Package: nftbart (via r-universe)

October 18, 2026

Type Package

Title Nonparametric Failure Time Bayesian Additive Regression Trees

Version 2.3

Date 2025-12-02

Author Rodney Sparapani [aut, cre], Robert McCulloch [aut], Matthew Pratola [ctb], Hugh Chipman [ctb]

Maintainer Rodney Sparapani <rsparapa@mcw.edu>

Description Nonparametric Failure Time (NFT) Bayesian Additive Regression Trees (BART): Time-to-event Machine Learning with Heteroskedastic Bayesian Additive Regression Trees (HBART) and Low Information Omnibus (LIO) Dirichlet Process Mixtures (DPM). An NFT BART model is of the form Y = mu + f(x) + sd(x) E where functions f and sd have BART and HBART priors, respectively, while E is a nonparametric error distribution due to a DPM LIO prior hierarchy. See the following for a description of the model at <doi:10.1111/biom.13857>.

License GPL (>= 2)

Depends R (>= 4.2.0), survival, nnet, lattice

Imports Rcpp

LinkingTo Rcpp

NeedsCompilation yes

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

Date/Publication 2025-12-03 08:47:08 UTC

RemoteUrl https://github.com/cran/nftbart

RemoteRef HEAD

RemoteSha 2ca4ed21be01d7d3a350fd0f99c5fd77feb5b09e

Contents

bartModelMatrix 2

bMM . . . 3
bmx . . . 5
CDCheight . . . 6
CDimpute . . . 6
Cindex . . . 7
lung . . . 8
nft2 . . . 9
predict.aftree . . . 15
predict.nft2 . . . 17
tsvs2 . . . 20
xicuts . . . 24

Index **26**

`bartModelMatrix` *Deprecated: use bMM instead*

Description

Create a matrix out of a vector or `data.frame`. The compiled functions of this package operate on matrices in memory. Therefore, if the user submits a vector or data.frame, then this function converts it to a matrix. Also, it determines the number of cutpoints necessary for each column when asked to do so.

Usage

```
bartModelMatrix(X, numcut=0L, usequants=FALSE, type=7, rm.const=FALSE,
                cont=FALSE, xicuts=NULL, rm.vars=NULL)
```

Arguments

`X`	A vector or data.frame to create the matrix from.
`numcut`	The maximum number of cutpoints to consider. If `numcut=0`, then just return a matrix; otherwise, return a list.
`usequants`	If `usequants` is `FALSE`, then the cutpoints in `xinfo` are generated uniformly; otherwise, if `TRUE`, quantiles are used for the cutpoints.
`type`	Determines which quantile algorithm is employed.
`rm.const`	Whether or not to remove constant variables.
`cont`	Whether or not to assume all variables are continuous.
`xicuts`	To specify your own cut-points, use the `xicuts` argument.
`rm.vars`	The variables that you want removed.

Value

If numcut==0 (the default), then a matrix of the covariates is returned; otherwise, a list is returned with the following values.

X	A matrix of the covariates with n rows and p columns.
numcut	A vector of length p of the number of cut-points for each covariate.
grp	A vector that corresponds to variables in the input data.frame that were translated into dummy columns in the output matrix, i.e., for each input variable in order, there is a number in the vector corresponding to the number of output columns created for it.

See Also

bMM

Examples

```
## set.seed(99)

## a <- rbinom(10, 4, 0.4)

## table(a)

## x <- runif(10)

## df <- data.frame(a=factor(a), x=x)

## (b <- bartModelMatrix(df))

## (b <- bartModelMatrix(df, numcut=9))

## (b <- bartModelMatrix(df, numcut=9, usequants=TRUE))

## Not run:
    ## this is an error
    ## f <- bartModelMatrix(as.character(a))

## End(Not run)
```

---

bMM *Create a matrix out of a vector or data.frame*

Description

Adapted from bartModelMatrix(). The compiled functions of this package operate on matrices in memory. Therefore, if the user submits a vector or data.frame, then this function converts it to a matrix. Also, it determines the number of cutpoints necessary for each column when asked to do so.

Usage

```
bMM(X, numcut=0L, usequants=FALSE, type=7, xicuts=NULL, rm.const=FALSE,
    rm.dupe=FALSE, method="spearman", use="pairwise.complete.obs")
```

Arguments

X	A vector or data.frame to create the matrix from.
numcut	The maximum number of cutpoints to consider. If numcut=0, then just return a matrix; otherwise, return a list.
usequants	If usequants is FALSE, then the cutpoints in xinfo are generated uniformly; otherwise, if TRUE, quantiles are used for the cutpoints.
type	Determines which quantile algorithm is employed.
xicuts	To specify your own cut-points, use the xicuts argument.
rm.const	To remove constant variables or not.
rm.dupe	To remove duplicate variables or not.
method, use	Correlation options.

Value

If numcut==0 (the default), then a matrix of the covariates is returned; otherwise, a list is returned with the following values.

X	A matrix of the covariates with n rows and p columns.
numcut	A vector of length p of the number of cut-points for each covariate.
grp	A vector that corresponds to variables in the input data.frame that were translated into dummy columns in the output matrix, i.e., for each input variable in order, there is a number in the vector corresponding to the number of output columns created for it.
dummy	Corresponds to grp with a two row matrix including the start column of each dummy group in row 1 and the end column in row 2.

