

# Package ‘causalLearning’

February 11, 2018

**Title** Methods for heterogeneous treatment effect estimation

**Version** 1.0.0

**Description** The main functions are `cv.causalBoosting` and `bagged.causalMARS`, which build upon the simpler `causalBoosting` and `causalMARS` functions. All of these functions have their own predict methods.

**Depends** R ( $\geq 3.3.0$ )

**Imports** ranger

**License** GPL-2

**LazyData** true

**RoxygenNote** 6.0.1

**NeedsCompilation** yes

**Author** Scott Powers [aut, cre],  
Junyang Qian [aut],  
Trevor Hastie [aut],  
Robert Tibshirani [aut]

**Maintainer** Scott Powers <saberpowers@gmail.com>

## R topics documented:

bagged.causalMARS . . . . .	2
causalBoosting . . . . .	3
causalMARS . . . . .	4
cv.causalBoosting . . . . .	7
pollinated.ranger . . . . .	8
predict.bagged.causalMARS . . . . .	9
predict.causalBoosting . . . . .	9
predict.causalMARS . . . . .	10
predict.causalTree . . . . .	11
predict.cv.causalBoosting . . . . .	11
predict.pollinated.ranger . . . . .	12
predict.PTOforest . . . . .	13
PTOforest . . . . .	13
stratify . . . . .	14
<b>Index</b>	<b>16</b>

---

bagged.causalMARS	<i>Fit a bag of causal MARS models</i>
-------------------	--

---

## Description

Fit a bag of causal MARS models

## Usage

```
bagged.causalMARS(x, tx, y, nbag = 20, maxterms = 11, nquant = 5,
  degree = ncol(x), eps = 1, backstep = FALSE, propensity = FALSE,
  stratum = rep(1, nrow(x)), minnum = 5, verbose = FALSE)
```

## Arguments

x	matrix of covariates
tx	vector of treatment indicators (0 or 1)
y	vector of response values
nbag	number of models to bag
maxterms	maximum number of terms to include in the regression basis (e.g. maxterms = 11 means intercept + 5 pairs added)
nquant	number of quantiles used in splitting
degree	max number of different predictors that can interact in model
eps	shrinkage factor for new term added
backstep	logical: should out-of-bag samples be used to prune each model? otherwise full regression basis is used for each model
propensity	logical: should propensity score stratification be used?
stratum	optional vector giving propensity score stratum for each observation (only used if propensity = TRUE)
minnum	minimum number of observations in each arm of each propensity score stratum needed to estimate regression coefficients for basis (only used if propensity = TRUE)
verbose	logical: should progress be printed to console?

## Value

an object of class bagged.causalMARS, which is itself a list of causalMARS objects

## Examples

```
# Randomized experiment example

n = 100 # number of training-set patients to simulate
p = 10  # number of features for each training-set patient

# Simulate data
x = matrix(rnorm(n * p), nrow = n, ncol = p) # simulate covariate matrix
tx_effect = x[, 1] + (x[, 2] > 0) # simple heterogeneous treatment effect
tx = rbinom(n, size = 1, p = 0.5) # random treatment assignment
```

```

y = rowMeans(x) + tx * tx_effect + rnorm(n, sd = 0.001) # simulate response

# Estimate bagged causal MARS model
fit_bcm = causalLearning::bagged.causalMARS(x, tx, y, nbag = 10)
pred_bcm = predict(fit_bcm, newx = x)

# Visualize results
plot(tx_effect, pred_bcm, main = 'Bagged causal MARS',
     xlab = 'True treatment effect', ylab = 'Estimated treatment effect')
abline(0, 1, lty = 2)

```

causalBoosting

*Fit a causal boosting model***Description**

Fit a causal boosting model

**Usage**

```

causalBoosting(x, tx, y, num.trees = 500, maxleaves = 4, eps = 0.01,
  splitSpread = 0.1, x.est = NULL, tx.est = NULL, y.est = NULL,
  propensity = FALSE, stratum = NULL, stratum.est = NULL,
  isConstVar = TRUE)

```

**Arguments**

x	matrix of covariates
tx	vector of treatment indicators (0 or 1)
y	vector of response values
num.trees	number of shallow causal trees to build
maxleaves	maximum number of leaves per causal tree
eps	learning rate
splitSpread	how far apart should the candidate splits be for the causal trees? (e.g. splitSpread = 0.1) means we consider 10 quantile cutpoints as candidates for making split
x.est	optional matrix of estimation-set covariates used for honest re-estimation (ignored if tx.est = NULL or y.est = NULL)
tx.est	optional vector of estimation-set treatment indicators (ignored if x.est = NULL or y.est = NULL)
y.est	optional vector of estimation-set response values (ignored if x.est = NULL or y.est = NULL)
propensity	logical: should propensity score stratification be used?
stratum	optional vector giving propensity score stratum for each observation (only used if propensity = TRUE)
stratum.est	optional vector giving propensity score stratum for each estimation-set observation (ignored if x.est = NULL or tx.est = NULL or y.est = NULL)
isConstVar	logical: for the causal tree splitting criterion (T-statistic), should it be assumed that the noise variance is the same in treatment and control arms?

