Package ‘htree’

July 13, 2018

Version 2.0.0
Date 2018-06-23
Depends R (>= 2.9.0), parallel
Title Historical Tree Ensembles for Longitudinal Data
Author Joe Sexton <joesexton@gmail.com>
Description Historical regression trees are an extension of standard trees, producing a non-parametric estimate of how the response depends on all of its prior realizations as well as that of any time-varying predictor variables. The method applies equally to regularly as well as irregularly sampled data. The package implements random forest and boosting ensembles based on historical regression trees, suitable for longitudinal data. Standard error estimation and Z-score variable importance is also implemented.
Maintainer Joe Sexton <joesexton@gmail.com>
License GPL (>= 2)
Repository CRAN
NeedsCompilation yes
Date/Publication 2018-07-13 15:00:03 UTC

R topics documented:

  cd4 ................................................................. 2
  hrf ............................................................... 2
  htb ............................................................... 9
  misc ............................................................ 15
  mscm ........................................................... 16
  partdep_hrf ................................................... 17
  predict_hrf ..................................................... 20
  varimp_hrf ..................................................... 23

Index 27
Description
CD4 cell counts for subjects over time.

Usage
data(cd4)

Format
cd4

Source
The data was obtained from http://www.lancaster.ac.uk/staff/diggle/APTS-data-sets/CD4_data.txt.

References

Description
Random forest for longitudinal data

Usage
hrf(x,
    time=NULL,
    id=NULL,
    yindx,
    ntrees = 100,
    method="freq",
    mtry=NULL,
    se=FALSE,
Arguments

- **x**: A data frame containing response and predictors.
- **time**: A vector of observation times associated with rows of x.
- **id**: Subject identifier, if NULL observations are assumed independent.
- **yindx**: Name of response variable, alt. its column number in x.
- **ntrees**: Number of trees in ensemble.
- **method**: Historical summary method, can be freq, frac, mean0, freqw, fracw and mean0w (see below).
- **mtry**: Number of predictors sampled at each split.
- **se**: If TRUE then bootstrap standard errors are computed. Total number of trees fit for bootstrapping is B*R.
- **B**: Only used if se=TRUE, number of bootstrap samples, defaults to 100.
- **R**: Only used if se=TRUE, forest size for each bootstrap sample, defaults to R=10.
- **nsamp**: Number of sampled delta values, see below.
- **historical**: If TRUE then historical splitting is done, else only standard (ie concurrent predictor) splitting.
- **vh**: Optional vector of variable names to be used as historical predictors, can be a numeric vector giving column numbers of x.
- **vc**: Optional vector of variable names to be used as concurrent predictors, can be a numeric vector giving column numbers of x.
- **delta**: An optional vector of time-lags to be used (see below).
- **classify**: If TRUE then a classification tree is built (using gini-impurity node splitting).
- **control**: A list of control parameters (see below). All arguments, except those describing the data, can be set in control. Arguments in control are used if both are given.

Details

The **hrf** function fits a random forest model to longitudinal data. Data is assumed to be of form: 
\[ z_{ij} = (y_{ij}, t_{ij}, x_{ij}) \]  
for \( i = 1, \ldots, n \) and \( j = 1, \ldots, n_i \), with \( y_{ij} \) being the response for the \( i \)-th subject at the \( j \)-th observation time \( t_{ij} \). The vector of predictors at time \( t_{ij} \) are \( x_{ij} \). The number of observations can vary across subjects, and the sampling in time can be irregular.
hrf estimates a model for the response \( y_{ij} \) using both \((t_{ij}, x_{ij})\) (the observations concurrent with \( y_{ij} \)) and all preceding observations of the \( i \)-th subject up to (but not including) time \( t_{ij} \). The model is fit using historical regression (alt. classification) trees. Here a predictor is one of two types, either concurrent or historical. The concurrent predictors for \( y_{ij} \) are the elements of the vector \((t_{ij}, x_{ij})\), while a historical predictor is the set of all preceding values (i.e. prior to time \( t_{ij} \)) of a given element of \((y_{ij}, t_{ij}, x_{ij})\), for subject \( i \). In a historic regression tree, node splitting on a concurrent predictor follows the approach in standard regression (classification) trees. For historical predictors the splitting is modified since, associated with each observed response \( y_{ij} \), the number (and observation times) of observations of a historical predictor will vary according to \( i \) and \( j \). For these, the splitting is done by first transforming the preceding values of a predictor using a summary function. This summary is invertible, in the sense that knowledge of it is equivalent to knowledge of the covariate history. Letting \( z_{ijk} \) denote the set of historical values of the \( k \)-th element of \( z_{ij} \), the summary function is denoted \( s(\eta; \bar{z}_{ijk}) \) where \( \eta \) is the argument vector of the summary function. Node splitting based on a historical predictor is done by solving

\[
\arg\min_{(ij) \in \text{Node}} \sum_{i \leq t_{ij} - \eta_i \leq t_{ij} - \eta_2} \left( y_{ij} - \mu_L I(s(\eta; \bar{z}_{ijk}) < c) - \mu_R I(s(\eta; \bar{z}_{ijk}) \geq c) \right)^2,
\]

where the minimization is over the vector \((k, \mu, c, \eta)\). Each node of historical regression tree is split using the best split among all splits of concurrent and historical predictors.