See Also

xicuts

Examples

```
set.seed(99)

a <- rbinom(10, 4, 0.4)

table(a)

x <- runif(10)

df <- data.frame(a=factor(a), x=x)
```

```
(b <- bMM(df))

(b <- bMM(df, numcut=9))

(b <- bMM(df, numcut=9, usequants=TRUE))

## Not run:
    ## this is an error
    f <- bMM(as.character(a))

## End(Not run)
```

---

bmx *NHANES 1999-2000 Body Measures and Demographics*

Description

This data set was created from the National Health and Nutrition Examination Survey (NHANES) 1999-2000 Body Measures Exam and Demographics. To create growth charts, this data is restricted to 3435 children aged 2 to 17.

Usage

```
data(bmx)
```

Format

SEQN:	Sequence number
BMXHT:	Height in cm
RIAGENDR:	Gender: 1=male, 2=female
RIDAGEEX:	Age in years with fractions for months
RIDRETH2:	Race/ethnicity: 1=Non-Hispanic White, 2=Non-Hispanic Black, 3=Hispanic
BMXWT:	Weight in kg

References

National Health and Nutrition Examination Survey 1999-2000 Body Measures Exam. https://wwwn.cdc.gov/Nchs/Nhanes/1999-2000/BMX.htm

National Health and Nutrition Examination Survey 1999-2000 Demographics. https://wwwn.cdc.gov/Nchs/Nhanes/1999-2000/DEMO.htm

CDCheight *CDC height for age growth charts*

Description

Using the Cole and Green LMS method, here we provide percentiles of height by age and sex based on the US National Center for Health Statistics data for children aged 2 to 17.

Usage

```
data(CDCheight)
```

Format

age:	Age in years
sex:	1=male, 2=female
height.XXX:	Height XXXth percentile in cm

References

Cole, Timothy J and Green, Pamela J (1992) Smoothing reference centile curves: the LMS method and penalized likelihood. *Statistics in medicine*, **11**, 1305–1319.

The US Centers for Disease Control and Prevention stature by age LMS parameters https://www.cdc.gov/growthcharts/data/zscore/statage.csv

CDimpute *Cold-deck missing imputation*

Description

This function imputes missing data.

Usage

```
CDimpute(x.train, x.test=matrix(0, 0, 0), impute.bin=NULL)
```

Arguments

x.train	The training matrix.
x.test	The testing matrix, if given.
impute.bin	An index of the columns to avoid imputing which will be handled by BART internally.

Details

We call this method cold-decking in analogy to hot-decking. Hot-decking was a method commonly employed with US Census data in the early computing era. For a particular respondent, missing data was imputed by randomly selecting from the responses of their neighbors since it is assumed that the values are likely similar. In our case, we make no assumptions about which values may, or may not, be nearby. We simply take a random sample from the matrix rows to impute the missing data. If the training and testing matrices are the same, then they receive the same imputation.

Value

`x.train`	The imputed training matrix.
`x.test`	The imputed testing matrix.
`miss.train`	A summary of the missing variables for training.
`miss.test`	A summary of the missing variables for testing.
`impute.flag`	Whether `impute.bin` columns were, or were not, imputed.
`same`	Whether `x.train` and `x.test` are identical.

Cindex *Calculate the C-index/concordance for survival analysis.*

Description

The C-index for survival analysis is the corollary of the c statistic (the area under the Receiver Operating Characteristic curve) for binary outcomes. As a probability, the higher is the C-index, the better is the model discrimination vs. lesser probability values. Similarly, the concordance is calculated like the C-index from z-draws via the posterior predictive distribution restricted to the horizon of the data (a la restricted mean survival time).

Usage

```
Cindex(risk, times, delta=NULL)

concordance(draws, times, delta=NULL)
```

Arguments

`risk`	A vector or prognostic risk scores.
`draws`	A vector of draws via the posterior predictive distribution restricted to the horizon of the data (a la restricted mean survival time).
`times`	A vector of failure times.
`delta`	The corresponding failure time status code: 0, right-censored; 1, failure; or 2, left-censored. Defaults to all failures if not specified.

Value

The return value is the calculated C-index/concordance.

References

Harrell FE, Califf RM, Pryor DB, Lee KL, Rosati RA. (1982) Evaluating the yield of medical tests. JAMA, May 14;247(18):2543-6.

See Also

predict.nft

Examples

```
data(lung)
N=length(lung$status)

##lung$status: 1=censored, 2=dead
##delta: 0=censored, 1=dead
delta=lung$status-1

## this study reports time in days
times=lung$time
times=times/7  ## weeks

## matrix of covariates
x.train=cbind(lung[ , -(1:3)])
## lung$sex:        Male=1 Female=2

## Not run:
    set.seed(99)
    post=nft(x.train, times, delta, K=0)
    pred=predict(post, x.train, XPtr=TRUE, seed=21)
    print(Cindex(pred$logt.test.mean, times, delta))

## End(Not run)
```

---

lung *NCCTG Lung Cancer Data*

Description

Survival for 228 patients with advanced lung cancer was recorded up to a median of roughly one year by the North Central Cancer Treatment Group. Performance scores rate how well the patient can perform usual daily activities.