## Details

This function exists primarily to be called by `cv.causalBoosting` because the `num.trees` parameter generally needs to be tuned via cross-validation.

## Value

an object of class `causalBoosting` with attributes:

- `CBM`: a list storing the intercept, the causal trees and `eps`
- `tauhat`: matrix of treatment effects for each patient for each step
- `G1`: estimated-treatment conditional mean for each patient
- `G0`: estimated-control conditional mean for each patient
- `err.y`: training error at each step, in predicting response
- `num.trees`: number of trees specified by function call

## Examples

```
# Randomized experiment example

n = 100 # number of training-set patients to simulate
p = 10  # number of features for each training-set patient

# Simulate data
x = matrix(rnorm(n * p), nrow = n, ncol = p) # simulate covariate matrix
tx_effect = x[, 1] + (x[, 2] > 0) # simple heterogeneous treatment effect
tx = rbinom(n, size = 1, p = 0.5) # random treatment assignment
y = rowMeans(x) + tx * tx_effect + rnorm(n, sd = 0.001) # simulate response

# Estimate causal boosting model
fit_cb = causalBoosting(x, tx, y, num.trees = 500)
pred_cb = predict(fit_cb, newx = x, num.trees = 500)

# Visualize results
plot(tx_effect, pred_cb, main = 'Causal boosting',
     xlab = 'True treatment effect', ylab = 'Estimated treatment effect')
abline(0, 1, lty = 2)
```

---

causalMARS

*Fit a causal MARS model*


---

## Description

Fit a causal MARS model

## Usage

```
causalMARS(x, tx, y, maxterms = 11, nquant = 5, degree = ncol(x),
  eps = 1, backstep = FALSE, x.val = NULL, tx.val = NULL,
  y.val = NULL, propensity = FALSE, stratum = rep(1, nrow(x)),
  stratum.val = NULL, minnum = 5)
```

**Arguments**

<code>x</code>	matrix of covariates
<code>tx</code>	vector of treatment indicators (0 or 1)
<code>y</code>	vector of response values
<code>maxterms</code>	maximum number of terms to include in the regression basis (e.g. <code>maxterms = 11</code> means intercept + 5 pairs added)
<code>nquant</code>	number of quantiles used in splitting
<code>degree</code>	max number of different predictors that can interact in model
<code>eps</code>	shrinkage factor for new term added
<code>backstep</code>	logical: after building out regression basis, should backward stepwise selection be used to create a sequence of models, with the criterion evaluated on a validation set to choose among the sequence?
<code>x.val</code>	optional matrix of validation-set covariates (only used if <code>backstep = TRUE</code> )
<code>tx.val</code>	optional vector of validation-set treatment indicators (only used if <code>backstep = TRUE</code> )
<code>y.val</code>	optional vector of validation-set response values (only used if <code>backstep = TRUE</code> )
<code>propensity</code>	logical: should propensity score stratification be used?
<code>stratum</code>	optional vector giving propensity score stratum for each observation (only used if <code>propensity = TRUE</code> )
<code>stratum.val</code>	optional vector giving propensity score stratum for each validation-set observation (only used if <code>propensity = backstep = TRUE</code> )
<code>minnum</code>	minimum number of observations in each arm of each propensity score stratum needed to estimate regression coefficients for basis (only used if <code>propensity = TRUE</code> )

**Details**

parallel arms mars with backward stepwise BOTH randomized case and propensity stratum. data structures: model terms (nodes) are numbered 1, 2, ... with 1 representing the intercept. forward stepwise: `modmatrix` contains basis functions as model is built up – two columns are added at each step. Does not include a column of ones for tidiness, we always add two terms, even when term added in linear (so that reflected version is just zero). backward stepwise: `khat` is the sequence of terms deleted at each step, based on  $\Delta\hat{rss}$  = relative change in rss. `rss.test` is rss over test (validation) set achieved by each reduced model in sequence- used later for selecting a member of the sequence. `active2` contains indices of columns with nonzero norm

