

Package: RTFBoost (via r-universe)

October 16, 2026

Title Robust Tree-Based Boosting for Functional Regression

Version 0.1.0

Description Implements a family of robust tree-based boosting methods for functional regression, including procedures based on least absolute deviation, M-estimation, and MM-estimation. The package also includes a non-robust tree-based boosting method introduced in Ju X and Salibian-Barrera M. (2024) [doi:10.1007/s00180-023-01364-2](https://doi.org/10.1007/s00180-023-01364-2). Functions are provided for model fitting and prediction in regression settings with functional predictors.

License GPL (>= 3)

Imports stats, rpart, lme4, splines, RobStatTM

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Xiaomeng Ju [aut, cre], Matias Salibian-Barrera [aut]

Maintainer Xiaomeng Ju <xiaomengju@uvic.ca>

Depends R (>= 3.5.0)

Repository <https://cran.r-universe.dev>

Date/Publication 2026-10-16 13:40:15 UTC

RemoteUrl <https://github.com/cran/RTFBoost>

RemoteRef HEAD

RemoteSha 8a306151279fb31b9950d727a6829a26b8ddae4b

Contents

fruitfly	2
GED	3
RTFBoost	3
RTFBoost.control	7
RTFBoost.predict	9
RTFBoost.validation	11
TREE	13
TREE.control	14
TREE.predict	15
Index	16

fruitfly	<i>Fruit fly data</i>
----------	-----------------------

Description

This data set consist of the eggs laid from 667 female fruit flies. This is a subset of the original dataset collected in Dr. Carey’s laboratory at University of California, Davis. A description of the experiment can be found in Carey et al.(1998). We processed the original data set available at <https://anson.ucdavis.edu/~mueller/data/medfly1000.html> to include the ones that lived past 30 days. The objective is to explore the relationship of egg laying pattern and longevity of fruit flies.

Usage

fruitfly

Format

- A list with 3 fields:
- eggs** a matrix of 667 by 30 representing the number of eggs laid at each day.
 - lifetime** an integer giving the lifetime (in days).
 - eggs30** an integer of the total number of eggs laid past 30 days. ...

Source

Carey, J.R., Liedo, P., Müller, H.G., Wang, J.L., Chiou, J.M. (1998). Relationship of age patterns of fecundity to mortality, longevity, and lifetime reproduction in a large cohort of Mediterranean fruit fly females. J. of Gerontology –Biological Sciences 53, 245-251.

GED

*German electricity prices and demand***Description**

This data set consist of electricity spot prices traded at the European Energy Exchange (EEX) in Leipzig and electricity demand reported by European Network of Transmission System Operators for Electricity from January 1st 2006 to September 30th 2008, excluding weekends and holidays. It was processed from its original version provided in the on-line supplementary materials of Liebl (2013).

Usage

GED

Format

A list with 3 fields:

price hourly evaluated electricity spot prices in 638 days, represented as a matrix of dimension 638 by 24.

demand daily average of electricity demand in 638 days, represented as a vector.

date dates represented as a Date object with each day displayed in the form of %d%b%y. ...

Source

Liebl, Dominik. "Modeling and forecasting electricity spot prices: A functional data perspective." The Annals of Applied Statistics 7.3 (2013): 1562-1592.

RTFBoost

*(Robust) tree-based functional boosting***Description**

This function implements tree-based boosting algorithms for functional regression, including TFBoost(LAD), TFBoost(LAD-M), and TFBoost(RR) for robust scalar-on-function regression, and TFBoost(L2) for L2 scalar-on-function regression.