Format

inst:	Institution code
time:	Survival time in days
status:	censoring status 1=censored, 2=dead
age:	Age in years
sex:	Male=1 Female=2
ph.ecog:	ECOG performance score (0=good 5=dead)
ph.karno:	Karnofsky performance score (bad=0-good=100) rated by physician
pat.karno:	Karnofsky performance score as rated by patient
meal.cal:	Calories consumed at meals
wt.loss:	Weight loss in last six months

Source

Terry Therneau

References

Loprinzi CL. Laurie JA. Wieand HS. Krook JE. Novotny PJ. Kugler JW. Bartel J. Law M. Bateman M. Klatt NE. et al. Prospective evaluation of prognostic variables from patient-completed questionnaires. North Central Cancer Treatment Group. Journal of Clinical Oncology. 12(3):601-7, 1994.

Examples

```
data(lung)
```

---

nft2 *Fit NFT BART models.*

Description

The `nft2()/nft()` function is for fitting NFT BART (Nonparametric Failure Time Bayesian Additive Regression Tree) models with different train/test matrices for f and sd functions.

Usage

```
nft2(
    ## data
    xftrain, xstrain, times, delta=NULL,
    xftest=matrix(nrow=0, ncol=0),
    xstest=matrix(nrow=0, ncol=0),
    rm.const=TRUE, rm.dupe=TRUE, right.max=Inf,
```

```
    ## multi-threading
    tc=getOption("mc.cores", 1),
    ##MCMC
    nskip=1000, ndpost=2000, nadapt=1000, adaptevery=100,
    chvf=NULL, chvs=NULL,
    method="spearman", use="pairwise.complete.obs",
    pbd=c(0.7, 0.7), pb=c(0.5, 0.5),
    stepwpert=c(0.1, 0.1), probchv=c(0.1, 0.1),
    minnumbot=c(5, 5),
    ## BART and HBART prior parameters
    ntree=c(50, 10), numcut=100,
    xifcuts=NULL, xiscuts=NULL,
    power=c(2, 2), base=c(0.95, 0.95),
    ## f function
    fmu=NA, k=5, tau=NA, dist='weibull',
    ## s function
    total.lambda=NA, total.nu=10, mask=NULL,
    ## survival analysis
    K=100, events=NULL, TSVS=FALSE,
    ## DPM LIO
    drawDPM=1L,
    alpha=1, alpha.a=1, alpha.b=0.1, alpha.draw=1,
    neal.m=2, constrain=1,
    m0=0, k0.a=1.5, k0.b=7.5, k0=1, k0.draw=1,
    a0=3, b0.a=2, b0.b=1, b0=1, b0.draw=1,
    ## misc
    na.rm=FALSE, probs=c(0.025, 0.975), printevery=100,
    transposed=FALSE, pred=FALSE
)

nft(
    ## data
    x.train, times, delta=NULL, x.test=matrix(nrow=0, ncol=0),
    rm.const=TRUE, rm.dupe=TRUE, right.max=Inf,
    ## multi-threading
    tc=getOption("mc.cores", 1),
    ##MCMC
    nskip=1000, ndpost=2000, nadapt=1000, adaptevery=100,
    chv=NULL,
    method="spearman", use="pairwise.complete.obs",
    pbd=c(0.7, 0.7), pb=c(0.5, 0.5),
    stepwpert=c(0.1, 0.1), probchv=c(0.1, 0.1),
    minnumbot=c(5, 5),
    ## BART and HBART prior parameters
    ntree=c(50, 10), numcut=100, xicuts=NULL,
    power=c(2, 2), base=c(0.95, 0.95),
    ## f function
    fmu=NA, k=5, tau=NA, dist='weibull',
```

```
    ## s function
    total.lambda=NA, total.nu=10, mask=NULL,
    ## survival analysis
    K=100, events=NULL, TSVS=FALSE,
    ## DPM LIO
    drawDPM=1L,
    alpha=1, alpha.a=1, alpha.b=0.1, alpha.draw=1,
    neal.m=2, constrain=1,
    m0=0, k0.a=1.5, k0.b=7.5, k0=1, k0.draw=1,
    a0=3, b0.a=2, b0.b=1, b0=1, b0.draw=1,
    ## misc
    na.rm=FALSE, probs=c(0.025, 0.975), printevery=100,
    transposed=FALSE, pred=FALSE
)
```

Arguments

xftrain	n x pf matrix of predictor variables for the training data.
xstrain	n x ps matrix of predictor variables for the training data.
x.train	n x p matrix of predictor variables for the training data.
times	n x 1 vector of the observed times for the training data.
delta	n x 1 vector of the time type for the training data: 0, for right-censoring; 1, for an event; and, 2, for left-censoring.
xftest	m x pf matrix of predictor variables for the test set.
xstest	m x ps matrix of predictor variables for the test set.
x.test	m x p matrix of predictor variables for the test set.
rm.const	To remove constant variables or not.
rm.dupe	To remove duplicate variables or not.
right.max	The maximum value augmented by right censoring.
tc	Number of OpenMP threads to use.
nskip	Number of MCMC iterations to burn-in and discard.
ndpost	Number of MCMC iterations kept after burn-in.
nadapt	Number of MCMC iterations for adaptation prior to burn-in.
adaptevery	Adapt MCMC proposal distributions every adaptevery iteration.
chvf, chvs, chv	Predictor correlation matrix used as a pre-conditioner for MCMC change-of-variable proposals.
method, use	Correlation options for change-of-variable proposal pre-conditioner.
pbd	Probability of performing a birth/death proposal, otherwise perform a rotate proposal.
pb	Probability of performing a birth proposal given that we choose to perform a birth/death proposal.
stepwpert	Initial width of proposal distribution for peturbing cut-points.

