Package 'BASS'

May 6, 2026

Version 1.3.1

Date 2023-7-1

Title Bayesian Adaptive Spline Surfaces

Description Bayesian fitting and sensitivity analysis methods for adaptive spline surfaces described in <doi:10.18637/jss.v094.i08>. Built to handle continuous and categorical inputs as well as functional or scalar output. An extension of the methodology in Denison, Mallick and Smith (1998) <doi:10.1023/A:1008824606259>.

License GPL-3

Suggests R.rsp, testthat (>= 2.1.0), parallel

VignetteBuilder R.rsp

Imports truncdist, hypergeo

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation no

Author Devin Francom [aut, cre],
Bruno Sanso [ths],
Kellin Rumsey [aut]

Maintainer Devin Francom <devinfrancom@gmail.com>

Repository CRAN

Date/Publication 2023-07-04 13:13:07 UTC

Contents

---

bass *Bayesian Adaptive Spline Surfaces (BASS)*

Description

Fits a BASS model using RJMCMC. Optionally uses parallel tempering to improve mixing. Can be used with scalar or functional response. Also can use categorical inputs.

Usage

```
bass(
  xx,
  y,
  maxInt = 3,
  maxInt.func = 3,
  maxInt.cat = 3,
  xx.func = NULL,
  degree = 1,
  maxBasis = 1000,
  npart = NULL,
  npart.func = NULL,
  nmcmc = 10000,
  nburn = 9000,
  thin = 1,
  g1 = 0,
  g2 = 0,
  s2.lower = 0,
  h1 = 10,
  h2 = 10,
  a.tau = 0.5,
  b.tau = NULL,
  w1 = 5,
  w2 = 5,
  beta.prior = "g",
  temp.ladder = NULL,
  start.temper = NULL,
  curr.list = NULL,
```

```
  save.yhat = TRUE,
  small = FALSE,
  verbose = TRUE,
  ret.str = F
)
```

Arguments

xx	a data frame or matrix of predictors. Categorical predictors should be included as factors.
y	a response vector (scalar response) or matrix (functional response). Note: If sum(y^2) is large (i.e. 1e10), please center/rescale (and rescale g1 and g2 if necessary).
maxInt	integer for maximum degree of interaction in spline basis functions. Defaults to the number of predictors, which could result in overfitting.
maxInt.func	(functional response only) integer for maximum degree of interaction in spline basis functions describing the functional response.
maxInt.cat	(categorical input only) integer for maximum degree of interaction of categorical inputs.
xx.func	a vector, matrix or data frame of functional variables.
degree	degree of splines. Stability should be examined for anything other than 1.
maxBasis	maximum number of basis functions. This should probably only be altered if you run out of memory.
npart	minimum number of non-zero points in a basis function. If the response is functional, this refers only to the portion of the basis function coming from the non-functional predictors. Defaults to 20 or 0.1 times the number of observations, whichever is smaller.
npart.func	same as npart, but for functional portion of basis function.
nmcmc	number of RJMCMC iterations.
nburn	number of the nmcmc iterations to disregard.
thin	keep every thin samples
g1	shape for IG prior on σ^2.
g2	scale for IG prior on σ^2.
s2.lower	lower bound for s2. Turns IG prior for s2 into a truncated IG.
h1	shape for gamma prior on λ.
h2	rate for gamma prior on λ. This is the primary way to control overfitting. A large value of h2 favors fewer basis functions.
a.tau	shape for gamma prior on τ.
b.tau	rate for gamma prior on τ. Defaults to one over the number of observations, which centers the prior for the basis function weights on the unit information prior.
w1	nominal weight for degree of interaction, used in generating candidate basis functions. Should be greater than 0.

w2	nominal weight for variables, used in generating candidate basis functions. Should be greater than 0.
beta.prior	what type of prior to use for basis coefficients, "g" or "jeffreys"
temp.ladder	temperature ladder used for parallel tempering. The first value should be 1 and the values should increase.
start.temper	when to start tempering (after how many MCMC iterations). Defaults to 1000 or half of burn-in, whichever is smaller.
curr.list	list of starting models (one element for each temperature), could be output from a previous run under the same model setup.
save.yhat	logical; should predictions of training data be saved?
small	logical; if true, returns a smaller object by leaving out curr.list and other unnecessary objects. Use in combination with save.yhat to get smaller memory footprint for very large models.
verbose	logical; should progress be displayed?
ret.str	logical; return data and prior structures