Different summary functions are available in hrf, specified by the argument method. If method="freq" the summary function summarizing a covariate history is:

for \( \eta \):

\[
s(\eta; \bar{z}_{ijk}) = \sum_{i \leq t_{ij} - \eta_i \leq t_{ij} - \eta_2} I(z_{ihk} < \eta_2);
\]

method="frac":

\[
s(\eta; \bar{z}_{ijk}) = \sum_{i \leq t_{ij} - \eta_i \leq t_{ij} - \eta_2} I(z_{ihk} < \eta_2)/n_{ij}(\eta);
\]

method="mean0":

\[
s(\eta; \bar{z}_{ijk}) = \sum_{i \leq t_{ij} - \eta_i \leq t_{ij} - \eta_2} z_{ihk}/n_{ij}(\eta);
\]

method="freqw":

\[
s(\eta; \bar{z}_{ijk}) = \sum_{i \leq t_{ij} - \eta_i \leq t_{ij} - \eta_2} I(z_{ihk} < \eta_3);
\]

method="fracw":

\[
s(\eta; \bar{z}_{ijk}) = \sum_{i \leq t_{ij} - \eta_i \leq t_{ij} - \eta_2} I(z_{ihk} < \eta_3)/n_{ij}(\eta);
\]

method="meanw":

\[
s(\eta; \bar{z}_{ijk}) = \sum_{i \leq t_{ij} - \eta_i \leq t_{ij} - \eta_2} z_{ihk}/n_{ij}(\eta).
\]

Here \( n_{ij}(\eta) \) denotes the number of observations of subject \( i \) in the time window \([t_{ij} - \eta_1, t_{ij} - \eta_2]\).

In the case \( n_{ij}(\eta) = 0 \), the summary function is set to zero, i.e \( s(\eta; \bar{z}_{ijk}) = 0 \). The default is method="freq". The possible values of \( \eta_1 \) in the summary function can be set by the argument delta. If not supplied, the set of possible values of \( \eta_1 \) is determined by the difference in time...
between within-subject successive observations in the data. When a split is attempted on a historical predictor, a sample of this set is taken from which the best split is selected. The size of this set equals that of the \texttt{nsamp} argument. See below on \texttt{control} for further arguments governing the historical splitting. See below on \texttt{control} for further arguments governing the historical splitting.

Setting \texttt{se=TRUE} performs standard error estimation. The number of bootstrap samples (sampling subjects with replacement) is determined by \texttt{B}. For each bootstrap sample a random forest with \texttt{R} trees is built, which defaults to \texttt{R=10}. The bias induced by using smaller bootstrap ensemble sizes is corrected for in the estimate. Using \texttt{se=TRUE} will influence summaries from the fitted model, such as providing approximate confidence intervals for partial dependence plots (when running \texttt{partdep_hrf}), and give standard errors for predictions when \texttt{predict_hrf} is used.

All arguments (except those describing the data \texttt{x}, \texttt{yindx}, \texttt{time} and \texttt{id}) can be set in the \texttt{control} list. The arguments supplied in \texttt{control} are used if both are supplied. So if \texttt{ntrees=300} and \texttt{control=list(ntrees=500)} then 500 trees are fit. Besides the arguments described above, a number of other parameters can be set in \texttt{control}. These are: \texttt{nodi} giving the minimum number of training observations in a terminal node; \texttt{sample_fraction} giving the fraction of data sample to train each tree; \texttt{dtry} the number of sampled delta values used when splitting based on a historical variable (note this is alternatively controlled by the argument \texttt{nsamp} above; \texttt{ndelta} the number of delta values to use if \texttt{delta} is not supplied, these are taken as the quantiles from the distribution of observed delta values in the data; \texttt{qtry} the number of sampled values of \( \eta_2 \) for \texttt{method=freq/frc}, \( \eta_3 \) for \texttt{method=freqw/frcw}; \texttt{quantiles} is a vector of probabilities, and is used when \texttt{method=frc} or \texttt{method=frcw}, i.e. when covariate histories are split on their windowed empirical distributions. Splitting is restricted to these quantiles.

**Value**

Returns a list with elements, the most important being: \texttt{h} is a list returned by \texttt{parLapply} for fitting trees in parallel, \texttt{error} gives the OOB error (mse for regression, misclassification rate for classification), \texttt{control} a list containing control arguments, \texttt{boot} gives bootstrapped model estimates (if \texttt{se=TRUE}), \texttt{x, id, time} give the training data.

**Author(s)**

Joe Sexton <joesexton@gmail.com>

**References**


**See Also**

\texttt{predict_hrf, partdep_hrf, varimp_hrf}.
Examples

## Not run:

```r
# Mother's stress on child illness:
# Investigate whether mother's stress is (Granger) causal for child illness
# 'hrf' model is fit using previous observations of mother's stress to predict
# child's illness at given time point, but not mother's stress at that time point
# # Predictor variables are classified into "historical" and "concurrent"
# # A predictor is "historical" if its prior realizations can be used to predict
# # the outcome.
# # A predictor is "concurrent" if its realization at the same timepoint as the outcome
# # can be used to predict the outcome at that timepoint
# # A predictor can be both "concurrent" and "historical", the status of the predictors
# # can be set by the 'vh' and 'vc' arguments of 'hrf'.
# # (if not set these are automatically determined)
# # # -----------------------------------------------

data(mscm)
mscm=as.data.frame(na.omit(mscm))

# -- set concurrent and historical predictors
historical_predictors=match(c("stress","illness"),names(mscm))
concurrent_predictors=which(names(mscm)!="stress")

control=list(vh=historical_predictors,vc=concurrent_predictors)

# -- fit historical random forest
# (NOTE: response is 0/1 so a regression tree is
# the same as a classification tree with Gini-index splitting)
ff=hrf(x=mscm,id=mscm$id,time=mscm$day,yindx="illness",control=control)

# out-of-bag error
plot(1:length(ff$error),ff$error,type="l",main="OOB error",xlab="forest size",ylab="mse")

# .. larger nodesize works slightly better
control$nodesize=20
ff=hrf(x=mscm,id=mscm$id,time=mscm$day,yindx="illness",control=control)
points(1:length(ff$error),ff$error,type="l",col="blue")

# -- variable importance table
vi=varimp_hrf(ff)
vi
```
# -- fit historical random forest with 'se=TRUE'
control$se=TRUE
ff=hrf(x=mscm,id=mscm$id,time=mscm$day,yindex="illness",control=control)