probchv — Probability of performing a change-of-variable proposal. Otherwise, only do a perturb proposal.

minnumbot — Minimum number of observations required in leaf (terminal) nodes.

ntree — Vector of length two for the number of trees used for the mean model and the number of trees used for the variance model.

numcut — Number of cutpoints to use for each predictor variable.

xifcuts, xiscuts, xicuts — More detailed construction of cut-points can be specified by the xicuts function and provided here.

power — Power parameter in the tree depth penalizing prior.

base — Base parameter in the tree depth penalizing prior.

fmu — Prior parameter for the center of the mean model.

k — Prior parameter for the mean model.

tau — Desired SD/ntree for f function leaf prior if known.

dist — Distribution to be passed to intercept-only AFT model to center y.train.

total.lambda — A rudimentary estimate of the process standard deviation. Used in calibrating the variance prior.

total.nu — Shape parameter for the variance prior.

mask — If a proportion is provided, then said quantile of max.i sd(x.i) is used to mask non-stationary departures (with respect to convergence) above this threshold.

K — Number of grid points for which to estimate survival probability.

events — Grid points for which to estimate survival probability.

TSVS — Setting to TRUE will avoid unnecessary processing for Thompson sampling variable selection, i.e., all that is needed is the variable counts from the tree branch decision rules.

drawDPM — Whether to utilize DPM or not.

alpha — Initial value of DPM concentration parameter.

alpha.a — Gamma prior parameter setting for DPM concentration parameter where E[alpha]=alpha.a/alpha.b.

alpha.b — See alpha.a above.

alpha.draw — Whether to draw alpha or it is fixed at the initial value.

neal.m — The number of additional atoms for Neal 2000 DPM algorithm 8.

constrain — Whether to perform constained DPM or unconstrained.

m0 — Center of the error distribution: defaults to zero.

k0.a — First Gamma prior argument for k0.

k0.b — Second Gamma prior argument for k0.

k0 — Initial value of k0.

k0.draw — Whether to fix k0 or draw it if from the DPM LIO prior hierarchy: k0~Gamma(k0.a, k0.b), i.e., E[k0]=k0.a/k0.b.

a0 — First Gamma prior argument for tau.

b0.a	First Gamma prior argument for b0.
b0.b	Second Gamma prior argument for b0.
b0	Initial value of b0.
b0.draw	Whether to fix b0 or draw it from the DPM LIO prior hierarchy: b0~Gamma(b0.a, b0.b), i.e., E[b0]=b0.a/b0.b.
na.rm	Value to be passed to the predict function.
probs	Value to be passed to the predict function.
printevery	Outputs MCMC algorithm status every printevery iterations.
transposed	Specify TRUE if all of the pre-processing for xftrain/xstrain/xftest/xstest has been conducted prior to the call (including tranposing).
pred	Specify TRUE if you want to return the pred item that is used to calculate soffset.

Details

nft2()/nft() is the function to fit time-to-event data. The most general form of the model allowed is $Y(\mathbf{x}) = mu + f(\mathbf{x}) + sd(\mathbf{x})Z$ where E follows a nonparametric error distribution by default. The nft2()/nft() function returns a fit object of S3 class type nft2/nft that is essentially a list containing the following items.

Value

ots, oid, ovar, oc, otheta
: These are XPtrs to the BART $f(x)$ objects in RAM that are only available for fits generated in the current R session.

sts, sid, svar, sc, stheta
: Similarly, these are XPtrs to the HBART $sd(x)$ objects.

fmu
: The constant mu.

f.train, s.train
: The trained $f(x)$ and $sd(x)$ respectively: matrices with ndpost rows and n columns.

f.train.mean, s.train.mean
: The posterior mean of the trained $f(x)$ and $sd(x)$ respectively: vectors of length n.

f.trees, s.trees
: Character strings representing the trained fits of $f(x)$ and $sd(x)$ respectively to facilitate usage of the predict function when XPtrs are unavailable.

dpalpha
: The draws of the DPM concentration parameter $alpha$.

dpn, dpn.
: The number of atom clusters per DPM, J, for all draws including burn-in and excluding burn-in respectively.

dpmu
: The draws of the DPM parameter $mu[i]$ where $i = 1, ..., n$ indexes subjects: a matrix with ndpost rows and n columns.

dpmu.
: The draws of the DPM parameter $mu[j]$ where $j = 1, ..., J$ indexes atom clusters: a matrix with ndpost rows and J columns.

`dpwt.` The weights for efficient DPM calculations by atom clusters (as opposed to subjects) for use with $dpmu.$ (and $dpsd.$; see below): a matrix with `ndpost` rows and J columns.

