Package ‘htetree’

November 29, 2023

Type Package

Title Causal Inference with Tree-Based Machine Learning Algorithms

Version 0.1.18

Description Estimating heterogeneous treatment effects with tree-based machine learning algorithms and visualizing estimated results in flexible and presentation-ready ways. For more information, see Brand, Xu, Koch, and Geraldo (2021) <doi:10.1177/0081175021993503>. Our current package first started as a fork of the 'causalTree' package on 'GitHub' and we greatly appreciate the authors for their extremely useful and free package.

Depends R (>= 3.6.0)

Imports Rcpp, grf, partykit, data.tree, Matching, dplyr, jsonlite, rpart, rpart.plot, shiny, stringr

Suggests optmatch, haven, foreign, data.table, remotes, party

License GPL-2 | GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation yes

Author Jiahui Xu [cre, aut],
Tanvi Shinkre [aut],
Jennie Brand [aut]

Maintainer Jiahui Xu <jiahuixu@ucla.edu>

Repository CRAN

Date/Publication 2023-11-29 18:50:05 UTC

R topics documented:

`bundScript` *Include the Javascript Used in Shiny*

Description

intermediate function used to include necessary javascript to visualize tree structures and estimated treatment effect in shiny

Usage

```
bundScript(...)
```

Arguments

`...`	There is no required arguments in this function. But user could manipulate to include different css files.

Value

No return value. It is used to pass the Javascript to Shiny.

---

causalTree *Causal Effect Regression and Estimation Trees*

Description

Fit a `causalTree` model to get an `rpart` object

Usage

```
causalTree(
  formula,
  data,
  weights,
  treatment,
  subset,
  na.action = na.causalTree,
  split.Rule,
  split.Honest,
  HonestSampleSize,
  split.Bucket,
  bucketNum = 5,
  bucketMax = 100,
  cv.option,
  cv.Honest,
  minsize = 2L,
  x = FALSE,
  y = TRUE,
  propensity,
  control,
  split.alpha = 0.5,
  cv.alpha = 0.5,
  cv.gamma = 0.5,
  split.gamma = 0.5,
  cost,
  ...
)
```

Arguments

- `formula` — a formula, with a response and features but no interaction terms. If this a a data frome, that is taken as the model frame (see `model.frame`).
- `data` — an optional data frame that includes the variables named in the formula.
- `weights` — optional case weights.
- `treatment` — a vector that indicates the treatment status of each observation. 1 represents treated and 0 represents control. Only binary treatment supported in this version.
- `subset` — optional expression saying that only a subset of the rows of the data should be used in the fit.
- `na.action` — the default action deletes all observations for which `y` is missing, but keeps those in which one or more predictors are missing.
- `split.Rule` — causalTree splitting options, one of `"TOT"`, `"CT"`, `"fit"`, `"tstats"`, four splitting rules in `causalTree`. Note that the `"tstats"` alternative does not have an associated cross-validation method `cv.option`; see Athey and Imbens (2016) for a discussion. Note further that `split.Rule` and `cv.option` can mix and match.
- `split.Honest` — boolean option, `TRUE` or `FALSE`, used for `split.Rule` as `"CT"` or `"fit"`. If set as `TRUE`, do honest splitting, with default `split.alpha` = 0.5; if set as `FALSE`, do adaptive splitting with `split.alpha` = 1. The user choice of `split.alpha` will be ignored if `split.Honest` is set as `FALSE`, but will be respected if set to `TRUE`. For `split.Rule="TOT"`, there is no honest splitting option and the parameter `split.alpha` does not matter. For `split.Rule="tstats"`, a value of `TRUE` enables use of `split.alpha` in calculating the risk function, which determines the order of pruning in cross-validation. Note also that causalTree function returns the estimates from the training data, no matter what the value of `split.Honest` is; the tree must be re-estimated to get the honest estimates using `estimate.causalTree`. The wrapper function `honest.CausalTree` does honest estimation in one step and returns a tree.
- `HonestSampleSize` — number of observations anticipated to be used in honest re-estimation after building the tree. This enters the risk function used in both splitting and cross-validation.
- `split.Bucket` — boolean option, `TRUE` or `FALSE`, used to specify whether to apply the discrete method in splitting the tree. If set as `TRUE`, in splitting a node, the observations in a leaf will be be partitioned into buckets, with each bucket containing `bucketNum` treated and `bucketNum` control units, and where observations are ordered prior to partitioning. Splitting will take place by bucket.
- `bucketNum` — number of observations in each bucket when set `split.Bucket` = `TRUE`. However, the code will override this choice in order to guarantee that there are at least `minsize` and at most `bucketMax` buckets.
- `bucketMax` — Option to choose maximum number of buckets to use in splitting when set `split.Bucket` = `TRUE`, `bucketNum` can change by choice of `bucketMax`.
- `cv.option` — cross validation options, one of `"TOT"`, `"matching"`, `"CT"`, `"fit"`, four cross validation methods in **causalTree**. There is no `cv.option` for the `split.Rule` `"tstats"`; see Athey and Imbens (2016) for discussion.

`cv.Honest` : boolean option, `TRUE` or `FALSE`, only used for `cv.option` as `"CT"` or `"fit"`, to specify whether to apply honest risk evalation function in cross validation. If set `TRUE`, use honest risk function, otherwise use adaptive risk function in cross validation. If set `FALSE`, the user choice of `cv.alpha` will be set to 1. If set `TRUE`, `cv.alpha` will default to 0.5, but the user choice of `cv.alpha` will be respected. Note that honest cv estimates within-leaf variances and may perform better with larger leaf sizes and/or small number of cross-validation sets.

`minsize` : in order to split, each leaf must have at least `minsize` treated cases and `minsize` control cases. The default value is set as 2.

`x` : keep a copy of the `x` matrix in the result.

`y` : keep a copy of the dependent variable in the result. If missing and `model` is supplied this defaults to `FALSE`.

`propensity` : propensity score used in `"TOT"` splitting and `"TOT"`, honest `"CT"` cross validation methods. The default value is the proportion of treated cases in all observations. In this implementation, the propensity score is a constant for the whole dataset. Unit-specific propensity scores are not supported; however, the user may use inverse propensity scores as case weights if desired.

`control` : a list of options that control details of the `rpart` algorithm. See `rpart.control`.

`split.alpha` : scale parameter between 0 and 1, used in splitting risk evaluation function for `"CT"`. When `split.Honest = FALSE`, `split.alpha` will be set as 1. For `split.Rule="tstats"`, if `split.Honest=TRUE`, `split.alpha` is used in calculating the risk function, which determines the order of pruning in cross-validation.