Usage

```

RTFBoost(
  x.train,
  z.train = NULL,
  y.train,
  x.val,
  z.val = NULL,
  y.val,
  x.test = NULL,
  z.test = NULL,
  y.test = NULL,
  seed = NULL,
  grid,
  t.range,
  control = RTFBoost.control(),
  ...
)

```

Arguments

<code>x.train</code>	a matrix or data.frame containing observations of the functional predictor in the training data.
<code>z.train</code>	a matrix or data.frame containing values of the scalar predictors in the training data (optional, defaults to NULL).
<code>y.train</code>	a vector containing values of the scalar response in the training data.
<code>x.val</code>	a matrix or data.frame containing observations of the functional predictor in the validation data.
<code>z.val</code>	a matrix or data.frame containing values of the scalar predictors in the validation data (optional, required when <code>z.train</code> is provided, defaults to NULL).
<code>y.val</code>	a vector containing values of the scalar response in the validation data.
<code>x.test</code>	a matrix or data.frame containing observations of the functional predictor in the test data (required when <code>make.prediction</code> in <code>control</code> is TRUE).
<code>z.test</code>	a matrix or data.frame containing values of the scalar predictors in the test data (required when <code>make.prediction</code> in <code>control</code> is TRUE, and <code>z.train</code> and <code>z.val</code> are provided).
<code>y.test</code>	a vector containing values of the scalar response in the test data. (optional).
<code>seed</code>	a random seed for generating potential parameters for the trees (optional).
<code>grid</code>	a vector of time points at which <code>x.train</code> and <code>x.val</code> are evaluated.
<code>t.range</code>	a vector provided in the form of <code>c(a, b)</code> , where <code>a</code> and <code>b</code> are numerical values specifying the boundaries of the domain of the functional predictor.
<code>control</code>	a named list of control parameters as returned by RTFBoost.control .
<code>...</code>	optional additional arguments passed to <code>RTFBoost</code> , including components <code>user.loss</code> , <code>tree.init</code> , and <code>fitted.1st</code> .

`user.loss` is a list that specifies a user-defined loss to replace the L2 loss used by TFBoost(L2), including components `func` and `func.grad`, denoting the loss function and its derivative respectively;

`tree.init` is a user-provided object that has the same form as the one returned from [TREE](#). It is used as the tree to initialize the boosting estimator. `fitted.1st` is a list of fitted values for the first stage, required if `stage.no = 1` in control, including components `f.train`, `f.val`, and `f.test` denoting the predicted values for the training, validation, and test set respectively; `f.test` is not required if `make.prediction = FALSE` in control.

Details

This function implements tree-based boosting algorithms for functional regression. The algorithms use functional multi-index regression trees as the base learners, which are constructed using functions available in the `rpart` and `lme4` packages (see <https://arxiv.org/abs/2109.02989>).

Value

A list with the following components:

<code>loss.train</code>	a vector of training errors from all iterations
<code>loss.val</code>	a vector of validation errors from all iterations
<code>f.train</code>	a vector of predicted values for the training data at the early stopping iteration
<code>f.val</code>	a vector of predicted values for the validation data at the early stopping iteration
<code>f.test</code>	a vector of predicted values for the test data at the early stopping iteration (returned if <code>make.prediction = TRUE</code> in control)
<code>early.stop.s1</code>	the early stopping iteration for the 1st stage of TFBoost(LAD-M) and TFBoost(RR) (returned if <code>type = "RR"</code> or <code>type = "LAD-M"</code> in control)
<code>early.stop</code>	the early stopping iteration (for TFBoost(LAD-M) and TFBoost(RR), equals to <code>n.init</code> + the early stopping time in the 2nd stage)
<code>err.train</code>	a vector of mean squared prediction errors for training data from all iterations
<code>err.val</code>	a vector of mean squared prediction errors for validation data from all iterations
<code>err.test</code>	a vector of mean squared prediction errors for test data before and at the early stopping iteration (returned if <code>make.prediction = TRUE</code> in control)
<code>init.vals</code>	a constant (if <code>init.type = "mean"</code> or <code>init.type = "median"</code> in control) or a tree object (if <code>init.type = "LADTree"</code> in control) to initialize the boosting algorithm
<code>tree.objs</code>	a list of trees objects (returned if <code>save.tree = TRUE</code> in control, one per iteration and returned from TREE)
<code>alpha</code>	a vector of step sizes (one per iteration)
<code>save.f.train</code>	a matrix of predicted values for the training data from all iterations (returned if <code>save.f = TRUE</code> in control, one column per iteration)
<code>save.f.val</code>	a matrix of predicted values for the validation data from all iterations (returned if <code>save.f = TRUE</code> in control, one column per iteration)