`dpsd`, `dpsd.` Similarly, the draws of the DPM parameter $tau[i]$ transformed into the standard deviation $sigma[i]$ for convenience.

`dpC` The indices j for each subject i corresponding to their shared atom cluster.

`z.train` The data values/augmentation draws of $logt$.

`f.tmind/f.tavgd/f.tmaxd`
The min/average/max tier degree of trees in the f ensemble.

`s.tmind/s.tavgd/s.tmaxd`
The min/average/max tier degree of trees in the s ensemble.

`f.varcount`, `s.varcount`
Variable importance counts of branch decision rules for each x of f and s respectively: matrices with `ndpost` rows and p columns.

`f.varcount.mean`, `s.varcount.mean`
Similarly, the posterior mean of the variable importance counts for each x of f and s respectively: vectors of length p.

`f.varprob`, `s.varprob`
Similarly, re-weighting the posterior mean of the variable importance counts as sum-to-one probabilities for each x of f and s respectively: vectors of length p.

`LPML` The log Pseudo-Marginal Likelihood as typically calculated for right-/left-censoring.

`pred` The object returned from the `predict` function where `x.test=x.train` in order to calculate the `soffset` item that is needed to use `predict` when `XPtrs` are not available.

`soffset` See `pred` above.

`aft` The AFT model fit used to initialize NFT BART.

`elapsed` The elapsed time of the run in seconds.

Author(s)

Rodney Sparapani: <rsparapa@mcw.edu>

References

Sparapani R., Logan B., Maiers M., Laud P., McCulloch R. (2023) Nonparametric Failure Time: Time-to-event Machine Learning with Heteroskedastic Bayesian Additive Regression Trees and Low Information Omnibus Dirichlet Process Mixtures *Biometrics (ahead of print)* <doi:10.1111/biom.13857>.

See Also

`predict.nft2`, `predict.nft`

Examples

```
##library(nftbart)
data(lung)
N=length(lung$status)

##lung$status: 1=censored, 2=dead
##delta: 0=censored, 1=dead
delta=lung$status-1

## this study reports time in days rather than weeks or months
times=lung$time
times=times/7  ## weeks

## matrix of covariates
x.train=cbind(lung[ , -(1:3)])
## lung$sex:        Male=1 Female=2

## token run just to test installation
post=nft2(x.train, x.train, times, delta, K=0,
         nskip=0, ndpost=10, nadapt=4, adaptevery=1)


set.seed(99)
post=nft2(x.train, x.train, times, delta, K=0)
XPtr=TRUE

x.test = rbind(x.train, x.train)
x.test[ , 2]=rep(1:2, each=N)
K=75
events=seq(0, 150, length.out=K+1)
pred = predict(post, x.test, x.test, K=K, events=events[-1],
               XPtr=XPtr, FPD=TRUE)

plot(events, c(1, pred$surv.fpd.mean[1:K]), type='l', col=4,
     ylim=0:1,
     xlab=expression(italic(t)), sub='weeks',
     ylab=expression(italic(S)(italic(t), italic(x))))
lines(events, c(1, pred$surv.fpd.upper[1:K]), lty=2, lwd=2, col=4)
lines(events, c(1, pred$surv.fpd.lower[1:K]), lty=2, lwd=2, col=4)
lines(events, c(1, pred$surv.fpd.mean[K+1:K]), lwd=2, col=2)
lines(events, c(1, pred$surv.fpd.upper[K+1:K]), lty=2, lwd=2, col=2)
lines(events, c(1, pred$surv.fpd.lower[K+1:K]), lty=2, lwd=2, col=2)
legend('topright', c('Adv. lung cancer\nmortality example',
                     'M', 'F'), lwd=2, col=c(0, 4, 2), lty=1)
```

`predict.aftree` *Estimating the survival and the hazard for AFT BART models.*

Description

The function `predict.aftree()` is provided for performing posterior inference via test data set estimates stored in a `aftree` object returned from `AFTree()` in a similar fashion as that of `predict.nft`. N.B. the `x.test` matrix must be provided on the `AFTree()` function call. Here we are only calculating the survival function by default, and, if requested, the hazard as well.

Usage

```
## S3 method for class 'aftree'
predict(
            ## data
            object,
            ## predictions
            events=NULL,
            FPD=FALSE,
            probs=c(0.025, 0.975),
            take.logs=TRUE,
            seed=NULL,
            ## default settings
            ndpost=nrow(object$mix.prop),
            nclust=ncol(object$mix.prop),
            ## etc.
            ...)
```

Arguments

`object`	Object of type `nft` from a previous call to `nft()`.
`events`	You must specify a grid of time-points; however, they can be a matrix with rows for each subject.
`FPD`	Whether to yield the usual predictions or marginal predictions calculated by the partial dependence function.
`probs`	A vector of length two containing the lower and upper quantiles to be calculated for the predictions.
`take.logs`	Whether or not to take logarithms.
`seed`	If provided, then this value is used to generate random natural logarithms of event times from the predictive distribution.
`ndpost`	The number of MCMC samples generated.
`nclust`	The number of DPM clusters generated.
`...`	The et cetera objects passed to the `predict` method. Currently, it has no functionality.