# -- partial dependence for top 4 predictors (with +/-2 SE estimates)
par(mfrow=c(2,2))
for(k in 1:4)
  pd=partdep_hrf(ff,xindex=as.character(m$Predictor[k]))
par(mfrow=c(1,1))

### -- Classification trees

### setting classify=TRUE builds classification tree (gini-impurity node splitting)
control$classify=TRUE
### ... standard error estimation not implemented .. turn off bootstrapping
control$se=FALSE

ff=hrf(x=mscm,id=mscm$id,time=mscm$day,yindex="illness",control=control)

# -- plot oob classification error
plot(1:length(ff$error),ff$error,type="l",xlab="forest size",ylab="oob classification error")
abline(mean(mscm$illness),0,lty=2)  # error of constant model

p=predict_hrf(ff)

### variable importance table (model error measured by gini-impurity)
vi=varimp_hrf(ff)

vi

# ---------------------------------- #
# Data w/irregular observation times
# ---------------------------------- #
data(cd4)

control=list(se=TRUE)
ff=hrf(x=cd4,id=cd4$id,time=cd4$time,yindex="count",control=control)

vi=varimp_hrf(ff)

# -- partial dependence for top 4 predictors (with +/-2 SE estimates)
par(mfrow=c(2,2))
for(k in 1:4)
  pd=partdep_hrf(ff,xindex=as.character(m$Predictor[k]))
par(mfrow=c(1,1))
plot(1:length(ff$error),ff$error,xlab="forest size",ylab="oob mse",type="l")

## by default, the number of delta values (parameter 'eta_1' above) is 20
## can set this using 'ndelta'
control$ndelta=50

ccontrol$se=FALSE # -- turning off bootstrapping .
ff=hrf(x=cd4$id=time,control=control)
points(1:length(ff$error),ff$error,type="l",lty=2)

# the grid of delta values
ff$control$delta

# ----------------------------------------
# Boston Housing data (not longitudinal)
# ----------------------------------------
library(htree)
library(mlbench)
library(randomForest)

data(BostonHousing)
dat=as.data.frame(na.omit(BostonHousing))

## omitting arguments time/id assumes rows are iid
control=list(ntrees=500,sample_fraction=.5,nodesize=1)
h=hrf(x=dat,yindex="medv",control=control)

## randomForest comparison
## (by default, randomForest samples with replacement, while hrf samples without)
r=randomForest(medv~.,data=dat,replace=F,sampsize=ceiling(.5*nrow(dat)),nodesize=1)

## plot oob-error for both
plot(1:length(r$mse),r$mse,type="l",ylab=,xlab="forest size",ylab="out-of-bag mse")
points(1:length(h$error),h$error,type="l",col="blue")

## -- variable importance table
vi=varimp_hrf(h)
vi

## -- partial dependence plots with approximate 95
control$se=TRUE
h=hrf(x=dat,yindex="medv",control=control)

par(mfrow=c(2,2))
for(k in 1:4)
    pd=partdep_hrf(h,xindex=as.character(vi$predictor[k]))

par(mfrow=c(1,1))
htb

Tree boosting for longitudinal data

Description

Fits a boosted ensemble of historical regression trees to longitudinal data.

Usage

```r
htb(x, 
  time=NULL, 
  id=NULL, 
  yindx, 
  ntrees = 100, 
  method="freq", 
  nsplit=1, 
  lambda=.05, 
  family="gaussian", 
  cv.fold=0, 
  cv.rep=NULL, 
  nsamp=5, 
  historical=TRUE, 
  vh=NULL, 
  vc=NULL, 
  delta=NULL, 
  control=list())
```

Arguments

- **x**: A data frame containing response and predictors
- **time**: A vector of length `nrow(x)` of observation times
- **id**: A vector of subject identifiers (length equal to `nrow(x)`), if NULL observations are assumed independent
- **yindx**: Name of response variable, alt. its column number in `x`
- **ntrees**: Number of trees in ensemble
- **method**: Historical summary method, can be `freq`, `frac`, `mean0`, `freqw`, `fracw` and `mean0w`
htb

nsplit Number of splits in each regression tree.
lambda Shrinkage parameter applied to each tree.
family Either "gaussian" (default) or "bernoulli".
cv.fold Number of cross-validation folds, if cv.fold<=1 no cross-validation is run.
cv.rep Number of times to repeat the cross-validation. If not given set to cv.fold.
historical If TRUE then historical splitting is done, else standard splitting.
nsamp Number of sampled delta values, see below
vh Optional vector of indexes giving the historical predictors.
vc Optional vector of indexes giving strictly concurrent effects.
delta A vector of history lags to be used (see below), defaults to NULL in which case all possible observed lags are available for splitting.
control A list of control parameters (see below). All arguments, except those describing the data, can be set in control. Arguments in control are used if both are given.