`save.f.test` a matrix of predicted values for the test data before and at the early stopping iteration (returned if `save.f = TRUE` and `make.prediction = TRUE` in control, one column per iteration)

`control` control from the input arguments

Author(s)

Xiaomeng Ju, <xiaomengju@uvic.ca>

Examples

```
data(fruitfly)
n <- nrow(fruitfly$eggs)
n0 <- floor(0.2 * n)
set.seed(123)
idx_test <- sample(n, n0)
idx_train <- sample((1:n)[-idx_test], floor(0.6 * n))
idx_val <- (1:n)[-c(idx_test, idx_train)]
xtrain <- fruitfly$eggs[idx_train, ]
ytrain <- fruitfly$lifetime[idx_train]
xval <- fruitfly$eggs[idx_val, ]
yval <- fruitfly$lifetime[idx_val]
xtest <- fruitfly$eggs[idx_test, ]
ytest <- fruitfly$lifetime[idx_test]
gg <- 1:30
tt <- c(0, 30)
# For illustrative purposes only; in practice, more iterations (niter),
# directions (num.dir) may be required for convergence.
niter <- 10
tree.control <- TREE.control(tree.type = "B", max.depth = 1, num.dir = 20)

# LAD loss
my.control <- RTFBoost.control(make.prediction = TRUE, save.tree = TRUE,
                             tree.control = tree.control, shrinkage = 0.05,
                             nknot = 3, type = "LAD")
model_TFBoost <- RTFBoost(x.train = xtrain, y.train = ytrain,
                        x.val = xval, y.val = yval,
                        grid = gg, t.range = tt, niter = niter,
                        control = my.control)
predictions_LAD <- RTFBoost.predict(model_TFBoost, newx = xtest,
                                   grid = gg, t.range = tt)

# Robust TFBoost
my.control <- RTFBoost.control(make.prediction = TRUE, save.tree = TRUE,
                             tree.control = tree.control, shrinkage = 0.05,
                             nknot = 3, type = "RR")
model_RTFBoost <- RTFBoost(x.train = xtrain, y.train = ytrain,
                        x.val = xval, y.val = yval,
                        grid = gg, t.range = tt, niter = niter,
                        control = my.control)
predictions_RR <- RTFBoost.predict(model_RTFBoost, newx = xtest,
```

```
grid = gg, t.range = tt)
```

RTFBoost.control	<i>Tuning and control parameters for (robust) tree-based functional boosting algorithms</i>
------------------	---

Description

Allow users to set the tuning and control parameters for tree-based functional boosting algorithms, including TFBoost(LAD), TFBoost(LAD-M), and TFBoost(RR) for robust scalar-on-function regression, and TFBoost(L2) for L2 scalar-on-function regression.

Usage

```
RTFBoost.control(
  make.prediction = TRUE,
  type = "RR",
  eff = 0.95,
  bb = 0.5,
  init.type = "median",
  init.tree.params = list(tree.type = "A", tree.nindex = 1, max.depth = 1, num.dir = 20,
    min.leafsize = 10),
  loss.s2 = "tukey",
  tree.control = TREE.control(),
  shrinkage = 0.05,
  n.init = 100,
  niter = 100,
  nknot = 3,
  precision = 4,
  save.f = FALSE,
  trace = FALSE,
  save.tree = FALSE,
  error.type = "mse",
  trim.prop = NULL,
  trim.c = 3,
  stage.no = 0
)
```