Details

Returns a list with the following entries. If `hazard=TRUE` is specified, then a similar set of entries for the hazard are produced.

Value

surv.fpd	Survival function posterior draws on a grid of time-points by the partial dependence function when requested.
surv.fpd.mean	Survival function estimates on a grid of time-points by the partial dependence function when requested.
surv.fpd.lower	Survival function lower quantiles on a grid of time-points by the partial dependence function when requested.
surv.fpd.upper	Survival function upper quantiles on a grid of time-points by the partial dependence function when requested.

Author(s)

Rodney Sparapani: <rsparapa@mcw.edu>

See Also

predict.nft

---

predict.nft2 *Drawing Posterior Predictive Realizations for NFT BART models.*

Description

The function predict.nft2()/predict.nft() is the main function for drawing posterior predictive realizations at new inputs using a fitted model stored in a nft2/nft object returned from nft2()/nft().

Usage

```
## S3 method for class 'nft2'
predict(
            ## data
            object,
            xftest=object$xftrain,
            xstest=object$xstrain,
            ## multi-threading
            tc=getOption("mc.cores", 1), ##OpenMP thread count
            ## current process fit vs. previous process fit
            XPtr=TRUE,
            ## predictions
            K=0,
            events=object$events,
            FPD=FALSE,
            probs=c(0.025, 0.975),
            take.logs=TRUE,
            na.rm=FALSE,
```

```
            RMST.max=NULL,
            ## default settings for NFT:BART/HBART/DPM
            fmu=object$NFT$fmu,
            soffset=object$soffset,
            drawDPM=object$drawDPM,
            ## etc.
            ...)

## S3 method for class 'nft'
predict(
            ## data
            object,
            x.test=object$x.train,
            ## multi-threading
            tc=getOption("mc.cores", 1), ##OpenMP thread count
            ## current process fit vs. previous process fit
            XPtr=TRUE,
            ## predictions
            K=0,
            events=object$events,
            FPD=FALSE,
            probs=c(0.025, 0.975),
            take.logs=TRUE,
            na.rm=FALSE,
            RMST.max=NULL,
            ## default settings for NFT:BART/HBART/DPM
            fmu=object$NFT$fmu,
            soffset=object$soffset,
            drawDPM=object$drawDPM,
            ## etc.
            ...)
```

Arguments

`object`	Object of type `nft2/nft` from a previous call to `nft2()/nft()`.
`xftest, xstest, x.test`	New input settings in the form of a matrix at which to construct predictions. Defaults to the training inputs.
`tc`	Number of OpenMP threads to use for parallel computing.
`XPtr`	If `object` was created during the currently running R process, then (via an Rcpp `XPtr`) the BART/HBART tree ensemble objects can be accessed in RAM; otherwise, those objects will need to be loaded from their string encodings.
`K`	The length of the grid of time-points to be used for survival predictions. Set to zero to avoid these calculations which can be time-consuming for large data sets.
`events`	You can specify the grid of time-points; otherwise, they are derived from quantiles of the augmented event times.

FPD	Whether to yield the usual predictions or marginal predictions calculated by the partial dependence function.
probs	A vector of length two containing the lower and upper quantiles to be calculated for the predictions.
take.logs	Whether or not to take logarithms.
na.rm	Whether NA values should be removed from the summaries.
RMST.max	To calculate Restricted Mean Survival Time (RMST), we need to set a reasonable time maxima. Typically, a clinically important time that a majority (or a large plurality) of censored subjects have been followed through that point or beyond.
fmu	BART centering parameter for the test data. Defaults to the value used by nft2()/nft() when training the model.
soffset	HBART centering parameter for the test data. Defaults to the value used by nft2()/nft() when training the model.
drawDPM	Whether NFT BART was fit with, or without, DPM.
...	The et cetera objects passed to the predict method. Currently, it has no functionality.

Details

predict.nft2()/predict.nft() is the main function for calculating posterior predictions and uncertainties once a model has been fit by nft2()/nft().

Returns a list with the following entries.

Value

f.test	Posterior realizations of the mean function stored in a matrix. Omitted if partial dependence functions are performed since these will typically be large.
s.test	Posterior realizations of the SD function stored in a matrix. Omitted if partial dependence functions are performed since these will typically be large.
f.test.mean	Posterior predictive mean of mean function.
f.test.lower	Posterior predictive lower quantile of mean function.
f.test.upper	Posterior predictive upper quantile of mean function.
s.test.mean	Posterior predictive mean of SD function.
s.test.lower	Posterior predictive lower quantile of SD function.
s.test.upper	Posterior predictive upper quantile of SD function.
surv.fpd	Survival function posterior draws on a grid of time-points by the partial dependence function when requested.
surv.fpd.mean	Survival function estimates on a grid of time-points by the partial dependence function when requested.
surv.fpd.lower	Survival function lower quantiles on a grid of time-points by the partial dependence function when requested.
surv.fpd.upper	Survival function upper quantiles on a grid of time-points by the partial dependence function when requested.

Author(s)

Rodney Sparapani: <rsparapa@mcw.edu>

See Also

nft2, nft

---

tsvs2 *Variable selection with NFT BART models.*

Description

The tsvs2()/tsvs() function is for Thompson sampling variable selection with NFT BART.