`cv.alpha` : scale paramter between 0 and 1, used in cross validation risk evaluation function for `"CT"` and `"fit"`. When `cv.Honest = FALSE`, `cv.alpha` will be set as 1.

`cv.gamma, split.gamma` : optional parameters used in evaluating policies.

`cost` : a vector of non-negative costs, one for each variable in the model. Defaults to one for all variables. These are scalings to be applied when considering splits, so the improvement on splitting on a variable is divided by its cost in deciding which split to choose.

`...` : arguments to `rpart.control` may also be specified in the call to `causalTree`. They are checked against the list of valid arguments. An example of a commonly set parameter would be `xval`, which sets the number of cross-validation samples. The parameter `minsize` is implemented differently in `causalTree` than in `rpart`; we require a minimum of `minsize` treated observations and a minimum of `minsize` control observations in each leaf.

Details

CausalTree differs from `rpart` function from **rpart** package in splitting rules and cross validation methods. Please check Athey and Imbens, *Recursive Partitioning for Heterogeneous Causal Effects* (2016) for more details.

Value

An object of class `rpart`. See `rpart.object`.

References

Breiman L., Friedman J. H., Olshen R. A., and Stone, C. J. (1984) *Classification and Regression Trees.* Wadsworth.

Athey, S and G Imbens (2016) *Recursive Partitioning for Heterogeneous Causal Effects*. http://arxiv.org/abs/1504.01132

See Also

honest.causalTree, rpart.control, rpart.object, summary.rpart, rpart.plot

Examples

```
library("htetree")
library("rpart")
library("rpart.plot")
tree <- causalTree(y~ x1 + x2 + x3 + x4, data = simulation.1,
treatment = simulation.1$treatment,
split.Rule = "CT", cv.option = "CT", split.Honest = TRUE, cv.Honest = TRUE,
split.Bucket = FALSE, xval = 5,
cp = 0, minsize = 20, propensity = 0.5)

opcp <- tree$cptable[,1][which.min(tree$cptable[,4])]

opfit <- prune(tree, opcp)

rpart.plot(opfit)

fittree <- causalTree(y~ x1 + x2 + x3 + x4, data = simulation.1,
                      treatment = simulation.1$treatment,
                      split.Rule = "fit", cv.option = "fit",
                      split.Honest = TRUE, cv.Honest = TRUE, split.Bucket = TRUE,
                      bucketNum = 5,
                      bucketMax = 200, xval = 10,
                      cp = 0, minsize = 20, propensity = 0.5)

tstatstree <- causalTree(y~ x1 + x2 + x3 + x4, data = simulation.1,
                         treatment = simulation.1$treatment,
                         split.Rule = "tstats", cv.option = "CT",
                         cv.Honest = TRUE, split.Bucket = TRUE,
                         bucketNum = 10,
                         bucketMax = 200, xval = 5,
                         cp = 0, minsize = 20, propensity = 0.5)
```

---

causalTree.branch *Compute the "branches" to be drawn for an* causalTree *object*

Description

Compute the "branches" to be drawn for an causalTree object

Usage

```
causalTree.branch(x, y, node, branch)
```

Arguments

x	covariates
y	outcome
node	node of the fitted tree
branch	branch of the fitted tree

Value

number of branches to be drawn

---

causalTree.control *Intermediate function for* causalTree

Description

Intermediate function for causalTree

Usage

```
causalTree.control(
  minsplit = 20L,
  minbucket = round(minsplit/3),
  cp = 0,
  maxcompete = 4L,
  maxsurrogate = 5L,
  usesurrogate = 2L,
  xval = 10L,
  surrogatestyle = 0L,
  maxdepth = 30L,
  ...
)
```

Arguments

minsplit	minimum number of splits
minbucket	minimum number of bucket
cp	default is 0
maxcompete	maximum number of compete
maxsurrogate	maximum number of surrogate
usesurrogate	initial number of surrogate

xval	cross-validation
surrogatestyle	the style of surrogate
maxdepth	Maximum depth
...	arguments to rpart.control may also be specified in the call to causalTree. They are checked against the list of valid arguments. An example of a commonly set parameter would be xval, which sets the number of cross-validation samples. The parameter minsize is implemented differently in causalTree than in rpart; we require a minimum of minsize treated observations and a minimum of minsize control observations in each leaf.

Value

parameters used to in causalTree

---

causalTree.matrix *Intermediate function for* causalTree

Description

Intermediate function for causalTree

Usage

```
causalTree.matrix(frame)
```

Arguments

frame	inherited from data.frame

Value

A covariate matrix used in the causal regression.

---

causalTreecallback *Intermediate function for* causalTree

Description

This routine sets up the callback code for user-written split routines in causalTree

Usage

```
causalTreecallback(mlist, nobs, init)
```

Arguments

`mlist`	a list of user written methods
`nobs`	number of observations
`init`	function name

Value

split method written by users

`causalTreeco` *Intermediate function for* `causalTree`

Description

Compute the x-y coordinates for a tree

Usage

```
causalTreeco(tree, parms)
```

Arguments

`tree`	an `causalTree` object
`parms`	parms

Value

the x-y coordinates for a tree

`clearTemp` *Clear Temporary Files*

Description

The files for shiny are saved in a temporary directory. The files can be cleared manually using the ‘clearTemp()‘ function, or will automatically be cleared when you close R

Usage

```
clearTemp()
```

Value

no return value, to unlink files under the temp folder

est.causalTree — *Intermediate function for* causalTree

Description

Run down the built tree and get the final leaf ids for estimation sample

Usage

```
est.causalTree(fit, x)
```

Arguments

fit	an causalTree object
x	covariates

Value

Intermediate estimation results for an causalTree object.

estimate.causalTree — *estimate causal Tree*

Description

estimate causal Tree

Usage

```
estimate.causalTree(
  object,
  data,
  weights,
  treatment,
  na.action = na.causalTree
)
```

Arguments

object	A tree-structured fit rpart object, such as one generated as a causalTree fit.
data	New data frame to be used for estimating effects within leaves.
weights	optional case weights.
treatment	The treatment status of observations in the new dataframe, where 1 represents treated and 0 represents control.
na.action	the default action deletes all observations for which y is missing, but keeps those in which one or more predictors are missing.

Details

When the leaf contains only treated or control cases, the function will trace back to the leaf's parent node recursively until the parent can be used to compute causal effect. Please see Athey and Imbens *Machine Learning Methods for Estimating Heterogeneous Causal Effects* (2015) for details.