Details

Explores BASS model space by RJMCMC. The BASS model has

$$y = f(x) + \epsilon, \;\; \epsilon \sim N(0, \sigma^2)$$

$$f(x) = a_0 + \sum_{m=1}^{M} a_m B_m(x)$$

and $B_m(x)$ is a BASS basis function (tensor product of spline basis functions). We use priors

$$a \sim N(0, \sigma^2/\tau(B'B)^{-1})$$

$$M \sim Poisson(\lambda)$$

as well as the priors mentioned in the arguments above.

Value

An object of class 'bass'. The other output will only be useful to the advanced user. Rather, users may be interested in prediction and sensitivity analysis, which are obtained by passing the entire object to the predict.bass or sobol functions.

See Also

predict.bass for prediction and sobol for sensitivity analysis.

Examples

```
## Not run:
#############################################################################################################
### univariate example
#############################################################################################################
## simulate data (Friedman function)
f<-function(x){
  10*sin(pi*x[,1]*x[,2])+20*(x[,3]-.5)^2+10*x[,4]+5*x[,5]
}
sigma<-1 # noise sd
n<-500 # number of observations
x<-matrix(runif(n*10),n,10) #10 variables, only first 5 matter
y<-rnorm(n,f(x),sigma)

## fit BASS, no tempering
mod<-bass(x,y)
plot(mod)
## fit BASS, tempering
mod<-bass(x,y,temp.ladder=1.3^(0:8),start.temper=1000)
plot(mod)

## prediction
npred<-1000
xpred<-matrix(runif(npred*10),npred,10)
pred<-predict(mod,xpred,verbose=TRUE) # posterior predictive samples
true.y<-f(xpred)
plot(true.y,colMeans(pred),xlab='true values',ylab='posterior predictive means')
abline(a=0,b=1,col=2)

## sensitivity
sens<-sobol(mod)
plot(sens,cex.axis=.5)

#############################################################################################################
### functional example
#############################################################################################################
## simulate data (Friedman function with first variable as functional)
sigma<-1 # noise sd
n<-500 # number of observations
nfunc<-50 # size of functional variable grid
xfunc<-seq(0,1,length.out=nfunc) # functional grid
x<-matrix(runif(n*9),n,9) # 9 non-functional variables, only first 4 matter
X<-cbind(rep(xfunc,each=n),kronecker(rep(1,nfunc),x)) # to get y
y<-matrix(f(X),nrow=n)+rnorm(n*nfunc,0,sigma)

## fit BASS
mod<-bass(x,y,xx.func=xfunc)
plot(mod)

## prediction
npred<-100
xpred<-matrix(runif(npred*9),npred,9)
```

```
Xpred<-cbind(rep(xfunc,each=npred),kronecker(rep(1,nfunc),xpred))
ypred<-matrix(f(Xpred),nrow=npred)
pred<-predict(mod,xpred) # posterior predictive samples (each is a curve)
matplot(ypred,apply(pred,2:3,mean),type='l',xlab='observed',ylab='mean prediction')
abline(a=0,b=1,col=2)
matplot(t(ypred),type='l') # actual
matplot(t(apply(pred,2:3,mean)),type='l') # mean prediction

## sensitivity
sens<-sobol(mod,mcmc.use=1:10) # for speed, only use a few samples
plot(sens) # functional variable labelled "a"

sens.func<-sobol(mod,mcmc.use=1:10,func.var=1)
plot(sens.func)

## End(Not run)

## minimal example for CRAN testing
mod<-bass(1:2,1:2,nmcmc=2,nburn=1)
```

`bassBasis` *Bayesian Adaptive Spline Surfaces (BASS) with basis decomposition of response*

Description

Fits a BASS model to basis coefficients under the specified basis.