Arguments

make.prediction	a logical flag: if TRUE the algorithm make predictions using <code>x.test</code> (defaults to TRUE).
type	a string that specifies the type of the RTFBoost algorithm. Valid options are "L2", "LAD", "LAD-M", or "RR", corresponding to the algorithms of TFBoost(L2), TFBoost(LAD), TFBoost(LAD-M), and TFBoost(RR) respectively (defaults to "RR").

eff	a numeric value between 0 and 1 indicating the efficiency (measured in a linear model with Gaussian errors) of Tukey's loss function used in the 2nd stage of TFBoost(LAD-M) and TFBoost(RR) (defaults to 0.95).
bb	a numeric value indicating the breakdown point of the M-scale estimator used in the 1st stage of TFBoost(RR) (defaults to 0.5).
init.type	a string indicating the initial estimator to be used. Valid options are: "mean", "median" or "LADTree" (defaults to 'median')
init.tree.params	a list of parameters for the initial LADtree with components <code>tree.type</code> , <code>tree.nindex</code> , <code>max.depth</code> , <code>num.dir</code> , and <code>min.leafsize</code> , where
loss.s2	a string indicating the loss function used in the M-step of TFBoost(LAD-M). Valid options are "huber" and "tukey" (defaults to "tukey"). <code>tree.type</code> is a character ("A" or "B") denoting Type A or Type B tree (defaults to "A"), <code>tree.nindex</code> is an integer specifying the number of indices required by the initial Type A tree (defaults to 1), <code>num.dir</code> is an integer specifying the number of random directions required by the initial Type B tree (defaults to 20), <code>max.depth</code> is an integer specifying the maximum depth for the initial Type A or Type B tree (defaults to 1), <code>min.leafsize</code> is an integer specifying the minimum number of observations per leaf node for the initial Type A or Type B tree (defaults to 10).
tree.control	a list of control parameters for the tree base learner of RTFBoost (defaults to <code>TREE.control()</code> , see TREE.control).
shrinkage	a numeric shrinkage parameter in boosting (defaults to 0.05).
n.init	an integer specifying the number of iterations for the 1st stage of TFBoost(RR) (defaults to 100).
niter	an integer specifying the number of iterations (for TFBoost(LAD-M) and TFBoost(RR) $T_{1,max} + T_{2,max}$) (defaults to 200)
nknot	an integer denoting the number of interior knots of the cubic B-spline basis (defaults to 3)
precision	an integer denoting the number of significant digits to keep when using validation error to calculate early stopping time (defaults to 4)
save.f	a logical flag: if TRUE save the function estimates at all iterations (defaults to FALSE)
trace	a logical flag: if TRUE print the number of completed iterations for monitoring progress (defaults to FALSE)
save.tree	a logical flag: if TRUE save the tree objects at all iterations, required when the user calls <code>RTFBoost.predict</code> with the returned object from <code>RTFBoost</code> (defaults to FALSE)
error.type	a character string indicating the type of error metrics to be evaluated on the test set. Valid options are: "mse" (mean squared error), "aad" (average absolute deviation), and "tmse" (trimmed mean squared error)

trim.prop	a numeric value used as the trimming proportion if "tmse" is used as the performance metric. "tmse" calculates the mean-square error of residuals (r) of which $ r < \text{quantile}(r , 1 - \text{trim_prop})$ (e.g. $\text{trim_prop} = 0.1$ ignores 10% of the data and calculates MSE of residuals whose absolute values are below 90% quantile of $ r $). If both trim.prop and trim.c are specified, trim.c will be used.
trim.c	a numeric value used for trimming if "tmse" is used as the performance metric (defaults to 3). "tmse" calculates the mean squared error of the residuals(r) between $\text{median}(r) + \text{trim_c} \text{mad}(r)$ and $\text{median}(r) - \text{trim_c} \text{mad}(r)$. If both trim.prop and trim.c are specified, trim.c will be used.
stage.no	a numeric value indicating which stages are executed, required by RTFBoost(LAD-M) and RTFBoost(RR): 0 runs both stages, 1 runs the first stage only, and 2 runs the second stage only (defaults to 0).