**Value**

an object of class `causalMARS` with attributes:

- `parent`: indices of nodes that are parents at each stage
- `childvar`: index of predictor chosen at each forward step
- `childquant`: quantile of cutoff chosen at each forward step
- `quant`: quantiles of the columns of `x`
- `active`: indices of columns with nonzero norm
- `allvars`: list of variables appearing in each term
- `khat`: the sequence of terms deleted at each step
- `deltahat`: relative change in rss

- `rsstestthat`: validation-set rss achieved by each model in sequence
- `setestthat`: standard error for `rsstestthat`
- `tim1`: time elapsed during forward stepwise phase
- `tim2`: total time elapsed
- `x`
- `tx`
- `y`
- `maxterms`
- `eps`
- `backstep`
- `propensity`
- `x.val`
- `tx.val`
- `y.val`
- `stratum`
- `stratum.val`
- `minnum`

## Examples

```
# Randomized experiment example

n = 100 # number of training-set patients to simulate
p = 10  # number of features for each training-set patient

# Simulate data
x = matrix(rnorm(n * p), nrow = n, ncol = p) # simulate covariate matrix
tx_effect = x[, 1] + (x[, 2] > 0) # simple heterogeneous treatment effect
tx = rbinom(n, size = 1, p = 0.5) # random treatment assignment
y = rowMeans(x) + tx * tx_effect + rnorm(n, sd = 0.001) # simulate response

# Estimate causal MARS model
fit_cm = causalLearning::causalMARS(x, tx, y)
pred_cm = predict(fit_cm, newx = x)

# Visualize results
plot(tx_effect, pred_cm, main = 'Causal MARS',
     xlab = 'True treatment effect', ylab = 'Estimated treatment effect')
abline(0, 1, lty = 2)
```

---

cv.causalBoosting	<i>Fit a causal boosting model with cross validation</i>
-------------------	--

---

## Description

Fit a causal boosting model with cross validation

## Usage

```
cv.causalBoosting(x, tx, y, num.trees = 500, maxleaves = 4, eps = 0.01,
  splitSpread = 0.1, type.measure = c("effect", "response"), nfolds = 5,
  foldid = NULL, propensity = FALSE, stratum = NULL, isConstVar = TRUE)
```

## Arguments

x	matrix of covariates
tx	vector of treatment indicators (0 or 1)
y	vector of response values
num.trees	number of shallow causal trees to build
maxleaves	maximum number of leaves per causal tree
eps	learning rate
splitSpread	how far apart should the candidate splits be for the causal trees? (e.g. splitSpread = 0.1) means we consider 10 quantile cutpoints as candidates for making split
type.measure	loss to use for cross validation: 'response' returns mean-square error for predicting response in each arm. 'effect' returns MSE for treatment effect using honest over-fit estimation.
nfolds	number of cross validation folds
foldid	vector of fold membership
propensity	logical: should propensity score stratification be used?
stratum	optional vector giving propensity score stratum for each observation (only used if propensity = TRUE)
isConstVar	logical: for the causal tree splitting criterion (T-statistic), should it be assumed that the noise variance is the same in treatment and control arms?

## Value

an object of class `cv.causalBoosting` which is an object of class `causalBoosting` with these additional attributes:

- `num.trees.min`: number of trees with lowest CV error
- `cvm`: vector of mean CV error for each number of trees
- `cvstd`: vector of standard errors for mean CV errors

## Examples

```
# Randomized experiment example

n = 100 # number of training-set patients to simulate
p = 10  # number of features for each training-set patient

# Simulate data
x = matrix(rnorm(n * p), nrow = n, ncol = p) # simulate covariate matrix
tx_effect = x[, 1] + (x[, 2] > 0) # simple heterogeneous treatment effect
tx = rbinom(n, size = 1, p = 0.5) # random treatment assignment
y = rowMeans(x) + tx * tx_effect + rnorm(n, sd = 0.001) # simulate response

# Estimate causal boosting model with cross-validation
fit_cv = causalLearning::cv.causalBoosting(x, tx, y)
fit_cv$num.trees.min.effect # number of trees chosen by cross-validation
pred_cv = predict(fit_cv, newx = x)

# Visualize results
plot(tx_effect, pred_cv, main = 'Causal boosting w/ CV',
     xlab = 'True treatment effect', ylab = 'Estimated treatment effect')
abline(0, 1, lty = 2)
```