Value

Intermediate estimation results for an `causalTree` object

`formatg` *Intermediate function for* `causalTree`

Description

Intermediate function for `causalTree`

Usage

```
formatg(x, digits = getOption("digits"), format = paste0("%.", digits, "g"))
```

Arguments

`x`	input training data
`digits`	number of digits to be kept
`format`	format of exported vector

Value

No return value, called for formatting the exported estimates

---

`getDefaultPath` *Get the Current Working Directory*

Description

get the current work directory and set it as the default directory to save the shiny files temporarily

Usage

```
getDefaultPath()
```

Value

a temporary file path

getDensities *Getting Distribution in Treatment and Control Groups*

Description

Getting the density of distribution in treatment and control groups, which will be displayed in the

Usage

```
getDensities(treatment, outcome)
```

Arguments

`treatment`	A character representing the name of treatment indicator.
`outcome`	A character representing the name of outcome variable.

Value

vector of corresponding densities for each value of outcome vector

---

honest.causalTree *Causal Effect Regression and Estimation Trees: One-step honest estimation*

Description

Fit a `causalTree` model to get an honest causal tree, with tree structure built on training sample (including cross-validation) and leaf estimates taken from estimation sample. Return an `rpart` object.

Usage

```
honest.causalTree(
  formula,
  data,
  weights,
  treatment,
  subset,
  est_data,
  est_weights,
  est_treatment,
  est_subset,
  na.action = na.causalTree,
  split.Rule,
  split.Honest,
```

```
    HonestSampleSize,
    split.Bucket,
    bucketNum = 10,
    bucketMax = 40,
    cv.option,
    cv.Honest,
    minsize = 2L,
    model = FALSE,
    x = FALSE,
    y = TRUE,
    propensity,
    control,
    split.alpha = 0.5,
    cv.alpha = 0.5,
    cv.gamma = 0.5,
    split.gamma = 0.5,
    cost,
    ...
)
```

Arguments

`formula`	a formula, with a response and features but no interaction terms. If this a a data frome, that is taken as the model frame (see `model.frame`).
`data`	an optional data frame that includes the variables named in the formula.
`weights`	optional case weights.
`treatment`	a vector that indicates the treatment status of each observation. 1 represents treated and 0 represents control. Only binary treatment supported in this version.
`subset`	optional expression saying that only a subset of the rows of the data should be used in the fit.
`est_data`	data frame to be used for leaf estimates; the estimation sample. Must contain the variables used in training the tree.
`est_weights`	optional case weights for estimation sample
`est_treatment`	treatment vector for estimation sample. Must be same length as estimation data. A vector indicates the treatment status of the data, 1 represents treated and 0 represents control. Only binary treatment supported in this version.
`est_subset`	optional expression saying that only a subset of the rows of the estimation data should be used in the fit of the re-estimated tree.
`na.action`	the default action deletes all observations for which `y` is missing, but keeps those in which one or more predictors are missing.
`split.Rule`	causalTree splitting options, one of `"TOT"`, `"CT"`, `"fit"`, `"tstats"`, four splitting rules in `causalTree`. Note that the `"tstats"` alternative does not have an associated cross-validation method `cv.option`; see Athey and Imbens (2016) for a discussion. Note further that `split.Rule` and `cv.option` can mix and match.

`split.Honest` boolean option, `TRUE` or `FALSE`, used for `split.Rule` as `"CT"` or `"fit"`. If set as `TRUE`, do honest splitting, with default `split.alpha` = 0.5; if set as `FALSE`, do adaptive splitting with `split.alpha` = 1. The user choice of `split.alpha` will be ignored if `split.Honest` is set as `FALSE`, but will be respected if set to `TRUE`. For `split.Rule="TOT"`, there is no honest splitting option and the parameter `split.alpha` does not matter. For `split.Rule="tstats"`, a value of `TRUE` enables use of `split.alpha` in calculating the risk function, which determines the order of pruning in cross-validation. Note also that causalTree function returns the estimates from the training data, no matter what the value of `split.Honest` is; the tree must be re-estimated to get the honest estimates using `estimate.causalTree`. The wrapper function `honest.CausalTree` does honest estimation in one step and returns a tree.

`HonestSampleSize` number of observations anticipated to be used in honest re-estimation after building the tree. This enters the risk function used in both splitting and cross-validation.

`split.Bucket` boolean option, `TRUE` or `FALSE`, used to specify whether to apply the discrete method in splitting the tree. If set as `TRUE`, in splitting a node, the observations in a leaf will be be partitioned into buckets, with each bucket containing `bucketNum` treated and `bucketNum` control units, and where observations are ordered prior to partitioning. Splitting will take place by bucket.

`bucketNum` number of observations in each bucket when set `split.Bucket = TRUE`. However, the code will override this choice in order to guarantee that there are at least `minsize` and at most `bucketMax` buckets.

`bucketMax` Option to choose maximum number of buckets to use in splitting when set `split.Bucket = TRUE`, `bucketNum` can change by choice of `bucketMax`.

`cv.option` cross validation options, one of `"TOT"`, `"matching"`, `"CT"`, `"fit"`, four cross validation methods in **causalTree**. There is no `cv.option` for the `split.Rule` `"tstats"`; see Athey and Imbens (2016) for discussion.

`cv.Honest` boolean option, `TRUE` or `FALSE`, only used for `cv.option` as `"CT"` or `"fit"`, to specify whether to apply honest risk evalation function in cross validation. If set `TRUE`, use honest risk function, otherwise use adaptive risk function in cross validation. If set `FALSE`, the user choice of `cv.alpha` will be set to 1. If set `TRUE`, `cv.alpha` will default to 0.5, but the user choice of `cv.alpha` will be respected. Note that honest cv estimates within-leaf variances and may perform better with larger leaf sizes and/or small number of cross-validation sets.

`minsize` in order to split, each leaf must have at least `minsize` treated cases and `minsize` control cases. The default value is set as 2.

`model` model frame of `causalTree`, same as `rpart`

`x` keep a copy of the `x` matrix in the result.

`y` keep a copy of the dependent variable in the result. If missing and `model` is supplied this defaults to `FALSE`.

`propensity` propensity score used in `"TOT"` splitting and `"TOT"`, honest `"CT"` cross validation methods. The default value is the proportion of treated cases in all observations. In this implementation, the propensity score is a constant for the whole dataset.

Unit-specific propensity scores are not supported; however, the user may use inverse propensity scores as case weights if desired.

control
: a list of options that control details of the rpart algorithm. See rpart.control.

split.alpha
: scale parameter between 0 and 1, used in splitting risk evaluation function for "CT". When split.Honest = FALSE, split.alpha will be set as 1. For split.Rule="tstats", if split.Honest=TRUE, split.alpha is used in calculating the risk function, which determines the order of pruning in cross-validation.

cv.alpha
: scale paramter between 0 and 1, used in cross validation risk evaluation function for "CT" and "fit". When cv.Honest = FALSE, cv.alpha will be set as 1.

cv.gamma, split.gamma
: optional parameters used in evaluating policies.

cost
: a vector of non-negative costs, one for each variable in the model. Defaults to one for all variables. These are scalings to be applied when considering splits, so the improvement on splitting on a variable is divided by its cost in deciding which split to choose.