Details

The htb function fits a boosted tree ensemble to longitudinal data. Data are assumed to be of form: \( z_{ij} = (y_{ij}, t_{ij}, x_{ij}) \) for \( i = 1, \ldots, n \) and \( j = 1, \ldots, n_i \), with \( y_{ij} \) being the response for the \( i \)-th subject at the \( j \)-th observation time \( t_{ij} \). The predictors at time \( t_{ij} \) are \( x_{ij} \). The number of observations can vary across subjects, and the sampling in time can be irregular.

htb estimates a model for the response \( y_{ij} \) using both \( (t_{ij}, x_{ij}) \) (the observations concurrent with \( y_{ij} \)) and all preceeding observations of the \( i \)-th subject up to (but not including) time \( t_{ij} \). The model is fit using historical regression (alt. classification) trees. Here a predictor is one of two types, either concurrent or historical. The concurrent predictors for \( y_{ij} \) are the elements of the vector \( (t_{ij}, x_{ij}) \), while a historic predictor is the set of all preceeding values (preceeding time \( t_{ij} \)) of a given element of \( (y_{ij}, t_{ij}, x_{ij}) \), for subject \( i \). In a historic regression tree, node splitting on a concurrent predictor follows the approach in standard regression (classification) trees. For historical predictors the splitting is modified since, associated with each observed response \( y_{ij} \), the set of observations of a historical predictor will vary according to \( i \) and \( j \). For these, the splitting is done by first transforming the preceeding values of a predictor using a summary function. This summary is invertible, in the sense that knowledge of it is equivalent to knowledge of the covariate history. Letting \( \tilde{z}_{ijk} \) denote the set of historical values of the \( k \)-th element of \( z_{ij} \), the summary function is denoted \( s(\eta; \tilde{z}_{ijk}) \) where \( \eta \) is the argument vector of the summary function. Node splitting based on a historical predictor is done by solving

\[
\text{argmin} \sum_{(ij) \in \text{Node}} (y_{ij} - \mu_L I(s(\eta; \tilde{z}_{ijk}) < c) - \mu_R I(s(\eta; \tilde{z}_{ijk}) \geq c))^2,
\]

where the minimization is over the vector \( (k, \mu, c, \eta) \). Each node of historical regression tree is split using the best split among all splits of concurrent and historical predictors.

Different summary functions are available in htb, specified by the argument method. Setting method="freq" corresponds the argument method. Setting method="freq" corresponds the summary

\[
s(\eta; \tilde{z}_{ijk}) = \sum_{h: t_{ih} - \eta_1 < t_{ih} < t_{ij}} I(z_{ihk} < \eta_2);
\]
method="frac": 
\[ s(\eta; \tilde{z}_{ijk}) = \sum_{h:t_{ij} - \eta_1 \leq t_{ih} < t_{ij}} I(z_{ikh} < \eta_2) / n_{ij}(\eta); \]

method="mean0": 
\[ s(\eta; \tilde{z}_{ijk}) = \sum_{h:t_{ij} - \eta_1 \leq t_{ih} < t_{ij}} z_{ikh} / n_{ij}(\eta); \]

method="freqw": 
\[ s(\eta; \tilde{z}_{ijk}) = \sum_{h:t_{ij} - \eta_1 \leq t_{ih} < t_{ij} - \eta_2} I(z_{ikh} < \eta_3); \]

method="fracw": 
\[ s(\eta; \tilde{z}_{ijk}) = \sum_{h:t_{ij} - \eta_1 \leq t_{ih} < t_{ij} - \eta_2} I(z_{ikh} < \eta_3) / n_{ij}(\eta); \]

method="meanw0": 
\[ s(\eta; \tilde{z}_{ijk}) = \sum_{h:t_{ij} - \eta_1 \leq t_{ih} < t_{ij} - \eta_2} z_{ikh} / n_{ij}(\eta). \]

Here \( n_{ij}(\eta) \) denotes the number of observations of subject \( i \) in the time window \([t_{ij} - \eta_1, t_{ij} - \eta_2)\). In the case \( n_{ij}(\eta) = 0 \), the summary function is set to zero, i.e \( s(\eta; \tilde{z}_{ijk}) = 0 \). The default is method="freqw". The possible values of \( \eta_1 \) in the summary function can be set by the argument delta. If not supplied, the set of possible values of \( \eta_1 \) is determined by the difference in time between successive observations in the data. When a split is attempted on a historical predictor, a sample of this set is taken from which the best split is selected. The size of this set equals that of the nsamp argument. See below on control for further arguments governing the historical splitting.

When cv.fold>1 then cross-validation is performed. In subsequent summaries of the model, say the partial dependence plots from partdep_htb, these cross-validation model fits are used to estimate the standard error. This is done using the delete-d jackknife estimator (where deletion refers to subjects, instead of rows of the training data). Each cross-validation model fit is performed by removing a random sample of 1/\( \text{cv.fold} \) of the subjects. The number of cross-validation runs is determined by cv.rep which defaults to the value of cv.fold.