Usage

```
bassBasis(dat, n.cores = 1, parType = "fork", ...)
```

Arguments

`dat`	list that includes elements `xx`, `n.pc` (number of basis functions), `basis` (dimension m x `n.pc`), `newy` (dimension `n.pc` x n), `trunc.error` (optional truncation error with dimension n x m), `y.m` (vector mean removed before basis decomposition with dimension m), `y.s` (vector sd scaled before basis decomposition with dimension m). See the documentation of `bassPCA` for more details.
`n.cores`	integer number of cores (threads) to use
`parType`	either "fork" or "socket". Forking is typically faster, but not compatible with Windows. If `n.cores==1`, `parType` is ignored.
`...`	arguements to be passed to `bass` function calls.

Details

Under a user defined basis decomposition, fits a bass model to each PCA basis coefficient independently, `bass(dat$xx,dat$newy[i,],...)` for `i in 1 to n.pc`, possibly in parallel. The basis does not need to be orthogonal, but independent modeling of basis coefficients should be sensible.

Value

An object of class 'bassBasis' with two elements:

`mod.list`	list (of length `n.pc`) of individual bass models
`dat`	same as dat above

See Also

predict.bassBasis for prediction and sobolBasis for sensitivity analysis.

Examples

```
## Not run:
  ## simulate data (Friedman function)
f<-function(x){
  10*sin(pi*x[,1]*x[,2])+20*(x[,3]-.5)^2+10*x[,4]+5*x[,5]
}
## simulate data (Friedman function with first variable as functional)
sigma<-.1 # noise sd
n<-500 # number of observations
nfunc<-50 # size of functional variable grid
xfunc<-seq(0,1,length.out=nfunc) # functional grid
x<-matrix(runif(n*9),n,9) # 9 non-functional variables, only first 4 matter
X<-cbind(rep(xfunc,each=n),kronecker(rep(1,nfunc),x)) # to get y
y<-matrix(f(X),nrow=n)+rnorm(n*nfunc,0,sigma)

## fit BASS
library(parallel)
mod<-bassPCA(x,y,n.pc=5,n.cores=min(5,parallel::detectCores()))
plot(mod$mod.list[[1]])
plot(mod$mod.list[[2]])
plot(mod$mod.list[[3]])
plot(mod$mod.list[[4]])
plot(mod$mod.list[[5]])

hist(mod$dat$trunc.error)

## prediction
npred<-100
xpred<-matrix(runif(npred*9),npred,9)
Xpred<-cbind(rep(xfunc,each=npred),kronecker(rep(1,nfunc),xpred))
ypred<-matrix(f(Xpred),nrow=npred)
pred<-predict(mod,xpred,mcmc.use=1:1000) # posterior predictive samples (each is a curve)
matplot(ypred,apply(pred,2:3,mean),type='l',xlab='observed',ylab='mean prediction')
abline(a=0,b=1,col=2)
matplot(t(ypred),type='l') # actual
matplot(t(apply(pred,2:3,mean)),type='l') # mean prediction

## sensitivity
sens<-sobolBasis(mod,int.order = 2,ncores = max(parallel::detectCores()-2,1),
                 mcmc.use=1000) # for speed, only use a few samples
plot(sens)
```

```
## calibration
x.true<-runif(9,0,1) # what we are trying to learn
yobs<-f(cbind(xfunc,kronecker(rep(1,nfunc),t(x.true)))) +
  rnorm(nfunc,0,.1) # calibration data (with measurement error)
plot(yobs)

cal<-calibrate.bassBasis(y=yobs,mod=mod,
                         discrep.mean=rep(0,nfunc),
                         discrep.mat=diag(nfunc)[,1:2]*.0000001,
                         sd.est=.1,
                         s2.df=50,
                         s2.ind=rep(1,nfunc),
                         meas.error.cor=diag(nfunc),
                         bounds=rbind(rep(0,9),rep(1,9)),
                         nmcmc=10000,
                         temperature.ladder=1.05^(0:30),type=1)

nburn<-5000
uu<-seq(nburn,10000,5)

pairs(rbind(cal$theta[uu,1,],x.true),col=c(rep(1,length(uu)),2),ylim=c(0,1),xlim=c(0,1))

pred<-apply(predict(mod,cal$theta[uu,1,],nugget = T,trunc.error = T,
    mcmc.use = cal$ii[uu]),3,function(x) diag(x)+rnorm(length(uu),0,sqrt(cal$s2[uu,1,1])))
qq<-apply(pred,2,quantile,probs=c(.025,.975))
matplot(t(qq),col='lightgrey',type='l')
lines(yobs,lwd=3)



## End(Not run)
## minimal example for CRAN testing
mod<-bassPCA(1:10,matrix(1:20,10),n.pc=2,nmcmc=2,nburn=1)
```

---

bassPCA *Bayesian Adaptive Spline Surfaces (BASS) with PCA decomposition of response*

Description

Decomposes a multivariate or functional response onto a principal component basis and fits a BASS model to each basis coefficient.