Details

Various tuning and control parameters for tree-based functional boosting algorithms implemented in the function [RTFBoost](#).

Value

A list of all input parameters

Author(s)

Xiaomeng Ju, <xiaomengju@uvic.ca>

RTFBoost.predict	<i>RTFBoost.predict</i>
------------------	-------------------------

Description

A function to make predictions and calculate test error given an object returned by TFBoost and test data

Usage

```
RTFBoost.predict(model, newx, newz = NULL, newy = NULL, grid, t.range)
```

Arguments

model	an object returned by RTFBoost
newx	a matrix or data.frame containing observations of the functional predictor in the new data
newz	a matrix or data.frame containing values of the scalar predictors in the new data (required if z.train and z.val were used to fit model, defaults to NULL).
newy	a vector containing values of the scalar response in the new data. (optional, defaults to NULL)

<code>grid</code>	a vector of time points at which <code>x.test</code> are evaluated.
<code>t.range</code>	a vector provided in the form of <code>c(a, b)</code> , where <code>a</code> and <code>b</code> are numerical values specifying the boundaries of the domain of the functional predictor.

Details

A function to make predictions and calculate test error given an object returned by TFBoost and test data

Value

A list with with the following components:

<code>f.new</code>	predicted values with model at the early stopping iteration using <code>x_test</code> (or <code>x_test</code> and <code>z_test</code>) as the predictors
<code>err.new</code>	a vector of test errors before and at the early stopping iteration (returned if <code>newy</code> is provided)
<code>save.f.new</code>	a matrix of test function estimates at all iterations (returned if <code>save_f = TRUE</code> in control)

Author(s)

Xiaomeng Ju, <xiaomengju@uvic.ca>

Examples

```
data(fruitfly)
n <- nrow(fruitfly$eggs)
n0 <- floor(0.2 * n)
set.seed(123)
idx_test <- sample(n, n0)
idx_train <- sample((1:n)[-idx_test], floor(0.6 * n))
idx_val <- (1:n)[-c(idx_test, idx_train)]
xtrain <- fruitfly$eggs[idx_train, ]
ytrain <- fruitfly$lifetime[idx_train]
xval <- fruitfly$eggs[idx_val, ]
yval <- fruitfly$lifetime[idx_val]
xtest <- fruitfly$eggs[idx_test, ]
ytest <- fruitfly$lifetime[idx_test]
gg <- 1:30
tt <- c(0, 30)
# For illustrative purposes only; in practice, more iterations (niter),
# directions (num.dir) may be required for convergence.
niter <- 10
tree.control <- TREE.control(tree.type = "B", max.depth = 1, num.dir = 20)

# LAD loss
my.control <- RTFBoost.control(make.prediction = TRUE, save.tree = TRUE,
                             tree.control = tree.control, shrinkage = 0.05,
                             nknot = 3, type = "LAD")
model_TFBoost <- RTFBoost(x.train = xtrain, y.train = ytrain,
```

```

        x.val = xval, y.val = yval,
        grid = gg, t.range = tt, niter = niter,
        control = my.control)
predictions_LAD <- RTFBoost.predict(model_TFBoost, newx = xtest,
                                   grid = gg, t.range = tt)

# Robust TFBoost
my.control <- RTFBoost.control(make.prediction = TRUE, save.tree = TRUE,
                              tree.control = tree.control, shrinkage = 0.05,
                              nknot = 3, type = "RR")
model_RTFBoost <- RTFBoost(x.train = xtrain, y.train = ytrain,
                          x.val = xval, y.val = yval,
                          grid = gg, t.range = tt, niter = niter,
                          control = my.control)
predictions_RR <- RTFBoost.predict(model_RTFBoost, newx = xtest,
                                   grid = gg, t.range = tt)

```

RTFBoost.validation	<i>Robust tree-based functional boosting with initialization parameters chosen on a validation set</i>
---------------------	--

Description

A function to fit TFBoost(RR) (see also [RTFBoost](#)) where the initialization parameters are chosen based on the performance on the validation set.