All arguments (except those describing the data x, yindx, time and id) can be set in the control list. The arguments supplied in control are used if both are supplied, so if ntrees=300 and control=list(ntrees=500) then 500 trees are fit. Besides the arguments described above, a number of other parameters can be set in control. These are: nodesize giving the minimum number of training observations in a terminal node; sample_fraction giving the fraction of data sample to train each tree; dtry the number of sampled delta values used when splitting a variable based on a covariate history (note this is alternatively controlled by the argument nsamp above); ndelta the number of delta values to use if delta is not supplied, these are taken as the quantiles from the distribution of observed delta values in the data; qtry the number of sampled values of \( \eta_2 \) for method=freq/w, \( \eta_3 \) for method=freq/w/fracw; quantiles is a vector of probabilities, and is used when method=frac or method=fracw, ie when covariate histories are split on their windowed empirical distributions. Splitting is restricted to these quantiles. The fold-size for cross-validation can be set by control$cvfold, and the number of repetitions by control$nfold.
Value

Returns a list whose more important elements are: trees giving the tree ensemble, cv (if cv.fold>1) the cross-validation model estimates, cv_error cross-validated error (mse when family="gaussian" and negative log-likelihood when family="bernoulli"), control a list controlling the fit, x, id, time give the training data.

Author(s)

Joe Sexton <joesexton@gmail.com>

References


See Also

predict_htb, partdep_htb, varimp_htb.

Examples

```r
## Not run:

# --------------------------- #
# Mother's stress on child illness:
# Investigate whether mother's stress is (Granger) causal of child illness
# 'htb' model is fit using previous observations of mother's stress to predict
# child's illness at given time point, but not mother's stress at that time point
# # Predictor variables are classified into "historical" and "concurrent"
# # A predictor is "historical" if its prior realizations can be used to predict
# # the outcome.
# # A predictor is "concurrent" if its realization at the same timepoint as the outcome
# can be used to predict the outcome at that timepoint
# # A predictor can be both "concurrent" and "historical", the status of the predictors
# can be set by the 'vh' and 'vc' arguments of 'hrf'.
# (if not set these are automatically determined)
# # --------------------------- #

data(mscm)
mscm=as.data.frame(na.omit(mscm))
```
htb

# -- set concurrent and historical predictors
historical_predictors=match(c("illness","stress"),names(mscm))
concurrent_predictors=which(names(mscm)=="stress")
control=list(vh=historical_predictors,vc=concurrent_predictors,
ntrees=200,family="bernoulli",cvfold=10,lambda=.1)

# logistic regression
ff=htb(x=mscm,id=mscm$id,time=mscm$day,yindex="illness",control=control)

# cross-validated negative log-likelihood
plot(1:ff$ntrees,ff$cv_error,type="l",col="blue",
xlab="iterations",ylab="cross-validation error")

# -- variable importance table
vi=varimp_htb(ff)
vi

# -- plot partial dependence (with +/-2 approximate standard errors)
par(mfrow=c(2,2))
for(k in 1:4)
  partdep_htb(ff,xindx=as.character(vi$Predictor[k]))
par(mfrow=c(1,1))

# -- Standard errors are based on cross-validation runs (using delete-d
# (subjects) jackknife estimator, d="number-of-subjects"/cvfold)
# -- increasing nfold (which defaults to equal cvfold) gives less
# noisy standard error estimates ...
control$nfold=50
ff=htb(x=mscm$id,mscm$time,mscm$day,yindex="illness",control=control)

par(mfrow=c(2,2))
for(k in 1:4)
  partdep_htb(ff,xindx=as.character(vi$Predictor[k]))
par(mfrow=c(1,1))

# --------------------------------------------
# Data w/irregular observation times
# --------------------------------------------
data(cd4)
control=list(cvfold=10,lambda=.1,nsplit=3,ntrees=200)
ff=htb(x=cd4,id=cd4$id,time=cd4$time,yindex="count",control=control)

vi=varimp_htb(ff)

# -- partial dependence for top 4 predictors (with +/-2 SE estimates)
par(mfrow=c(2,2))
for(k in 1:4)
pd=partdep_htb(ff,xindex=as.character(vi$Predictor[k]))
par(mfrow=c(1,1))

# -- cv error
plot(1:ff$control$ntrees,ff$cv_error,xlab="boosting iteration",
ylab="cross-validated mse",type="l")
# estimated boosting iteration
abline(v=ff$nboost_cv,lt=2)
## by default, the number of delta values (parameter 'eta_1' above) is 20
## can set this using 'ndelta'
control$ndelta=50

ff=htb(x=cd4$id,cd4$time,yindex="count",control=control)
points(1:ff$control$ntrees,ff$cv_error,type="l",lt=2)
## note: differences in cv_error can be due (in part) to randomness
## in out-of-sample selection

# the grid of delta values
ff$control$delta

# ----------------------------------------------- #
# Boston Housing data (not longitudinal)
# ----------------------------------------------- #
# library(htree)
library(mlbench)
data(BostonHousing)
dat=as.data.frame(na.omit(BostonHousing))

# omitting arguments 'time' and 'id' assumes rows are iid
control=list(cvfold=10,ntrees=500)
h=htb(x=dat,yindex="medv",control=control)

# -- plot cross-validated Mean-squared error --- #
plot(1:h$ntrees,h$cv_error,type="l",xlab="Boosting iterations",
ylab="MSE",main="Cross-validated Mean-squared error")

# -- variable importance table
vi=varimp_htb(h,nperm=20)
vi

# -- partial dependence of top 4 predictors with +/-2 S.E.
par(mfrow=c(2,2))
for(k in 1:4)
partdep_htb(h,xindex=as.character(vi$Predictor[k]))

par(mfrow=c(1,1))
internal helper functions

Description
Helper functions

Usage

hrf_boot(object, B=50, R=10)
predict_viaux(trees, x, oob, id, time, yindx, concurrent, nconcurrent,
 historic, nhistoric, delta, delta0, type, quantiles, nperm, rf=1, ncat=-10)
partdep(object, xindx, xlim=NULL, ngrid=100, ntrees=NULL,
 subsample=1, which.class=1, se=FALSE, cond=NULL)