Usage

```
bassPCA(
  xx = NULL,
  y = NULL,
```

```
  dat = NULL,
  n.pc = NULL,
  perc.var = 99,
  n.cores = 1,
  parType = "fork",
  center = T,
  scale = F,
  ...
)
```

Arguments

xx	a data frame or matrix of predictors with dimension n x p. Categorical predictors should be included as factors.
y	a response matrix (functional response) with dimension n x m.
dat	optional (for more control) list with elements xx (same as above), y (same as above), n.pc (number of principal components used), basis (principal components with dimension m x n.pc), newy (reduced dimension y with dimension n.pc x n), trunc.error (optional truncation error with dimension n x m), y.m (vector mean removed before PCA with dimension m), y.s (vector sd scaled before PCA with dimension m). If dat is specified, xx, y and n.pc do not need to be specified.
n.pc	number of principal components to use
perc.var	optionally specify percent of variance to explain instead of n.pc
n.cores	integer number of cores (threads) to use
parType	either "fork" or "socket". Forking is typically faster, but not compatible with Windows. If n.cores==1, parType is ignored.
center	logical whether to subtract the mean before getting the principal components, or else a numeric vector of dimension m for the center to be used
scale	logical whether to divide by the standard deviation before getting the principal components, or else a numeric vector of dimension m for the scale to be used
...	arguements to be passed to bass function calls.

Details

Gets the PCA decomposition of the response y, and fits a bass model to each PCA basis coefficient, `bass(dat$xx,dat$newy[i,],...)` for `i in 1 to n.pc`, possibly in parallel.

Value

An object of class 'bassBasis' with two elements:

mod.list	list (of length n.pc) of individual bass models
dat	same as dat above

See Also

predict.bassBasis for prediction and sobolBasis for sensitivity analysis.

Examples

```
## Not run:
  ## simulate data (Friedman function)
f<-function(x){
  10*sin(pi*x[,1]*x[,2])+20*(x[,3]-.5)^2+10*x[,4]+5*x[,5]
}
## simulate data (Friedman function with first variable as functional)
sigma<-.1 # noise sd
n<-500 # number of observations
nfunc<-50 # size of functional variable grid
xfunc<-seq(0,1,length.out=nfunc) # functional grid
x<-matrix(runif(n*9),n,9) # 9 non-functional variables, only first 4 matter
X<-cbind(rep(xfunc,each=n),kronecker(rep(1,nfunc),x)) # to get y
y<-matrix(f(X),nrow=n)+rnorm(n*nfunc,0,sigma)

## fit BASS
library(parallel)
mod<-bassPCA(x,y,n.pc=5,n.cores=min(5,parallel::detectCores()))
plot(mod$mod.list[[1]])
plot(mod$mod.list[[2]])
plot(mod$mod.list[[3]])
plot(mod$mod.list[[4]])
plot(mod$mod.list[[5]])

hist(mod$dat$trunc.error)

## prediction
npred<-100
xpred<-matrix(runif(npred*9),npred,9)
Xpred<-cbind(rep(xfunc,each=npred),kronecker(rep(1,nfunc),xpred))
ypred<-matrix(f(Xpred),nrow=npred)
pred<-predict(mod,xpred,mcmc.use=1:1000) # posterior predictive samples (each is a curve)
matplot(ypred,apply(pred,2:3,mean),type='l',xlab='observed',ylab='mean prediction')
abline(a=0,b=1,col=2)
matplot(t(ypred),type='l') # actual
matplot(t(apply(pred,2:3,mean)),type='l') # mean prediction

## sensitivity
sens<-sobolBasis(mod,int.order = 2,ncores = max(parallel::detectCores()-2,1),
                 mcmc.use=1000) # for speed, only use a few samples
plot(sens)


## calibration
x.true<-runif(9,0,1) # what we are trying to learn
yobs<-f(cbind(xfunc,kronecker(rep(1,nfunc),t(x.true)))) +
  rnorm(nfunc,0,.1) # calibration data (with measurement error)
plot(yobs)

cal<-calibrate.bassBasis(y=yobs,mod=mod,
                         discrep.mean=rep(0,nfunc),
                         discrep.mat=diag(nfunc)[,1:2]*.0000001,
```

```
                          sd.est=.1,
                          s2.df=50,
                          s2.ind=rep(1,nfunc),
                          meas.error.cor=diag(nfunc),
                          bounds=rbind(rep(0,9),rep(1,9)),
                          nmcmc=10000,
                          temperature.ladder=1.05^(0:30),type=1)

nburn<-5000
uu<-seq(nburn,10000,5)

pairs(rbind(cal$theta[uu,1,],x.true),col=c(rep(1,length(uu)),2),ylim=c(0,1),xlim=c(0,1))

pred<-apply(predict(mod,cal$theta[uu,1,],nugget = T,trunc.error = T,
    mcmc.use = cal$ii[uu]),3,function(x) diag(x)+rnorm(length(uu),0,sqrt(cal$s2[uu,1,1])))
qq<-apply(pred,2,quantile,probs=c(.025,.975))
matplot(t(qq),col='lightgrey',type='l')
lines(yobs,lwd=3)



## End(Not run)
## minimal example for CRAN testing
mod<-bassPCA(1:10,matrix(1:20,10),n.pc=2,nmcmc=2,nburn=1)
```