Usage

```
tsvs2(
                ## data
                xftrain, xstrain, times, delta=NULL,
                rm.const=TRUE, rm.dupe=TRUE, right.max=Inf,
                ##tsvs args
                K=20, a.=1, b.=0.5, C=0.5,
                rds.file='tsvs2.rds', pdf.file='tsvs2.pdf',
                ## multi-threading
                tc=getOption("mc.cores", 1), ##OpenMP thread count
                ##MCMC
                nskip=1000, ndpost=2000,
                nadapt=1000, adaptevery=100,
                chvf=NULL, chvs=NULL,
                method="spearman", use="pairwise.complete.obs",
                pbd=c(0.7, 0.7), pb=c(0.5, 0.5),
                stepwpert=c(0.1, 0.1), probchv=c(0.1, 0.1),
                minnumbot=c(5, 5),
                ## BART and HBART prior parameters
                ntree=c(10, 2), numcut=100,
                xifcuts=NULL, xiscuts=NULL,
                power=c(2, 2), base=c(0.95, 0.95),
                ## f function
                fmu=NA, k=5, tau=NA, dist='weibull',
                ## s function
                total.lambda=NA, total.nu=10, mask=0.95,
                ## survival analysis
                ##K=100, events=NULL,
                ## DPM LIO
                drawDPM=1L,
                alpha=1, alpha.a=1, alpha.b=0.1, alpha.draw=1,
                neal.m=2, constrain=1,
```

```
                m0=0, k0.a=1.5, k0.b=7.5, k0=1, k0.draw=1,
                a0=3, b0.a=2, b0.b=1, b0=1, b0.draw=1,
                ## misc
                na.rm=FALSE, probs=c(0.025, 0.975), printevery=100,
                transposed=FALSE
)

tsvs(
                ## data
                x.train, times, delta=NULL,
                rm.const=TRUE, rm.dupe=TRUE, right.max=Inf,
                ##tsvs args
                K=20, a.=1, b.=0.5, C=0.5,
                rds.file='tsvs.rds', pdf.file='tsvs.pdf',
                ## multi-threading
                tc=getOption("mc.cores", 1), ##OpenMP thread count
                ##MCMC
                nskip=1000, ndpost=2000,
                nadapt=1000, adaptevery=100,
                chv=NULL,
                method="spearman", use="pairwise.complete.obs",
                pbd=c(0.7, 0.7), pb=c(0.5, 0.5),
                stepwpert=c(0.1, 0.1), probchv=c(0.1, 0.1),
                minnumbot=c(5, 5),
                ## BART and HBART prior parameters
                ntree=c(10, 2), numcut=100, xicuts=NULL,
                power=c(2, 2), base=c(0.95, 0.95),
                ## f function
                fmu=NA, k=5, tau=NA, dist='weibull',
                ## s function
                total.lambda=NA, total.nu=10, mask=0.95,
                ## survival analysis
                ##K=100, events=NULL,
                ## DPM LIO
                drawDPM=1L,
                alpha=1, alpha.a=1, alpha.b=0.1, alpha.draw=1,
                neal.m=2, constrain=1,
                m0=0, k0.a=1.5, k0.b=7.5, k0=1, k0.draw=1,
                a0=3, b0.a=2, b0.b=1, b0=1, b0.draw=1,
                ## misc
                na.rm=FALSE, probs=c(0.025, 0.975), printevery=100,
                transposed=FALSE
)
```

Arguments

xftrain	n x pf matrix of predictor variables for the training data.

xstrain	n x ps matrix of predictor variables for the training data.
x.train	n x ps matrix of predictor variables for the training data.
times	nx1 vector of the observed times for the training data.
delta	nx1 vector of the time type for the training data: 0, for right-censoring; 1, for an event; and, 2, for left-censoring.
rm.const	To remove constant variables or not.
rm.dupe	To remove duplicate variables or not.
right.max	The maximum value augmented by right censoring.
K	The number of Thompson sampling steps to take. Not to be confused with the size of the time grid for survival distribution estimation.
a.	The prior parameter for successes of a Beta distribution.
b.	The prior parameter for failures of a Beta distribution.
C	The probability cut-off for variable selection.
rds.file	File name to store RDS object containing Thompson sampling parameters.
pdf.file	File name to store PDF graphic of variables selected.
tc	Number of OpenMP threads to use.
nskip	Number of MCMC iterations to burn-in and discard.
ndpost	Number of MCMC iterations kept after burn-in.
nadapt	Number of MCMC iterations for adaptation prior to burn-in.
adaptevery	Adapt MCMC proposal distributions every adaptevery iteration.
chvf, chvs, chv	Predictor correlation matrix used as a pre-conditioner for MCMC change-of-variable proposals.
method, use	Correlation options for change-of-variable proposal pre-conditioner.
pbd	Probability of performing a birth/death proposal, otherwise perform a rotate proposal.
pb	Probability of performing a birth proposal given that we choose to perform a birth/death proposal.
stepwpert	Initial width of proposal distribution for peturbing cut-points.
probchv	Probability of performing a change-of-variable proposal. Otherwise, only do a perturb proposal.
minnumbot	Minimum number of observations required in leaf (terminal) nodes.
ntree	Vector of length two for the number of trees used for the mean model and the number of trees used for the variance model.
numcut	Number of cutpoints to use for each predictor variable.
xifcuts, xiscuts, xicuts	More detailed construction of cut-points can be specified by the xicuts function and provided here.
power	Power parameter in the tree depth penalizing prior.
base	Base parameter in the tree depth penalizing prior.