Arguments

object       An object of class htree.
B            Number of bootstrap samples.
R            Ensemble size for each bootstrap sample.
trees       Matrix with trees.
x            Data matrix.
oob          OOB matrix.
id           Vector of subject id’s.
time         Vector of observation times.
yindx        Column index of response variable in x.
concurrent   Concurrent predictors.
nconcurrent  Number of concurrent predictors.
historic     Historic predictors.
nhistoric    Number of historic predictors.
delta        Delta values.
delta0       Delta0 values.
type         Type of historical summary function.
quantiles    Quantiles to be used in summary function, if applicable.
nperm        Number of permutations for variable importance.
rf            Indicator for random forest model or not.
ncat  Number of categories.
xindx  Column index for partial dependence variable.
xlim  Range of values in plot.
ngrid  Number of grid points.
ntrees  Number of trees.
subsample  Sub-sample fraction.
which.class  Which class to plot partial dependence for.
se  Logical for whether to return standard errors.
cond  Logical subsetting.

Details

These functions are used internally by htree.

---

**mscm**  
*Mothers Stress Child Morbidity Data*

### Description

Mother stress and child sickness over period of around 30 days.

### Usage

```r
data(mscm)
```

### Format

```
mscm
```

### Source

The data is described in Zeger and Liang (1986). Mothers and infants were recruited by Professor Cheryl Alexander of the John Hopkins University Department of Maternal and Child Health from the Ambulatory Pediatric Health Services Clinic at Baltimore City Hospital. The mothers were asked to keep a diary of their own stress level and child illness. Baseline characteristics were obtained from a preliminary interview. Robins, Greenland, and Hu (1999) provide an analysis of the causal effect of mothers stress on child illness using this data.

The data was obtained from [https://faculty.washington.edu/heagerty/Books/AnalysisLongitudinal/datasets.html](https://faculty.washington.edu/heagerty/Books/AnalysisLongitudinal/datasets.html).
partdep_hrf

References


partdep_hrf Partial dependence

Description
Marginal effects for historical tree ensembles.

Usage

partdep_htb(object, xindx, xlim=NULL, ngrid=10, subsample=.5, plot.it=TRUE, cond=NULL)
partdep_hrf(object, x indx, xlim=NULL, ngrid=10, subsample=.5, plot.it=TRUE, which.class=1, cond=NULL)

Arguments

object An object of class htree.
xindx Name of predictor to compute marginal effect, alternatively the column corresponding this predictor in object$x.
xlim Optional range for partial dependence.
ngrid Number of values in grid for partial dependence.
subsample Fraction of training data sampled for calculation.
plot.it If TRUE then a plot is produced.
which.class Only used with partdep_hrf and when hrf is run with classify=TRUE. Determines for which class to show the partial dependence.
cond A logical condition for restricting the partial dependence, see below for details

Value

Returns x and y, the grid and partial dependence values. If htb run includes cross-validation then approximate standard errors are also output (based on the leave-out-m jack-knife estimate, where leave-out refers to subjects). If hrf estimation is run with se=TRUE then approximate bootstrap standard errors are returned (resampling subjects with replacement).
References


See Also

hrf, htb

Examples

```r
## Not run:

# library(htree)
data(mscm)
x=as.data.frame(na.omit(mscm))

# historical predictors
vh=c("stress","illness")

# concurrent predictors
vc=names(x)[-which(is.element(names(x),vh))]
control=list(vc=vc,vh=vh,nodesize=20)

### -- hrf
ff=hrf(x=x,yindx="illness",id=x$id,time=x$day,control=control)

### -- baseline illness
pp=partdep_hrf(ff,xindx="illness")

### -- with bootstrap standard errors
control$se=TRUE
ff=hrf(x=x,yindx="illness",id=x$id,time=x$day,control=control)

### -- baseline illness
pp=partdep_hrf(ff,xindx="illness")

### -- mothers stress
pp=partdep_hrf(ff,xindx="stress")

### -- partial dependence for a subset is done using the 'cond' argument

### ... only first week
pp=partdep_hrf(ff,xindx="illness",cond="day<=7")

### ... last week
pp=partdep_hrf(ff,xindx="illness",cond="day>23",plot.it=FALSE)
```
partdep_hrf

points(pp$x,pp$y,type="l",lwd=2,col="blue")

## ... first week and employed mothers
pp=partdep_hrf(ff,xindx="illness",cond="day<=7&emp==1")

### hbt -----  
# library(htree)
data(mscm)
x=as.data.frame(na.omit(mscm))

# historic predictors
vh=c("stress","illness")
# concurrent predictors
vc=names(x)[-which(is.element(names(x),vh))]

control=list(vc=vc,vh=vh,cvfold=10,family="bernoulli",ntrees=200,lambda=.1)
ff=htb(x=x,yindx="illness",id=x$id,time=x$day,control=control)

vn=c("illness","illness","day","stress")
par(mfrow=c(2,2))
for(k in vn)
  pp=partdep_hrb(ff,xindx=k)
  par(mfrow=c(1,1))

### --- standard error bands and model estimates
# library(htree)
sim_data=function(n=100,p=5,noise=2,sigma_e=.5,sigma_a=.5){
  ## -- random intercept data simulator
  random_intercept=as.numeric(mapply(rep,rnorm(n,sd=sigma_a),times=p))
  dat=data.frame(time=rep(1:p,n),x=(random_intercept+rnorm(n*p,sd=sigma_e)),
                 znoise=matrix(rnorm(n*p*noise),ncol=noise),id=sort(rep(1:n,p)))
  dat
}