`calibrate.bassBasis` *Calibrate a bassPCA or bassBasis Model to Data*

Description

Robust modular calibration of a bassPCA or bassBasis emulator using adaptive Metropolis, tempering, and decorrelation steps in an effort to be free of any user-required tuning.

Usage

```
calibrate.bassBasis(
  y,
  mod,
  type,
  sd.est,
  s2.df,
  s2.ind,
  meas.error.cor,
  bounds,
  discrep.mean,
  discrep.mat,
  nmcmc = 10000,
  temperature.ladder = 1.05^(0:30),
  decor.step.every = 100,
```

```
  verbose = T
)
```

Arguments

y	vector of calibration data
mod	a emulator of class bassBasis, whose predictions should match y (i.e., predictions from mod should be the same length as y)
type	one of c(1,2). 1 indicates a model that uses independent truncation error variance, no measurement error correlation, and discrepancy on a basis while type 2 indicates a model that uses a full truncation error covariance matrix, a full measurement error correlation matrix, a fixed full discrepancy covariance matrix, and a fixed discrepancy mean. 1 is for situations where computational efficiency is important (because y is dense), while 2 is only for cases where y is a short vector.
sd.est	vector of prior estimates of measurement error standard deviation
s2.df	vector of degrees of freedom for measurement error sd prior estimates
s2.ind	index vector, same length as y, indicating which sd.est goes with which y
meas.error.cor	a fixed correlation matrix for the measurement errors
bounds	a 2xp matrix of bounds for each input parameter, where p is the number of input parameters.
discrep.mean	discrepancy mean (fixed), only used if type=2
discrep.mat	discrepancy covariance (fixed, for type 2) or basis (if not square, for type 1)
nmcmc	number of MCMC iterations.
temperature.ladder	an increasing vector, all greater than 1, for tempering. Geometric spacing is recommended, so that you have (1+delta)^(0:ntemps), where delta is small (typically between 0.05 and 0.2) and ntemps is the number of elements in the vector.
decor.step.every	integer number of MCMC iterations between decorrelation steps.
verbose	logical, whether to print progress.

Details

Fits a modular Bayesian calibration model, with

$$y = Kw(\theta) + Dv + \epsilon, \;\; \epsilon \sim N(0, \sigma^2 R)$$

$$f(x) = a_0 + \sum_{m=1}^{M} a_m B_m(x)$$

and $B_m(x)$ is a BASS basis function (tensor product of spline basis functions). We use priors

$$a \sim N(0, \sigma^2/\tau(B'B)^{-1})$$

$$M \sim Poisson(\lambda)$$

as well as the priors mentioned in the arguments above.