...
: arguments to rpart.control may also be specified in the call to causalTree. They are checked against the list of valid arguments. An example of a commonly set parameter would be xval, which sets the number of cross-validation samples. The parameter minsize is implemented differently in causalTree than in rpart; we require a minimum of minsize treated observations and a minimum of minsize control observations in each leaf.

Value

An object of class rpart. See rpart.object.

References

Breiman L., Friedman J. H., Olshen R. A., and Stone, C. J. (1984) *Classification and Regression Trees.* Wadsworth.

Athey, S and G Imbens (2016) *Recursive Partitioning for Heterogeneous Causal Effects*. http://arxiv.org/abs/1504.01132

See Also

causalTree, estimate.causalTree, rpart.object, summary.rpart, rpart.plot

Examples

```
library("rpart")
library("rpart.plot")
library("htetree")
n <- nrow(simulation.1)

trIdx <- which(simulation.1$treatment == 1)

conIdx <- which(simulation.1$treatment == 0)

train_idx <- c(sample(trIdx, length(trIdx) / 2), sample(conIdx,
```

```
    length(conIdx) / 2))

train_data <- simulation.1[train_idx, ]

est_data <- simulation.1[-train_idx, ]

honestTree <- honest.causalTree(y ~ x1 + x2 + x3 + x4, data = train_data,
     treatment = train_data$treatment,
     est_data = est_data,
     est_treatment = est_data$treatment,
     split.Rule = "CT", split.Honest = TRUE,
     HonestSampleSize = nrow(est_data),
     split.Bucket = TRUE, cv.option = "CT")

opcp <-  honestTree$cptable[,1][which.min(honestTree$cptable[,4])]

opTree <- prune(honestTree, opcp)

rpart.plot(opTree)
```

---

honest.est.causalTree *honest re-estimation and change the frame of object using estimation sample*

Description

honest re-estimation and change the frame of object using estimation sample

Usage

```
honest.est.causalTree(fit, x, wt, treatment, y)
```

Arguments

fit	an causalTree object
x	input training data
wt	optional weights
treatment	treatment variable
y	outcome variable

Value

An object of class rpart. See rpart.object.

`honest.est.rparttree` *honest re-estimation and change the frame of object using estimation sample*

Description

honest re-estimation and change the frame of object using estimation sample

Usage

```
honest.est.rparttree(fit, x, wt, y)
```

Arguments

`fit`	an `causalTree` object
`x`	input training data
`wt`	optional weights
`y`	outcome variable

Value

Intermediate estimation results for an honest estimation of `causalTree`.

`honest.rparttree` *Honest recursive partitioning Tree*

Description

The recursive partitioning function, for R

Usage

```
honest.rparttree(
  formula,
  data,
  weights,
  subset,
  est_data,
  est_weights,
  na.action = na.rpart,
  method,
  model = FALSE,
  x = FALSE,
  y = TRUE,
  parms,
```

```
  control,
  cost,
  ...
)
```

Arguments

formula	a formula, with a response and features but no interaction terms. If this a a data frome, that is taken as the model frame (see model.frame).
data	an optional data frame that includes the variables named in the formula.
weights	optional case weights.
subset	optional expression saying that only a subset of the rows of the data should be used in the fit.
est_data	data frame to be used for leaf estimates; the estimation sample. Must contain the variables used in training the tree.
est_weights	optional case weights for estimation sample
na.action	the default action deletes all observations for which y is missing, but keeps those in which one or more predictors are missing.
method	one of "anova", "poisson", "class" or "exp". If method is missing then the routine tries to make an intelligent guess. If y is a survival object, then method = "exp" is assumed, if y has 2 columns then method = "poisson" is assumed, if y is a factor then method = "class" is assumed, otherwise method = "anova" is assumed. It is wisest to specify the method directly, especially as more criteria may added to the function in future. Alternatively, method can be a list of functions named init, split and eval. Examples are given in the file 'tests/usersplits.R' in the sources, and in the vignettes 'User Written Split Functions'.
model	model frame of causalTree, same as rpart
x	keep a copy of the x matrix in the result.
y	keep a copy of the dependent variable in the result. If missing and model is supplied this defaults to FALSE.
parms	optional parameters for the splitting function. Anova splitting has no parameters. Poisson splitting has a single parameter, the coefficient of variation of the prior distribution on the rates. The default value is 1. Exponential splitting has the same parameter as Poisson. For classification splitting, the list can contain any of: the vector of prior probabilities (component prior), the loss matrix (component loss) or the splitting index (component split). The priors must be positive and sum to 1. The loss matrix must have zeros on the diagonal and positive off-diagonal elements. The splitting index can be gini or information. The default priors are proportional to the data counts, the losses default to 1, and the split defaults to gini.
control	a list of options that control details of the rpart algorithm. See rpart.control.

`cost`	a vector of non-negative costs, one for each variable in the model. Defaults to one for all variables. These are scalings to be applied when considering splits, so the improvement on splitting on a variable is divided by its cost in deciding which split to choose.
`...`	arguments to `rpart.control` may also be specified in the call to `causalTree`. They are checked against the list of valid arguments. An example of a commonly set parameter would be `xval`, which sets the number of cross-validation samples. The parameter `minsize` is implemented differently in `causalTree` than in `rpart`; we require a minimum of `minsize` treated observations and a minimum of `minsize` control observations in each leaf.

Value

An object of class `rpart` after running an honest recursive partitioning tree. .

`htetree.anova` *Intermediate function for* `causalTree`

Description

Intermediate function for `causalTree`

Usage

```
htetree.anova(y, offset, wt)
```

Arguments

`y`	outcome variable
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.
`wt`	optional weights

Value

No return value.

hte_causalTree *Estimate Heterogeneous Treatment Effect via Causal Tree*

Description

Estimate heterogeneous treatment effect via causal tree. In each leaf, the treatment effect is the difference of mean outcome in treatment group and control group.

Usage

```
hte_causalTree(
  outcomevariable,
  minsize = 20,
  crossvalidation = 20,
  data,
  treatment_indicator,
  ps_indicator,
  covariates,
  negative = FALSE,
  drawplot = TRUE,
  varlabel = NULL,
  maintitle = "Heterogeneous Treatment Effect Estimation",
  legend.x = 0.08,
  legend.y = 0.25,
  check = FALSE,
  ...
)
```

Arguments

`outcomevariable`
: a character representing the column name of the outcome variable.