## simulate data and estimate model and partial-dependence with standard errors
sdat=sim_data()
control=list(se=TRUE)
h=hrf(x=sdat,yindx="x",id=sdat$id,time=sdat$time,control=control)
pp=partdep_hrf(h,xindx="x",xlim=c(-2,2),ngrid=20)

## estimate and plot partial dependence on simulated data sets
p_est=NULL
nsim=10
for(s in 1:nsim){
  sdat=sim_data()
  hs=hrf(x=sdat,yindx="x",id=sdat$id,time=sdat$time)
  ps=partdep_hrf(hs,xindx="x",plot.it=FALSE,xlim=c(-2,2),ngrid=20)
}
predict_hrf

Description

Prediction functions for hrf and htb.

Usage

predict_htb(object, x=NULL, time=NULL, id=NULL, all.trees=FALSE, type="response", ntrees=NULL, se=FALSE)
predict_hrf(object, x=NULL, time=NULL, id=NULL, all.trees=FALSE, se=FALSE)

Arguments

object An object of class htree.
x A data frame or matrix containing new data. If NULL then training data in object is used.
time A vector of observation times.
id A vector of length nrow(x) identifying the subjects.
type If response then predictions on same scale as response are given.
ntrees The number of trees to use in prediction.
all.trees If TRUE then predictions associated with each tree are returned.
se If TRUE then standard errors of predictions are returned.

Value

Returns predictions.
**predict_hrf**

See Also

htb, hrf

Examples

```r
## Not run:

# Simulated data example
library(htree)
p=5; sigma_e=.5; sigma_a=.5; n=500; pnoise=2
random_intercept=as.numeric(mapply(rep,rnorm(n,sd=sigma_a),times=p))
dat=data.frame(time=rep(1:p,n),
               x=(random_intercept+rnorm(n*p,sd=sigma_e)),
               znoise=matrix(rnorm(n*p*pnoise),ncol=pnoise))

# fit historical random forest
hb=hrf(x=dat,time=dat$time,id=id,yindx=2,se=TRUE)

# get predictions with standard errors
pred=predict_hrf(hb,se=TRUE)

## Comparison of SE-estimates with actual standard errors for 'hrf'

## Evaluation points
n=200
datp=data.frame(y=rep(0,n),w=seq(-2,2,length=n),z=rep(0,n))

## -- estimate model on 50 simulated data sets
pred=NULL
pred_se=NULL
nsim=20

## -- B=100 bootstrap samples, ensemble size of R=10 on each
control=list(ntrees=500,B=100,R=10,se=TRUE,nodesize=5)
for(k in 1:nsim){
  if(is.element(k,seq(1,nsim,by=10))){
    cat(paste("simulation: ",k," of ",nsim," 
",sep=""))
    # -- simulation model -- #
    dat=data.frame(y=(4*datp$w+rnorm(n)),x=datp$w,z=rnorm(n))
    # ------------------------- #
    h=hrf(x=datp$w+y",control=control)
    mm=predict_hrf(object=h,x=datp,se=TRUE)
    pred=cbind(pred,mm[,1])
  }
}
```
```R
pred_se=cbind(pred_se,mm[,2])
}

# --- Actual Standard errors at datp
pred_se_true=apply(pred,1,sd)

# --- Mean of estimated standard errors
pred_se_est=apply(pred_se,1,mean)
pred_se_lower=apply(pred_se,1,quantile,prob=.1)
pred_se_upper=apply(pred_se,1,quantile,prob=.9)

# -- Plot estimated SE and true SE (+smooth)
z=c(pred_se_true,pred_se_est,pred_se_lower,pred_se_upper)
ylim=c(min(z),max(z))
plot(datp$w,pred_se_est,ylim=ylim,col="blue",xlab="w",ylab="Standard error",type="l",main="SE-true (red) SE-est (blue)"
points(datp$w,pred_se_lower,col="blue",type="l",lty=2)
points(datp$w,pred_se_upper,col="blue",type="l",lty=2)
points(datp$w,pred_se_true,col="red",type="l")

# ------------------------------------------------------------------ #
# Comparison of SE-estimates with actual standard errors for 'htb'
# ------------------------------------------------------------------ #

### -- evaluation points
n=200
datp=data.frame(y=rep(0,n),w=seq(-2,2,length=n),z=rep(0,n))

### -- estimate model on 50 simulated data sets
pred=NULL
pred_se=NULL
nsim=20
for(k in 1:nsim){
  if(is.element(k,seq(1,nsim,by=10)))
    cat(paste("simulation: ",k," of ",nsim," 
"
sep=""))
  # -- simulation model -- #
  dat=data.frame(y=4*datp$w+rnorm(n)),x=datp$w,z=rnorm(n)
  # ------------------------ #
  h=htb(x=dat,yindex="y",ntrees=200,cv.fold=10)
  mm=predict_htb(object=h,x=datp,se=TRUE)
pred=cbind(pred,mm[,1])
pred_se=cbind(pred_se,mm[,2])
}
```
# --- Actual Standard errors at datp
pred_se_true=apply(pred, 1, sd)

# --- Mean of estimated standard errors
pred_se_est=apply(pred, 1, mean)
pred_se_lower=apply(pred, 1, quantile, prob=.1)
pred_se_upper=apply(pred, 1, quantile, prob=.9)