Value

An object

See Also

predict.bassBasis for prediction and sobolBasis for sensitivity analysis.

Examples

```
## Not run:
  ## simulate data (Friedman function)
f<-function(x){
  10*sin(pi*x[,1]*x[,2])+20*(x[,3]-.5)^2+10*x[,4]+5*x[,5]
}
## simulate data (Friedman function with first variable as functional)
sigma<-.1 # noise sd
n<-500 # number of observations
nfunc<-50 # size of functional variable grid
xfunc<-seq(0,1,length.out=nfunc) # functional grid
x<-matrix(runif(n*9),n,9) # 9 non-functional variables, only first 4 matter
X<-cbind(rep(xfunc,each=n),kronecker(rep(1,nfunc),x)) # to get y
y<-matrix(f(X),nrow=n)+rnorm(n*nfunc,0,sigma)

## fit BASS
library(parallel)
mod<-bassPCA(x,y,n.pc=5,n.cores=min(5,parallel::detectCores()))
plot(mod$mod.list[[1]])
plot(mod$mod.list[[2]])
plot(mod$mod.list[[3]])
plot(mod$mod.list[[4]])
plot(mod$mod.list[[5]])

hist(mod$dat$trunc.error)

## prediction
npred<-100
xpred<-matrix(runif(npred*9),npred,9)
Xpred<-cbind(rep(xfunc,each=npred),kronecker(rep(1,nfunc),xpred))
ypred<-matrix(f(Xpred),nrow=npred)
pred<-predict(mod,xpred,mcmc.use=1:1000) # posterior predictive samples (each is a curve)
matplot(ypred,apply(pred,2:3,mean),type='l',xlab='observed',ylab='mean prediction')
abline(a=0,b=1,col=2)
matplot(t(ypred),type='l') # actual
matplot(t(apply(pred,2:3,mean)),type='l') # mean prediction

## sensitivity
sens<-sobolBasis(mod,int.order = 2,ncores = max(parallel::detectCores()-2,1),
                 mcmc.use=1000) # for speed, only use a few samples
plot(sens)


## calibration
```

```
x.true<-runif(9,0,1) # what we are trying to learn
yobs<-f(cbind(xfunc,kronecker(rep(1,nfunc),t(x.true)))) +
  rnorm(nfunc,0,.1) # calibration data (with measurement error)
plot(yobs)

cal<-calibrate.bassBasis(y=yobs,mod=mod,
                         discrep.mean=rep(0,nfunc),
                         discrep.mat=diag(nfunc)[,1:2]*.0000001,
                         sd.est=.1,
                         s2.df=50,
                         s2.ind=rep(1,nfunc),
                         meas.error.cor=diag(nfunc),
                         bounds=rbind(rep(0,9),rep(1,9)),
                         nmcmc=10000,
                         temperature.ladder=1.05^(0:30),type=1)

nburn<-5000
uu<-seq(nburn,10000,5)

pairs(rbind(cal$theta[uu,1,],x.true),col=c(rep(1,length(uu)),2),ylim=c(0,1),xlim=c(0,1))

pred<-apply(predict(mod,cal$theta[uu,1,],nugget = T,trunc.error = T,
    mcmc.use = cal$ii[uu]),3,function(x) diag(x)+rnorm(length(uu),0,sqrt(cal$s2[uu,1,1])))
qq<-apply(pred,2,quantile,probs=c(.025,.975))
matplot(t(qq),col='lightgrey',type='l')
lines(yobs,lwd=3)



## End(Not run)
## minimal example for CRAN testing
mod<-bassPCA(1:10,matrix(1:20,10),n.pc=2,nmcmc=2,nburn=1)
```

`plot.bass` *BASS Plot Diagnostics*

Description

Generate diagnostic plots for BASS model fit.

Usage

```
## S3 method for class 'bass'
plot(x, quants = c(0.025, 0.975), ...)
```

Arguments

`x`	a bass object.
`quants`	quantiles for intervals, if desired. NULL if not desired.
`...`	graphical parameters.

Details

The first two plots are trace plots for diagnosing convergence. The third plot is posterior predicted vs observed, with intervals for predictions. The fourth plot is a histogram of the residuals (of the posterior mean model), with a red curve showing the assumed Normal density (using posterior mean variance). If `bass` was run with `save.yhat = FALSE`, the third and fourth plots are omitted.