---

pollinated.ranger

*Pollinate a fitted ranger random forest model*

---

## Description

Pollinate a fitted ranger random forest model

## Usage

```
pollinated.ranger(object, x, y)
```

## Arguments

object	a fitted ranger object
x	matrix of covariates
y	vector of response values

## Value

an object of class `pollinated.ranger` which is a ranger object that has been pollinated with the data in (x, y)



---

predict.bagged.causalMARS

*Make predictions from a bag of fitted causal MARS models*


---

### Description

Make predictions from a bag of fitted causal MARS models

### Usage

```
## S3 method for class 'bagged.causalMARS'
predict(object, newx, type = c("average", "all"),
  ...)
```

### Arguments

object	a fitted bagged.causalMARS object
newx	matrix of new covariates for which estimated treatment effects are desired
type	type of prediction required: 'average' returns a vector of the averages of the bootstrap estimates. 'all' returns a matrix of all of the bootstrap estimates.
...	ignored

### Value

a vector of estimated personalized treatment effects corresponding to the rows of newx

---

predict.causalBoosting

*Make predictions from a fitted causal boosting model*


---

### Description

Make predictions from a fitted causal boosting model

### Usage

```
## S3 method for class 'causalBoosting'
predict(object, newx, newtx = NULL,
  type = c("treatment.effect", "conditional.mean", "response"),
  num.trees = 1:object$num.trees, honest = FALSE, naVal = 0, ...)
```

**Arguments**

object	a fitted causalBoosting object
newx	matrix of new covariates for which estimated treatment effects are desired
newtx	option vector of new treatment assignments (only used if type = 'response')
type	type of prediction required: 'treatment.effect' returns estimated treatment effect. 'conditional.mean' returns two predictions, one for each arm. 'response' returns prediction for arm corresponding to newtx.
num.trees	number(s) of shallow causal trees to use for prediction
honest	logical: should honest re-estimates of leaf means be used for prediction? This requires that x.est, tx.est, y.est were specified when the causal boosting model was fit
naVal	value with which to replace NA predictions
...	ignored

**Value**

a vector or matrix of predictions corresponding to the rows of newx

---

predict.causalMARS	<i>Make predictions from a fitted causal MARS model</i>
--------------------	---

---

**Description**

Make predictions from a fitted causal MARS model

**Usage**

```
## S3 method for class 'causalMARS'
predict(object, newx, active, ...)
```

**Arguments**

object	a fitted causalMARS object
newx	matrix of new covariates for which estimated treatment effects are desired
active	indices of columns with nonzero norm (defaults to model selected via backward stepwise phase, or the full model if backstep = FALSE)
...	ignored

**Value**

a vector of estimated personalized treatment effects corresponding to the rows of newx

---

predict.causalTree	<i>Make predictions from a fitted causal tree model</i>
--------------------	---

---

### Description

Make predictions from a fitted causal tree model

### Usage

```
## S3 method for class 'causalTree'
predict(object, newx, newtx = NULL,
  type = c("treatment.effect", "conditional.mean", "response"),
  honest = FALSE, naVal = 0, ...)
```

### Arguments

object	a fitted causalTree object
newx	matrix of new covariates for which estimated treatment effects are desired
newtx	option vector of new treatment assignments (only used if type = 'response')
type	type of prediction required: 'treatment.effect' returns estimated treatment effect. 'conditional.mean' returns two predictions, one for each arm. 'response' returns prediction for arm corresponding to newtx.
honest	logical: should honest re-estimates of leaf means be used for prediction? This requires that x.est, tx.est, y.est were specified when the causal boosting model was fit
naVal	value with which to replace NA predictions
...	ignored

### Value

a vector or matrix of predictions corresponding to the rows of newx

---

predict.cv.causalBoosting	<i>Make predictions from a fitted cross-validated causal boosting model</i>
---------------------------	---

---

### Description

Make predictions from a fitted cross-validated causal boosting model

### Usage

```
## S3 method for class 'cv.causalBoosting'
predict(object, newx, newtx = NULL,
  type = c("treatment.effect", "conditional.mean", "response"),
  num.trees = object$num.trees.min.effect, naVal = 0, ...)
```

**Arguments**

object	a fitted cv.causalBoosting object
newx	matrix of new covariates for which estimated treatment effects are desired
newtx	option vector of new treatment assignments (only used if type = 'individual')
type	type of prediction required: 'treatment.effect' returns estimated treatment effect. 'conditional.mean' returns two predictions, one for each arm. 'response' returns prediction for arm corresponding to newtx.
num.trees	number of shallow causal trees to use for prediction
naVal	value with which to replace NA predictions
...	ignored

