frame containing variables named in vars_name and possible other variables, or a list of four components with four tags ATE_mean, ATE_se, CATE_mean_se, and univariate_p reference four components with data type numeric, numeric, data.frame, and list respectively. ATE_mean is a numeric vector of length 1 containing the point estimate of the treatment effect in target.data, ATE_se is a numeric vector of length 1 containing the standard error the treatment effect, CATE_mean_se is a data frame containing five vectors name, value, cate, se, and size of length N, where name is variables that divide the target population into smaller groups, value is levels of variables in name, cate is the provided conditional average treatment effect of a subgroup defined by a variable in name with the corresponding level in value, se is the standard error of the cate, size is the group size, N is the number of stratum based on variables in the vector name and levels in value. univariate_p is a list of length equal to the number of variables to divide the population with tags equal to variable names, each component containing a vector, which, each containing the name of a variable, number of levels of the variable, levels of the variable, and the distribution of each level by order.

vars_name A list containing four vectors confounders_internal, confounders_external, 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; confounders_external is a character vector containing variables for weighting as to generalize estimates from source.data to target.data; 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.

outcome_method, treatment_method, weighting_method
A string specifying which model for outcome, treatment, and selection to use. Possible values are found using names(getModelInfo()). See http://topepo.github.io/caret/train-models-by-tag.html. A list of functions can also be passed for a custom model function. See http://topepo.github.io/caret/using-your-own-model-in-train.html for details.

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

stratification An optional string vector containing variables to define subgroups. source.obj will compute both weighted and unweighted conditional average treatment effect, targe.obj will calculate the conditional average treatment effect based on these variables.

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

`strata_cut` An optional list containing lists. Each component is a list with tag named by a variable to discretize, containing `break` which is a vector specifying the interval of range of the variable to divide, `lable` which is a string 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` by modifying the public variable `data` of `Estimator` object in wrap function `Estimate`, and calculate the weight for generalization based on the discretized variables.

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

`...` An optional arguments 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 for both studies are constructed by a wrapper function `Estimate()` with user's input of data and estimator for treatment effect. Then `Estimate()` return initialized objects `source.obj` and `target.obj`. `source.obj` replicates the target RCT estimate via the class method `RCTrep()` with input of target object `target.obj` and weighting method.

Value

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

Examples

```
library(RCTrep)
source.data <- RCTrep::source.data
target.data <- RCT::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)))
## Not run: 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)
```

`source.data` *Dutch Colon cancer registry data*

Description

A dataset containing synthetic dutch colon cancer registry data and other biomarkers

Usage

```
source.data
```

Format

A data frame with 2000 rows and 9 variables

vitstat 1=death, 0=survival

Stage2 pathological stage, 1=stage2, 0=stage3

Index