# -- Plot estimated SE and true SE (+smooth)
z=c(pred_se_true, pred_se_est, pred_se_lower, pred_se_upper)
ylim=c(min(z), max(z))
plot(dat$p, pred_se_est, ylim=ylim, col="blue", xlab="w", ylab="Standard error", type="l", main="SE-true (red) SE-est (blue)"
points(dat$p, pred_se_lower, col="blue", type="l", lty=2)
puntos(dat$p, pred_se_upper, col="blue", type="l", lty=2)
points(dat$p, pred_se_true, col="red", type="l")

## End(Not run)

---

**varimp_hrf**

<table>
<thead>
<tr>
<th>Variable importance</th>
</tr>
</thead>
</table>

**Description**

Z-score variable importance for hrf and htb

**Usage**

varimp_hrf(object, nperm=20, parallel=TRUE)
varimp_htb(object, nperm=20)

**Arguments**

- **object**
  - Return list from hrf or htb
- **nperm**
  - Number of permutations.
- **parallel**
  - If TRUE, run in parallel.
Details

To measure the importance of a predictor, `varimp_hrf` and `varimp_htb` compare the prediction errors of the estimated model with the prediction errors obtained after integrating the predictor out of the model. If $F$ denotes the estimated model, the model obtained by integrating out predictor $k$ is $F_k(x) = \int F(x) dP(x_k)$, where $P(x_k)$ is the marginal distribution of $x_k$. In practice, the integration is done by averaging over multiple predictions from $F$, each obtained using a random permutation of the observed values of $x_k$. The number of permutations is set by `nperm`. Letting $L(y, y_{\hat{y}})$ be the loss of predicting $y$ with $y_{\hat{y}}$, the vector $w_i = L(y_i, F_k(x_i)) - L(y_i, F(x_i))$ for $i = 1, \ldots, n$ gives the difference in the prediction error between the original and marginalized model. The corresponding z-score $z = \text{mean}(w_i)/\text{se}(w_i)$ corresponds a paired test for the equality of the prediction errors, in which case it is approximately distributed as N(0,1). Larger z-score values indicate that the prediction error increases if $x_k$ is marginalized out, and thus that $x_k$ is useful. On the other hand, large negative values of the z-score indicate that the integrated model is more accurate. For longitudinal data, the $w_i$ are computed by averaging across all observations from the i-th subject. For `htb` the prediction error is calculated based on the cross-validation model estimates, for `hrf` out-of-bag predictions are used.

Value

A data.frame with columns: Predictor giving predictor being marginalized; Marginalized error gives the prediction error of model with Predictor marginalized out; Model error the prediction error with original model; Relative change gives relative change in prediction error due to marginalization; Z-value: Z value from test comparing prediction errors of original and marginalized models.

References


See Also

`hrf`, `htb`

Examples

```r
# Not run:

data(mscm)
mscm=as.data.frame(na.omit(mscm))

# -- set concurrent and historical predictors
historical_predictors=match(c("stress","illness"),names(mscm))
concurrent_predictors=which(names(mscm)!="stress")
control=list(vh=historical_predictors,vc=concurrent_predictors,nodesize=20)

# -- fit model
ff=hrf(x=mscm,id=mscm$id,time=mscm$day,yindx="illness",control=control)

# -- variable importance table
```
```r
# same with htb

control=list(vh=historical_predictors, vc=concurrent_predictors, lambda=.1, ntrees=200, nsplit=3, family="bernoulli")
control$cvfold=10  # need cross-validation runs to run varimp_htb
ff=htb(x=mscm, id=mscm$id, time=mscm$day, yindex="illness", control=control)

# -- variable importance table
vi=varimp_htb(ff)
vi

# boston housing data
library(mlbench)
data(BostonHousing)
dat=as.data.frame(na.omit(BostonHousing))
dat$chas=as.numeric(dat$chas)

# -- random forest
h=hrf(x=dat, yindex="medv")

# -- tree boosting
hb=htb(x=dat, yindex="medv", ntrees=1000, cv.fold=10, nsplit=3)

# -- Comparison of variable importance Z-scores and Z-scores from linear model
vi=varimp_hrf(h)
vb=varimp_htb(hb)
dvi=data.frame(var=as.character(vi$Predictor), Z_hrf=vi$Z)
dvb=data.frame(var=as.character(vb$Predictor), Z_htb=vb$Z)

dlm=summary(lm(medv~., dat))$coeffi
dlm=data.frame(var=rownames(dlm), Z_lm=round(abs(dlm[,3]),3))
dlm=merge(dlm[-1,], dvi, by="var", all.x=TRUE)

# -- Z-scores of hrf and lm for predictor variables
merge(dlm, dvb, by="var", all.x=TRUE)
```
## End (Not run)
Index

*Topic **classif**
  predict_hrf, 20
*Topic **datasets**
  cd4, 2
  mscm, 16
*Topic **longitudinal**
  hrf, 2
  htb, 9
*Topic **nonparametric**
  hrf, 2
  htb, 9
*Topic **regression**
  predict_hrf, 20
*Topic **tree**
  hrf, 2
  htb, 9

cd4, 2

hrf, 2, 18, 21, 24
hrf_boot (misc), 15
htb, 9, 18, 21, 24

misc, 15
mscm, 16

partdep (misc), 15
partdep_hrf, 5, 17
partdep_hrb, 12
partdep_hrb (partdep_hrf), 17
predict_hrf, 5, 20
predict_hrb, 12
predict_hrb (predict_hrf), 20
predict_viaux (misc), 15

varimp_hrf, 5, 23
varimp_hrb, 12
varimp_hrb (varimp_hrf), 23