See Also

bass, predict.bass, sobol

Examples

```
# See examples in bass documentation.
```

---

plot.bassBasis *BASS Plot Diagnostics*

Description

Generate diagnostic plots for BASS model fit.

Usage

```
## S3 method for class 'bassBasis'
plot(x, quants = c(0.025, 0.975), pred = T, ...)
```

Arguments

`x`	a `bassBasis` object.
`quants`	quantiles for intervals, if desired. NULL if not desired.
`pred`	logical, should predictive performance be plotted?
`...`	graphical parameters.

Details

The first two plots are trace plots for diagnosing convergence. The third plot is posterior predicted vs observed, with intervals for predictions. The fourth plot is a histogram of the residuals (of the posterior mean model). If `pred = FALSE`, the third and fourth plots are omitted.

See Also

bassBasis, bassPCA, predict.bassBasis, sobolBasis

Examples

```
# See examples in bassBasis documentation.
```

plot.bassSob *Plot BASS sensitivity indices*

Description

Generate plots for sensitivity analysis of BASS.

Usage

```
## S3 method for class 'bassSob'
plot(x, ...)
```

Arguments

x	a bassSob object, returned from sobol.
...	graphical parameters.

Details

If func.var in the call to sobol was NULL, this returns boxplots of sensitivity indices and total sensitivity indices. If there were functional variables, they are labeled with letters alphabetically. Thus, if I fit a model with 4 categorical/continuous inputs and 2 functional inputs, the functional inputs are labeled a and b. If func.var was not NULL, then posterior mean functional sensitivity indices are plotted, along with the functional partitioned variance. Variables and interactions that are excluded did not explain any variance.

See Also

bass, predict.bass, sobol

Examples

```
# See examples in bass documentation.
```

predict.bass *BASS Prediction*

Description

Predict function for BASS. Outputs the posterior predictive samples based on the specified MCMC iterations.

Usage

```
## S3 method for class 'bass'
predict(
  object,
  newdata,
  newdata.func = NULL,
  mcmc.use = NULL,
  verbose = FALSE,
  nugget = FALSE,
  ...
)
```

Arguments

object	a fitted model, output from the bass function.
newdata	a matrix of new input values at which to predict. The columns should correspond to the same variables used in the bass function.
newdata.func	a matrix of new values of the functional variable. If none, the same values will be used as in the training data.
mcmc.use	a vector indexing which MCMC iterations to use for prediction.
verbose	logical; should progress be displayed?
nugget	logical; should predictions include error? If FALSE, predictions will be for mean.
...	further arguments passed to or from other methods.

Details

Efficiently predicts when two MCMC iterations have the same basis functions (but different weights).

Value

If model output is a scalar, this returns a matrix with the same number of rows as newdata and columns corresponding to the the MCMC iterations mcmc.use. These are samples from the posterior predictive distribution. If model output is functional, this returns an array with first dimension corresponding to MCMC iteration, second dimension corresponding to the rows of newdata, and third dimension corresponding to the rows of newdata.func.

See Also

bass for model fitting and sobol for sensitivity analysis.

Examples

```
# See examples in bass documentation.
```

predict.bassBasis *BASS Prediction*

Description

Predict function for BASS. Outputs the posterior predictive samples based on the specified MCMC iterations.

Usage

```
## S3 method for class 'bassBasis'
predict(
  object,
  newdata,
  mcmc.use = NULL,
  trunc.error = FALSE,
  nugget = T,
  n.cores = 1,
  parType = "fork",
  ...
)
```

Arguments

object	a fitted model, output from the bass function.
newdata	a matrix of new input values at which to predict. The columns should correspond to the same variables used in the bassBasis or bassPCA functions.
mcmc.use	a vector indexing which MCMC iterations to use for prediction.
trunc.error	logical, use basis truncation error when predicting?
nugget	logical, use individual bass nugget variances when predicting?
n.cores	number of cores, though 1 is often the fastest.
parType	either "fork" or "socket". Forking is typically faster, but not compatible with Windows. If n.cores==1, parType is ignored.
...	further arguments passed to or from other methods.

Details

Prediction combined across bass models.