`minsize`
: the minimum number of observations in each leaf. The default is set as 20.

`crossvalidation`
: number of cross validations. The default is set as 20.

`data`
: a data frame containing the variables in the model.

`treatment_indicator`
: a character representing the column name of the treatment indicator.

`ps_indicator`
: a character representing the column name of the propensity score.

`covariates`
: a vector of column names of all covariates (linear terms andpropensity score).

`negative`
: a logical value indicating whether we expect the treatment effect to be negative. The default is set as FALSE.

`drawplot`
: a logical value indicating whether to plot the model as part of the output. The default is set as TRUE.

`varlabel`
: a named vector containing variable labels.

maintitle	a character string indicating the main title displayed when plotting the tree and results. The default is set as "Heterogeneous Treatment Effect Estimation".
legend.x, legend.y	x and y coordinate to position the legend. The default is set as (0.08, 0.25).
check	if TRUE, generates 100 trees and outputs most common tree structures and their frequency
...	further arguments passed to or from other methods.

Value

predicted treatment effect and the associated tree

Examples

```
library(rpart)
library(htetree)
hte_causalTree(outcomevariable="outcome",
    data=data.frame("confounder"=c(0, 1, 1, 0, 1, 1),
                    "treatment"=c(0,0,0,1,1,1),
                    "prop_score"=c(0.4, 0.4, 0.5, 0.6, 0.6, 0.7),
                    "outcome"=c(1, 2, 2, 1, 4, 4)),
   treatment_indicator = "treatment",
   ps_indicator = "prop_score",
   covariates = "confounder")
```

---

hte_forest *Estimate Heterogeneous Treatment Effect via Random Forest*

Description

Estimate heterogeneous treatment effect via random forest. In each leaf, the treatment effect is the difference of mean outcome weighted by inverse propensity scores in treatment group and control group.

Usage

```
hte_forest(
  outcomevariable,
  minsize = 20,
  crossvalidation = 20,
  data = edurose_mediation_20181126,
  treatment_indicator = "compcoll25",
  ps_indicator = "propsc_com25",
  ps_linear = "propsc_com25lin",
  covariates = c(linear_terms, ps_indicator),
  negative = FALSE,
  drawplot = TRUE,
```

```
  legend.x = 0.08,
  legend.y = 0.25,
  gf,
  ...
)
```

Arguments

outcomevariable
: a character representing the column name of the outcome variable.

minsize
: the minimum number of observations in each leaf. The default is set as 20.

crossvalidation
: number of cross validations. The default is set as 20.

data
: a data frame containing the variables in the model.

treatment_indicator
: a character representing the column name of the treatment indicator.

ps_indicator
: a character representing the column name of the propensity score.

ps_linear
: a character representing name of a column that stores linearized propensity scores.

covariates
: a vector of column names of all covariates (linear terms andpropensity score).

negative
: a logical value indicating whether we expect the treatment effect to be negative. The default is set as FALSE.

drawplot
: a logical value indicating whether to plot the model as part of the output. The default is set as TRUE.

legend.x, legend.y
: x and y coordinate to position the legend. The default is set as (0.08, 0.25).

gf
: a fitted generalized random forest object

...
: further arguments passed to or from other methods.

Value

A list with three elements. The first one is the predicted outcome for each unit. The second is an `causalTree` object with the tree split information. The third is a `data.frame` summarizing the prediction results.

`hte_ipw` *Estimate Heterogeneous Treatment Effect via Adjusted Causal Tree*

Description

Estimate heterogeneous treatment effect via adjusted causal tree. In each leaf, the treatment effect is the difference of mean outcome weighted by inverse propensity scores in treatment group and control group.

Usage

```
hte_ipw(
  outcomevariable,
  minsize = 20,
  crossvalidation = 20,
  data,
  treatment_indicator,
  ps_indicator,
  ps_linear = NULL,
  covariates,
  negative = FALSE,
  drawplot = TRUE,
  varlabel = NULL,
  maintitle = "Heterogeneous Treatment Effect Estimation",
  legend.x = 0.08,
  legend.y = 0.25,
  check = FALSE,
  ...
)
```

Arguments

`outcomevariable`
: a character representing the column name of the outcome variable.

`minsize`
: the minimum number of observations in each leaf. The default is set as 20.

`crossvalidation`
: number of cross validations. The default is set as 20.

`data`
: a data frame containing the variables in the model.

`treatment_indicator`
: a character representing the column name of the treatment indicator.

`ps_indicator`
: a character representing the column name of the propensity score.

`ps_linear`
: a character representing name of a column that stores linearized propensity scores.

`covariates`
: a vector of column names of all covariates (linear terms andpropensity score).

`negative`
: a logical value indicating whether we expect the treatment effect to be negative. The default is set as FALSE.

`drawplot`
: a logical value indicating whether to plot the model as part of the output. The default is set as TRUE.

`varlabel`
: a named vector containing variable labels.

`maintitle`
: a character string indicating the main title displayed when plotting the tree and results. The default is set as "Heterogeneous Treatment Effect Estimation".

`legend.x, legend.y`
: x and y coordinate to position the legend. The default is set as (0.08, 0.25).

`check`
: if TRUE, generates 100 trees and outputs most common tree structures and their frequency

`...`
: further arguments passed to or from other methods.

Value

predicted treatment effect and the associated tree

Examples

```
library(rpart)
library(htetree)
hte_ipw(outcomevariable="outcome",
data=data.frame("confounder"=c(0, 1, 1, 0, 1, 1),
"treatment"=c(0,0,0,1,1,1), "prop_score"=c(0.4, 0.4, 0.5, 0.6, 0.6, 0.7),
"outcome"=c(1, 2, 2, 1, 4, 4)), treatment_indicator = "treatment",
ps_indicator = "prop_score", covariates = "confounder")
```

---

hte_match *Estimate Heterogeneous Treatment Effect via Adjusted Causal Tree*

Description

Estimate heterogeneous treatment effect via adjusted causal tree. In each leaf, the treatment effect estimated from nn matching.

Usage

```
hte_match(
  outcomevariable,
  minsize = 20,
  crossvalidation = 20,
  data,
  treatment_indicator,
  ps_indicator,
  ps_linear = NULL,
  covariates,
  negative = FALSE,
  drawplot = TRUE,
  con.num = 1,
  varlabel = NULL,
  maintitle = "Heterogeneous Treatment Effect Estimation",
  legend.x = 0.08,
  legend.y = 0.25,
  check = FALSE,
  ...
)
```

Arguments

`outcomevariable`
: a character representing the column name of the outcome variable.

`minsize` the minimum number of observations in each leaf. The default is set as 20.