**Value**

a vector or matrix of predictions corresponding to the rows of newx

---

predict.pollinated.ranger

*Make predictions from a pollinated ranger random forest model*

---

**Description**

Make predictions from a pollinated ranger random forest model

**Usage**

```
## S3 method for class 'pollinated.ranger'
predict(object, newx, predict.all = FALSE,
        na.treatment = c("omit", "replace", "NA"), ...)
```

**Arguments**

object	a fitted pollinated.ranger object
newx	matrix of new covariates for which predictions are desired
predict.all	logical: should predictions from all trees be returned? Otherwise the average across trees is returned
na.treatment	how to treat NA predictions from individual trees: 'omit' only uses trees for which the prediction is not NA. 'replace' replaces NA predictions with the over-all mean response. 'NA' returns NA if any tree prediction is NA.
...	additional arguments passed on to predict.ranger

**Value**

a vector of predicted treatment effects corresponding to the rows of newx

---

predict.PTOforest	<i>Make predictions from a fitted PTO forest model</i>
-------------------	--

---

**Description**

Make predictions from a fitted PTO forest model

**Usage**

```
## S3 method for class 'PTOforest'
predict(object, newx, ...)
```

**Arguments**

object	a fitted PTOforest object
newx	matrix of new covariates for which estimated treatment effects are desired
...	ignored

**Value**

a vector of predictions corresponding to the rows of newx

---

PTOforest	<i>Fit a pollinated transformed outcome (PTO) forest model</i>
-----------	--

---

**Description**

Fit a pollinated transformed outcome (PTO) forest model

**Usage**

```
PTOforest(x, tx, y, pscore = rep(0.5, nrow(x)), num.trees = 500,
  mtry = ncol(x), min.node.size = max(25, nrow(x)/40), postprocess = TRUE,
  verbose = FALSE)
```

**Arguments**

x	matrix of covariates
tx	vector of treatment indicators (0 or 1)
y	vector of response values
pscore	vector of propensity scores
num.trees	number of trees for transformed outcome forest
mtry	number of variables to possibly split at in each node
min.node.size	minimum node size for transformed outcome forest
postprocess	logical: should optional post-processing random forest be fit at end?
verbose	logical: should progress be printed to console?

**Value**

an object of class PTOforest with attributes:

- x: matrix of covariates supplied by function call
- pscore: vector of propensity score supplied by function call
- postprocess: logical supplied by function call
- TOfit: fitted random forest on transformed outcomes
- PTOfit1: TOfit pollinated with treatment-arm outcomes
- PTOfit0: TOfit pollinated with control-arm outcomes
- postfit: post-processing random forest summarizing results

**Examples**

```
# Randomized experiment example

n = 100 # number of training-set patients to simulate
p = 10  # number of features for each training-set patient

# Simulate data
x = matrix(rnorm(n * p), nrow = n, ncol = p) # simulate covariate matrix
tx_effect = x[, 1] + (x[, 2] > 0) # simple heterogeneous treatment effect
tx = rbinom(n, size = 1, p = 0.5) # random treatment assignment
y = rowMeans(x) + tx * tx_effect + rnorm(n, sd = 0.001) # simulate response

# Estimate PTO forest model
fit_pto = PTOforest(x, tx, y)
pred_pto = predict(fit_pto, newx = x)

# Visualize results
plot(tx_effect, pred_pto, main = 'PTO forest',
     xlab = 'True treatment effect', ylab = 'Estimated treatment effect')
abline(0, 1, lty = 2)
```

---

stratify

---

*Get propensity strata from propensity scores*


---

**Description**

Get propensity strata from propensity scores

**Usage**

```
stratify(pscore, tx, min.per.arm = 30)
```

**Arguments**

pscore	vector of propensity scores
tx	vector of treatment indicators
min.per.arm	minimum number of observations for each arm within each stratum

**Value**

a vector of integers with length equal to the length of `pscore`, reporting the propensity stratum corresponding to each propensity score

# Index

`bagged.causalMARS`, [2](#)

`causalBoosting`, [3](#)

`causalMARS`, [4](#)

`cv.causalBoosting`, [7](#)

`pollinated.ranger`, [8](#)

`predict.bagged.causalMARS`, [9](#)

`predict.causalBoosting`, [9](#)

`predict.causalMARS`, [10](#)

`predict.causalTree`, [11](#)

`predict.cv.causalBoosting`, [11](#)

`predict.pollinated.ranger`, [12](#)

`predict.PTOforest`, [13](#)

`PTOforest`, [13](#)

`stratify`, [14](#)