Usage

```

RTFBoost.validation(
  x.train,
  z.train = NULL,
  y.train,
  x.val,
  z.val = NULL,
  y.val,
  x.test,
  z.test = NULL,
  y.test,
  seed = NULL,
  grid,
  t.range,
  max.depth.init.set = c(1, 2, 3, 4),
  min.leafsize.init.set = c(10, 20, 30),
  control = RTFBoost.control(),
  save.all = TRUE
)

```

Arguments

<code>x.train</code>	a matrix or data.frame containing observations of the functional predictor in the training data.
<code>z.train</code>	a matrix or data.frame containing values of the scalar predictors in the training data (optional, defaults to NULL).
<code>y.train</code>	a vector containing values of the scalar response in the training data.
<code>x.val</code>	a matrix or data.frame containing observations of the functional predictor in the validation data.
<code>z.val</code>	a matrix or data.frame containing values of the scalar predictors in the validation data (optional, required when <code>z.train</code> is provided, defaults to NULL).
<code>y.val</code>	a vector containing values of the scalar response in the validation data.
<code>x.test</code>	a matrix or data.frame containing observations of the functional predictor in the test data (required when <code>make.prediction.in.control</code> is TRUE).
<code>z.test</code>	a matrix or data.frame containing values of the scalar predictors in the test data (required when <code>make.prediction.in.control</code> is TRUE, and <code>z.train</code> and <code>z.val</code> are provided).
<code>y.test</code>	a vector containing values of the scalar response in the test data. (optional).
<code>seed</code>	a random seed for generating potential parameters for the trees (optional).
<code>grid</code>	a vector of time points at which <code>x.train</code> and <code>x.val</code> are evaluated.
<code>t.range</code>	a vector provided in the form of <code>c(a, b)</code> , where <code>a</code> and <code>b</code> are numerical values specifying the boundaries of the domain of the functional predictor.
<code>max.depth.init.set</code>	a vector of possible values of the maximum depth of the initial LADTree that the algorithm choses from (defaults to <code>c(1,2,3,4)</code>).
<code>min.leafsize.init.set</code>	a vector of possible values of the minimum observations per node of the initial LADTree that the algorithm choses from (defaults to <code>c(10,20,30)</code>).
<code>control</code>	a named list of control parameters as returned by RTFBoost.control .
<code>save.all</code>	a logical flag: if TRUE the algorithm saves the predictions for median initialization and LADTree initialization with every combination of <code>max.depth.init.set</code> and <code>min.leaf.size.init.set</code> (defaults to TRUE).

Details

This function runs the TFBoost(RR) algorithm (see [RTFBoost](#)) on different combinations of the parameters for the initial fit, and chooses the optimal set based on the performance on the validation set.

Value

A list with the following components:

<code>model.best</code>	an object returned by RTFBoost that is trained with selected initialization parameters
-------------------------	--

<code>param</code>	a vector of selected initialization parameters (return (0,0) if selected initialization is the median of the training responses)
<code>errs.val</code>	a vector of robust prediction errors for validation data from all iterations
<code>errs.test</code>	a vector of robust prediction errors for test data from all iterations before and at the early stopping iteration (returned if <code>make.prediction = TRUE</code> in control)
<code>pred.list.train</code>	a list of predicted values for the training data at the early stopping time (returned if <code>save.all = TRUE</code> , one element per combination in the order of median initialization, and then <code>expand.grid(sort(min.leafsize.init.set, TRUE), max_depths= sort(max.depth.init.set))</code>)
<code>pred.list.val</code>	a list of predicted values for the validation data at the early stopping time (returned if <code>save.all = TRUE</code> , one column per combination)
<code>pred.list.test</code>	a list of predicted values for the test data at the early stopping time (returned if <code>save.all = TRUE</code> and <code>make.prediction = TRUE</code> in control, one element per combination)

 TREE

Functional multi-index tree

Description

This function implements a algorithm for functional multi-index tree.