`crossvalidation`
number of cross validations. The default is set as 20.

`data` a data frame containing the variables in the model.

`treatment_indicator`
a character representing the column name of the treatment indicator.

`ps_indicator` a character representing the column name of the propensity score.

`ps_linear` a character representing name of a column that stores linearized propensity scores.

`covariates` a vector of column names of all covariates (linear terms andpropensity score).

`negative` a logical value indicating whether we expect the treatment effect to be negative. The default is set as FALSE.

`drawplot` a logical value indicating whether to plot the model as part of the output. The default is set as TRUE.

`con.num` a number indicating the number of units from control groups to be used in matching.

`varlabel` a named vector containing variable labels.

`maintitle` a character string indicating the main title displayed when plotting the tree and results. The default is set as "Heterogeneous Treatment Effect Estimation".

`legend.x, legend.y`
x and y coordinate to position the legend. The default is set as (0.08, 0.25).

`check` if TRUE, generates 100 trees and outputs most common tree structures and their frequency

`...` further arguments passed to or from other methods.

Value

predicted treatment effect and the associated tree

Examples

```
library(rpart)
library(htetree)
hte_match(outcomevariable="outcome",
data=data.frame("x1"=c(0, 1, 1, 0, 1, 1),"x2"=c(3, 2, 1, 5, 7, 1),
"treatment"=c(0,0,0,1,1,1), "prop_score"=c(0.4, 0.4, 0.5, 0.6, 0.6, 0.7),
"outcome"=c(1, 2, 2, 1, 4, 4)), treatment_indicator = "treatment",
ps_indicator = "prop_score", covariates = c("x1","x2"))
```

hte_plot *Visualize the Estimated Results*

Description

The function hte_plot takes a model created by causal tree, as well as the adjusted version, and plots the distribution of the outcome variable in treated and control groups in each leaf of the tree. This visualization aims to show how the predicted treatment effect changes with each split in the tree.

Usage

```
hte_plot(
  model,
  data,
  treatment_indicator = NULL,
  outcomevariable,
  propensity_score,
  plot.title = "Visualization of the Tree"
)
```

Arguments

model	a tree model constructed by hte_causalTree, hte_matchinleaves,or hte_ipw.
data	a data frame containing the variables in the model.
treatment_indicator	a character representing the column name for the treatment variable in the causal setup.
outcomevariable	a character representing the column name of the outcome variable.
propensity_score	a character representing the column name of the propensity score.
plot.title	character representing the main title of the plot.

Value

no return value

hte_plot_line *Visualize the Estimated Results*

Description

The function `hte_plot_line` takes a model created by causal tree, as well as the adjusted version, and plots the different least squares models used to estimate heterogeneous treatment effects(HTE) at each node. At each node, this visualization aims to show how the estimated treatment effect differs when using ordinary least squares and weighted least squares methods. The weighted least squares method in this package uses inverse propensity scores as weights, in order to reduce bias due to confounding variables.

Usage

```
hte_plot_line(
  model,
  data,
  treatment_indicator = NULL,
  outcomevariable,
  propensity_score,
  plot.title = "Visualization of the Tree",
  gamma = 0,
  lambda = 0,
  ...
)
```

Arguments

`model`	a tree model constructed by `hte_causalTree`, `hte_matchinleaves`,or `hte_ipw`.
`data`	a data frame containing the variables in the model.
`treatment_indicator`	a character representing the column name for the treatment variable in the causal setup.
`outcomevariable`	a character representing the column name of the outcome variable.
`propensity_score`	a character representing the column name of the propensity score.
`plot.title`	character representing the main title of the plot.
`gamma, lambda`	numbers indicating the bias level used in sensitivity analysis
`...`	further arguments passed to or from other methods.

Value

No return value, used for plotting the estimated results with lines.

importance — *Caclulate variable importance*

Description

Each primary split is credited with the value of splits$improve Each surrogate split gets split$adj times the primary split's value

Usage

```
importance(fit)
```

Arguments

`fit`	a fitted `causalTree` object.

Value

same as the `importance` function in **rpart**.

init.causalForest — *Causal Effect Regression and Estimation Forests (Tree Ensembles)*

Description

Build a random causal forest by fitting a user selected number of `causalTree` models to get an ensemble of `rpart` objects.

Usage

```
init.causalForest(
  formula,
  data,
  treatment,
  weights = FALSE,
  cost = FALSE,
  num.trees,
  ncov_sample
)

## S3 method for class 'causalForest'
predict(object, newdata, predict.all = FALSE, type = "vector", ...)

causalForest(
  formula,
  data,
```

```
    treatment,
    na.action = na.causalTree,
    split.Rule = "CT",
    double.Sample = TRUE,
    split.Honest = TRUE,
    split.Bucket = FALSE,
    bucketNum = 5,
    bucketMax = 100,
    cv.option = "CT",
    cv.Honest = TRUE,
    minsize = 2L,
    propensity,
    control,
    split.alpha = 0.5,
    cv.alpha = 0.5,
    sample.size.total = floor(nrow(data)/10),
    sample.size.train.frac = 0.5,
    mtry = ceiling(ncol(data)/3),
    nodesize = 1,
    num.trees = nrow(data),
    cost = FALSE,
    weights = FALSE,
    ncolx,
    ncov_sample
  )
```

Arguments

`formula`	a formula, with a response and features but no interaction terms. If this a a data frome, that is taken as the model frame (see `model.frame`).
`data`	an optional data frame that includes the variables named in the formula.
`treatment`	a vector that indicates the treatment status of each observation. 1 represents treated and 0 represents control. Only binary treatment supported in this version.
`weights`	optional case weights.
`cost`	a vector of non-negative costs, one for each variable in the model. Defaults to one for all variables. These are scalings to be applied when considering splits, so the improvement on splitting on a variable is divided by its cost in deciding which split to choose.
`num.trees`	Number of trees to be built in the causal forest
`ncov_sample`	Number of covariates randomly sampled to build each tree in the forest
`object`	a `causalTree` object
`newdata`	new data to predict
`predict.all`	If TRUE, return predicted individual effect for each observations. Otherwise, return the average effect.
`type`	the type of returned object