`fmu`	Prior parameter for the center of the mean model.
`k`	Prior parameter for the mean model.
`tau`	Desired `SD/ntree` for f function leaf prior if known.
`dist`	Distribution to be passed to intercept-only AFT model to center `y.train`.
`total.lambda`	A rudimentary estimate of the process standard deviation. Used in calibrating the variance prior.
`total.nu`	Shape parameter for the variance prior.
`mask`	If a proportion is provided, then said quantile of `max.i sd(x.i)` is used to mask non-stationary departures (with respect to convergence) above this threshold.
`drawDPM`	Whether to utilize DPM or not.
`alpha`	Initial value of DPM concentration parameter.
`alpha.a`	Gamma prior parameter setting for DPM concentration parameter where E[`alpha`]=`alpha.a`/`alpha.b`.
`alpha.b`	See `alpha.a` above.
`alpha.draw`	Whether to draw `alpha` or it is fixed at the initial value.
`neal.m`	The number of additional atoms for Neal 2000 DPM algorithm 8.
`constrain`	Whether to perform constained DPM or unconstrained.
`m0`	Center of the error distribution: defaults to zero.
`k0.a`	First Gamma prior argument for `k0`.
`k0.b`	Second Gamma prior argument for `k0`.
`k0`	Initial value of `k0`.
`k0.draw`	Whether to fix k0 or draw it if from the DPM LIO prior hierarchy: `k0~Gamma(k0.a, k0.b)`, i.e., `E[k0]=k0.a/k0.b`.
`a0`	First Gamma prior argument for tau.
`b0.a`	First Gamma prior argument for `b0`.
`b0.b`	Second Gamma prior argument for `b0`.
`b0`	Initial value of `b0`.
`b0.draw`	Whether to fix b0 or draw it from the DPM LIO prior hierarchy: `b0~Gamma(b0.a, b0.b)`, i.e., `E[b0]=b0.a/b0.b`.
`na.rm`	Value to be passed to the `predict` function.
`probs`	Value to be passed to the `predict` function.
`printevery`	Outputs MCMC algorithm status every printevery iterations.
`transposed`	`tsvs` handles all of the pre-processing for `x.train`/`x.test` (including tranposing) computational efficiency.

Details

`tsvs2()`/`tsvs()` is the function to perform variable selection. The `tsvs2()`/`tsvs()` function returns a fit object of S3 class type `list` as well as storing it in `rds.file` for sampling in progress.

Author(s)

Rodney Sparapani: <rsparapa@mcw.edu>

References

Sparapani R., Logan B., Maiers M., Laud P., McCulloch R. (2023) Nonparametric Failure Time: Time-to-event Machine Learning with Heteroskedastic Bayesian Additive Regression Trees and Low Information Omnibus Dirichlet Process Mixtures *Biometrics (ahead of print)* <doi:10.1111/biom.13857>.

Liu Y., Rockova V. (2021) Variable selection via Thompson sampling. *Journal of the American Statistical Association. Jun 29:1-8.*

See Also

tsvs

Examples

```
##library(nftbart)
data(lung)
N=length(lung$status)

##lung$status: 1=censored, 2=dead
##delta: 0=censored, 1=dead
delta=lung$status-1

## this study reports time in days rather than weeks or months
times=lung$time
times=times/7  ## weeks

## matrix of covariates
x.train=cbind(lung[ , -(1:3)])
## lung$sex:        Male=1 Female=2


##vars=tsvs2(x.train, x.train, times, delta)
vars=tsvs2(x.train, x.train, times, delta, K=0) ## K=0 just returns 0
```

---

xicuts *Specifying cut-points for the covariates*

Description

This function allows you to create a list that specifies the cut-points for the covariates.

Usage

```
xicuts(x.train, transposed=FALSE, numcut=100)
```

Arguments

x.train	The training matrix to derive cut-points from.
transposed	Whether or not the matrix has been tranposed yet.
numcut	The number of cut-points to create.

Details

The cut-points are generated uniformly from min. to max., i.e., the distribution of the data is ignored.

Value

An object is returned of type `BARTcutinfo` which is essentially a list.

Index

∗ **datasets**
bmx, 5
CDCheight, 6
lung, 8

bartModelMatrix, 2
bMM, 3, *3*
bmx, 5

cancer *(*lung*)*, 8
CDCheight, 6
CDimpute, 6
Cindex, 7
concordance *(*Cindex*)*, 7

lung, 8

nft, *20*
nft *(*nft2*)*, 9
nft2, 9, *20*

predict.aftree, 15
predict.nft, *8*, *14*, *17*
predict.nft *(*predict.nft2*)*, 17
predict.nft2, *14*, 17

tsvs, *24*
tsvs *(*tsvs2*)*, 20
tsvs2, 20

xicuts, *4*, 24