Usage

```
TREE(x, y, z, random.seed, control = TREE.control())
```

Arguments

<code>x</code>	a matrix or data.frame containing basis projections of the functional predictor in the training data.
<code>y</code>	a vector containing values of the scalar response in the training data (optional, defaults to NULL).
<code>z</code>	a matrix or data.frame containing values of the scalar predictors in the training data (optional, defaults to NULL).
<code>random.seed</code>	an integer as the seed to generate random directions for the Type B tree.
<code>control</code>	a named list of control parameters, as returned by TREE.control

Details

This function implements a functional multi-index tree algorithm developed based on functions available in the `rpart` package.

Value

A list with the following components:

beta.opt	a matrix of index coefficients estimated by the Type A tree, returned if control\$tree.type = "A"
betas.selected	a matrix of index coefficients selected by the Type B tree, returned if control\$tree.type = "B"
pred	a vector of predicted values for the training data
tree.model	an rpart object of the fitted tree
control	control from the input arguments

Author(s)

Xiaomeng Ju, <xiaomeng.ju@stat.ubc.ca>

TREE.control	<i>Tuning and control parameters for the functional multi-index tree</i>
--------------	--

Description

Tuning and control parameters for the functional multi-index tree.

Usage

```
TREE.control(
  tree.type = "A",
  tree.nindex = 1,
  max.depth = 1,
  num.dir = 20,
  min.leafsize = 2,
  nmulti = 3,
  nscreen = 30
)
```

Arguments

tree.type	a character ("A" or "B") denoting Type A or Type B tree (defaults to "A").
tree.nindex	an integer specifying the number of indices required by a Type A tree (defaults to 1).
max.depth	an integer specifying the maximum depth for a Type A or Type B tree (defaults to 1).
num.dir	an integer specifying the number of random directions required by the initial Type B tree (defaults to 20).
min.leafsize	an integer specifying the minimum number of observations per leaf node for a Type A or Type B tree (defaults to 2).

<code>nmulti</code>	an integer specifying the number of random initialization points to fit a Type A tree (defaults to 3).
<code>nscreen</code>	an integer specifying the number of random points from which to select the initial points to fit a Type A tree (defaults to 30).

Details

Various tuning and control parameters for the functional multi-index tree algorithm implemented in the function [TREE](#)

Value

A list of all input parameters

Author(s)

Xiaomeng Ju, <xiaomeng.ju@stat.ubc.ca>

<code>TREE.predict</code>	<i>TREE.predict</i>
---------------------------	---------------------

Description

A function to make predictions given an object returned by `TREE` and test data

Usage

```
TREE.predict(model, newx, newz)
```

Arguments

<code>model</code>	an object returned by <code>TREE</code>
<code>newx</code>	a matrix or data.frame containing basis projections of the functional predictor in the new data.
<code>newz</code>	a matrix or data.frame containing values of the scalar predictors in the new data (optional, defaults to <code>NULL</code> , required if <code>z</code> was used to fit <code>model</code>).

Details

A function to make predictions given an object returned by `TREE` and test data

Value

`pred` a vector of predicted values using `newx` (or `newx` and `newz`) as the predictors

Author(s)

Xiaomeng Ju, <xmengju@stat.ubc.ca>

Index

* datasets

fruitfly, [2](#)

GED, [3](#)

fruitfly, [2](#)

GED, [3](#)

RTFBoost, [3](#), [9](#), [11](#), [12](#)

RTFBoost.control, [4](#), [7](#), [12](#)

RTFBoost.predict, [9](#)

RTFBoost.validation, [11](#)

TREE, [5](#), [13](#), [15](#)

TREE.control, [8](#), [13](#), [14](#)

TREE.predict, [15](#)