...	arguments to rpart.control may also be specified in the call to causalForest. They are checked against the list of valid arguments. The parameter minsize is implemented differently in causalTree than in rpart; we require a minimum of minsize treated observations and a minimum of minsize control observations in each leaf.
na.action	the default action deletes all observations for which y is missing, but keeps those in which one or more predictors are missing.
split.Rule	causalTree splitting options, one of "TOT", "CT", "fit", "tstats", four splitting rules in causalTree. Note that the "tstats" alternative does not have an associated cross-validation method cv.option; see Athey and Imbens (2016) for a discussion. Note further that split.Rule and cv.option can mix and match.
double.Sample	boolean option, TRUE or FALSE, if set to True, causalForest will build honest trees.
split.Honest	boolean option, TRUE or FALSE, used to decide the splitting rule of the trees.
split.Bucket	boolean option, TRUE or FALSE, used to specify whether to apply the discrete method in splitting the tree. If set as TRUE, in splitting a node, the observations in a leaf will be be partitioned into buckets, with each bucket containing bucketNum treated and bucketNum control units, and where observations are ordered prior to partitioning. Splitting will take place by bucket.
bucketNum	number of observations in each bucket when set split.Bucket = TRUE. However, the code will override this choice in order to guarantee that there are at least minsize and at most bucketMax buckets.
bucketMax	Option to choose maximum number of buckets to use in splitting when set split.Bucket = TRUE, bucketNum can change by choice of bucketMax.
cv.option	cross validation options, one of "TOT", "matching", "CT", "fit", four cross validation methods in **causalTree**. There is no cv.option for the split.Rule "tstats"; see Athey and Imbens (2016) for discussion.
cv.Honest	boolean option, TRUE or FALSE, only used for cv.option as "CT" or "fit", to specify whether to apply honest risk evalation function in cross validation. If set TRUE, use honest risk function, otherwise use adaptive risk function in cross validation. If set FALSE, the user choice of cv.alpha will be set to 1. If set TRUE, cv.alpha will default to 0.5, but the user choice of cv.alpha will be respected. Note that honest cv estimates within-leaf variances and may perform better with larger leaf sizes and/or small number of cross-validation sets.
minsize	in order to split, each leaf must have at least minsize treated cases and minsize control cases. The default value is set as 2.
propensity	propensity score used in "TOT" splitting and "TOT", honest "CT" cross validation methods. The default value is the proportion of treated cases in all observations. In this implementation, the propensity score is a constant for the whole dataset. Unit-specific propensity scores are not supported; however, the user may use inverse propensity scores as case weights if desired.
control	a list of options that control details of the rpart algorithm. See rpart.control.
split.alpha	scale parameter between 0 and 1, used in splitting risk evaluation function for "CT". When split.Honest = FALSE, split.alpha will be set as 1. For split.Rule="tstats",

if `split.Honest=TRUE`, `split.alpha` is used in calculating the risk function, which determines the order of pruning in cross-validation.

`cv.alpha` scale paramter between 0 and 1, used in cross validation risk evaluation function for `"CT"` and `"fit"`. When `cv.Honest = FALSE`, `cv.alpha` will be set as 1.

`sample.size.total`
Sample size used to build each tree in the forest (sampled randomly with replacement).

`sample.size.train.frac`
Fraction of the sample size used for building each tree (training). For eexample, if the sample.size.total is 1000 and frac =0.5 then, 500 samples will be used to build the tree and the other 500 samples will be used the evaluate the tree.

`mtry` Number of data features used to build a tree (This variable is not used presently).

`nodesize` Minimum number of observations for treated and control cases in one leaf node

`ncolx` Total number of covariates

Details

CausalForest builds an ensemble of CausalTrees (See Athey and Imbens, *Recursive Partitioning for Heterogeneous Causal Effects* (2016)), by repeated random sampling of the data with replacement. Further, each tree is built using a randomly sampled subset of all available covariates. A causal forest object is a list of trees. To predict, call R's predict function with new test data and the causalForest object (estimated on the training data) obtained after calling the causalForest function. During the prediction phase, the average value over all tree predictions is returned as the final prediction by default. To return the predictions of each tree in the forest for each test observation, set the flag `predict.all=TRUE` CausalTree differs from `rpart` function from **rpart** package in splitting rules and cross validation methods. Please check Athey and Imbens, *Recursive Partitioning for Heterogeneous Causal Effects* (2016) and Stefan Wager and Susan Athey, *Estimation and Inference of Heterogeneous Treatment Effects using Random Forests* for more details.

Value

An object of class `rpart`. See `rpart.object`.

References

Breiman L., Friedman J. H., Olshen R. A., and Stone, C. J. (1984) *Classification and Regression Trees.* Wadsworth.

Athey, S and G Imbens (2016) *Recursive Partitioning for Heterogeneous Causal Effects*. http://arxiv.org/abs/1504.01132

Wager,S and Athey, S (2015) *Estimation and Inference of Heterogeneous Treatment Effects using Random Forests* http://arxiv.org/abs/1510.04342

See Also

`causalTree` `honest.causalTree`, `rpart.control`, `rpart.object`, `summary.rpart`, `rpart.plot`

Examples

```
library(rpart)
library("htetree")
cf <- causalForest(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10, data=simulation.1,
  treatment=simulation.1$treatment,
  split.Rule="CT", split.Honest=TRUE,
  split.Bucket=FALSE, bucketNum = 5,
  bucketMax = 100, cv.option="CT", cv.Honest=TRUE, minsize = 2L,
  split.alpha = 0.5, cv.alpha = 0.5,
  sample.size.total = floor(nrow(simulation.1) / 2),
  sample.size.train.frac = .5,
  mtry = ceiling(ncol(simulation.1)/3), nodesize = 3, num.trees= 5,
  ncolx=10,ncov_sample=3)

cfpredtest <- predict.causalForest(cf, newdata=simulation.1[1:100,],
  type="vector")
```

---

makeplots *Visualize Causal Tree and the Estimated Results*

Description

An intermediate function used for plotting

Usage

```
makeplots(
  negative,
  opfit. = opfit,
  trainset,
  covariates,
  outcomevariable,
  data. = data,
  hte_effect_setup,
  varlabel,
  maintitle,
  legend.x = 0.8,
  legend.y = 0.25,
  ...
)
```

Arguments

`negative`	a logical value indicating whether we expect the treatment effect to be negative. The default is set as FALSE.
`opfit.`	tree structure generated from causal tree algorithm.
`trainset`	a data frame only containing the variables used in the model and missings values are listwise deleted.

`covariates` a vector of column names of all covariates (linear terms andpropensity score).

`outcomevariable`
a character representing the column name of the outcome variable.

`data.` a data frame containing the variables in the model.

`hte_effect_setup`
a empty list to store the adjusted treatment effect.

`varlabel` a named vector containing variable labels.

`maintitle` a character string indicating the main title displayed when plotting the tree and results. The default is set as "Heterogeneous Treatment Effect Estimation".

`legend.x, legend.y`
x and y coordinate to position the legend. The default is set as (0.08, 0.25).

`...` further arguments passed to or from other methods.

Value

A plot visualizing the tree and estimated treatment effect in each node.

`matchinleaves` *NN Matching in Leaves*

Description

This intermediate function is used to adjust the heterogeneous treatment effect estimated in each leaf with NN matching.

Usage

```
matchinleaves(
  trainset = match_data,
  covariates = covariates,
  outcomevariable = outcomevariable,
  hte_effect_setup = hte_effect_setup,
  treatment_indicator,
  con.num = 1,
  ...
)
```

Arguments

`trainset` a data frame only containing the variables used in the model and missings values are listwise deleted.

`covariates` a vector of column names of all covariates (linear terms andpropensity score).

`outcomevariable`
a character representing the column name of the outcome variable.

`hte_effect_setup`
: a empty list to store the adjusted treatment effect.

`treatment_indicator`
: a character representing the column name of the treatment indicator.

`con.num`
: a number indicating the number of units from control groups to be used in matching

`...`
: further arguments passed to or from other methods.

Value

A list for summarizing the results after matching.

`model.frame.causalTree` *Intermediate function for* `causalTree`

Description

get model frame of causalTree, same as rpart

Usage

```
## S3 method for class 'causalTree'
model.frame(formula, ...)
```

Arguments

`formula`
: a formula, with a response but no interaction terms. If this is a data frame, it is taken as the model frame (see model.frame).

`...`
: arguments to `rpart.control` may also be specified in the call to `causalTree`. They are checked against the list of valid arguments. An example of a commonly set parameter would be `xval`, which sets the number of cross-validation samples. The parameter `minsize` is implemented differently in `causalTree` than in `rpart`; we require a minimum of `minsize` treated observations and a minimum of `minsize` control observations in each leaf.

Value

a model frame for `causalTree`.

na.causalTree — *Intermediate function for* `causalTree`

Description

requirement when missing values are included in sample.

Usage

```
na.causalTree(x)
```

Arguments

`x`	covariates

Value

No return value, used for handling missing values when thy are included in sample.

plotOutcomes — *Intermediate function for* `hte_plot_line`

Description

Plots the different least squares models used to estimate heterogeneous treatment effects(HTE) at each node. At each node, this visualization aims to show how the estimated treatment effect differs when using ordinary least squares and weighted least squares methods. The weighted least squares method in this package uses inverse propensity scores as weights, in order to reduce bias due to confounding variables.

Usage

```
plotOutcomes(
  treatment,
  outcome,
  propscores,
  confInt = TRUE,
  colbyWt = FALSE,
  ylab = "",
  xlab = "",
  title = "",
  gamma = 0,
  lambda = 0,
  ...
)
```

Arguments

`treatment` a character representing the column name for the treatment variable in the causal setup

`outcome` a character representing the column name of the outcome variable.

`propscores` a character representing the column name of the propensity score.

`confInt` a logical value indicating whether adding the 95 confidence interval. The default is set as `TRUE`.

`colbyWt` a logical value indicating whether the points are are colored according to inverse propensity scores. The default is set as FALSE.

`xlab, ylab, title` Characters representing the name for x axis, y axis, and main title for each node.

`gamma, lambda` numbers indicating the bias level used in sensitivity analysis

`...` further arguments passed to or from other methods.

Value

A summary table after adjusting the estimates with inverse probability weighting (ipw).

---

runDynamic — *Visualize Causal Tree and Treatment Effects via Shiny*

Description

Visualize Causal Tree and Treatment Effects via Shiny

Usage

```
runDynamic(
  model,
  data,
  outcomevariable,
  treatment_indicator,
  propensity_score = ""
)
```

Arguments

`model` a tree model constructed by `hte_causalTree`, `hte_matchinleaves`,or `hte_ipw`.

`data` a data frame containing the variables in the model.

`outcomevariable` a character representing the column name of the outcome variable.

`treatment_indicator` a character representing the column name for the treatment variable in the causal setup.

`propensity_score` a character representing the column name of the propensity score.

Value

a Shiny page.

`saveBCSS` *Save Javascript Embedded in Shiny App*

Description

Save Javascript Embedded in Shiny App

Usage

```
saveBCSS(filePath)
```

Arguments

`filePath`	a character string representing the path name to save the files temporarily.

Value

No return value. It is used to save necessary files temporarily to run Shiny App.

`saveFiles` *Save Necessary Files to Run Shiny App*

Description

This function is to save files necessary to run Shiny app to visualize causal tree and the estimated heterogeneous treatment effects in an interactive way.

Usage

```
saveFiles(
  model,
  data,
  outcomevariable,
  treatment_indicator,
  propensity_score = "",
  filePath = ""
)
```

Arguments

- `model` a tree model constructed by `hte_causalTree`, `hte_matchinleaves`, or `hte_ipw`.
- `data` a data frame containing the variables in the model.
- `outcomevariable` a character representing the column name of the outcome variable.
- `treatment_indicator` a character representing the column name for the treatment variable in the causal setup.
- `propensity_score` a character representing the column name of the propensity score.
- `filePath` a character string representing the path name to save the files temporarily.

Value

No return value. It is used to save necessary files temporarily to run Shiny App.

`saveGCSS` *Save CSS File Embedded in Shiny App*

Description

Save CSS File Embedded in Shiny App

Usage

```
saveGCSS(filePath)
```

Arguments

- `filePath` a character string representing the path name to save the files temporarily.

Value

No return value. It is used to save necessary files temporarily to run Shiny App.

saveInd — *Save HTML Index Embedded in Shiny App*

Description

Save HTML Index Embedded in Shiny App

Usage

```
saveInd(filePath)
```

Arguments

`filePath`	a character string representing the path name to save the files temporarily.

Value

No return value. It is used to save necessary files temporarily to run Shiny App.

saveServ — *Save Shiny Server Temporarily*

Description

Save Shiny Server Temporarily

Usage

```
saveServ(filePath)
```

Arguments

`filePath`	a character string representing the path name to save the files temporarily.

Value

No return value. It is used to save necessary files temporarily to run Shiny App.

saveUI *Save Shiny UI Temporarily*

Description

Save Shiny UI Temporarily

Usage

```
saveUI(filePath)
```

Arguments

filePath a character string representing the path name to save the files temporarily.

Value

No return value. It is used to save necessary files temporarily to run Shiny App.

simulation.1 *A Simulated Dataset*

Description

A simulated dataset inherited from `causalTree` package

Usage

```
simulation.1
```

Format

'simulation.1' A data frame with 500 observations on the following 12 variables.

`x1` a numeric vector

`x2` a numeric vector

`x3` a numeric vector

`x4` a numeric vector

`x5` a numeric vector

`x6` a numeric vector

`x7` a numeric vector

`x8` a numeric vector

`x9` a numeric vector

`x10` a numeric vector

`y` a numeric vector

`treatment` a numeric vector

Index