Value

An array with first dimension corresponding to MCMC iteration, second dimension corresponding to the rows of newdata, and third dimension corresponding to the multivariate/functional response.

See Also

bassPCA and bassBasis for model fitting and sobolBasis for sensitivity analysis.

Examples

```
# See examples in bass documentation.
```

`print.bass` *Print BASS Details*

Description

Print some of the details of a BASS model.

Usage

```
## S3 method for class 'bass'
print(x, ...)
```

Arguments

`x`	a `bass` object, returned from `bass`.
`...`	further arguments passed to or from other methods.

`print.bassBasis` *Print BASS Details*

Description

Print some of the details of a BASS model.

Usage

```
## S3 method for class 'bassBasis'
print(x, ...)
```

Arguments

`x`	a `bassBasis` object, returned from `bassPCA` or `bassBasis`.
`...`	further arguments passed to or from other methods.

sobol *BASS Sensitivity Analysis*

Description

Decomposes the variance of the BASS model into variance due to main effects, two way interactions, and so on, similar to the ANOVA decomposition for linear models. Uses the Sobol' decomposition, which can be done analytically for MARS models.

Usage

```
sobol(
  bassMod,
  prior = NULL,
  prior.func = NULL,
  mcmc.use = NULL,
  func.var = NULL,
  xx.func.var = NULL,
  verbose = TRUE,
  getEffects = FALSE
)
```

Arguments

bassMod	a fitted model output from the bass function.
prior	a list of priors; uniform, truncated mixture of Normals or Ts for continuous; vector of category weights for categorical. Default is uniform over range of data.
prior.func	prior for functional variable. In almost all cases, keep this as the uniform default.
mcmc.use	an integer vector indexing which MCMC iterations to use for sensitivity analysis.
func.var	an integer indicating which functional variable to make sensitivity indices a function of. Disregard if bassMod is non-functional or if scalar sensitivity indices are desired.
xx.func.var	grid for functional variable specified by func.var. Disregard if func.var is not specified. If func.var is specified and xx.func.var not specified, the grid used to fit bass will be used.
verbose	logical; should progress be displayed?
getEffects	logical; should Sobols ANOVA decomposition be computed?

Details

Performs analytical Sobol' decomposition for each MCMC iteration in mcmc.use (each corresponds to a MARS model), yeilding a posterior distribution of sensitivity indices. Can obtain Sobol' indices as a function of one functional variable.

Value

If non-functional (`func.var = NULL`), a list with two elements:

`S`	a data frame of sensitivity indices with number of rows matching the length of `mcmc.use`. The columns are named with a particular main effect or interaction. The values are the proportion of variance in the model that is due to each main effect or interaction.
`T`	a data frame of total sensitivity indices with number of rows matching the length of `mcmc.use`. The columns are named with a particular variable.

Otherwise, a list with four elements:

`S`	an array with first dimension corresponding to MCMC samples (same length as `mcmc.use`), second dimension corresponding to different main effects and interactions (labeled in `names.ind`), and third dimension corresponding to the grid used for the functional variable. The elements of the array are sensitivity indices.
`S.var`	same as `S`, but scaled in terms of total variance rather than percent of variance.
`names.ind`	a vector of names of the main effects and interactions used.
`xx`	the grid used for the functional variable.

See Also

bass for model fitting and predict.bass for prediction.

Examples

```
# See examples in bass documentation.
```

`sobolBasis` *BASS Sensitivity Analysis*

Description

Decomposes the variance of the BASS model into variance due to main effects, two way interactions, and so on, similar to the ANOVA decomposition for linear models. Uses the Sobol' decomposition, which can be done analytically for MARS models.

Usage

```
sobolBasis(
  mod,
  int.order,
  prior = NULL,
  mcmc.use = NULL,
  nind = NULL,
```

```
  n.cores = 1,
  parType = "fork",
  plot = F,
  verbose = T
)
```

Arguments

mod	output from the bassBasis or bassPCA function.
int.order	an integer indicating the highest order of interactions to include in the Sobol decomposition.
prior	a list with the same number of elements as there are inputs to mod. Each element specifies the prior for the particular input. Each prior is specified as a list with elements dist (one of c("normal", "student", "uniform")), trunc (a vector of dimension 2 indicating the lower and upper truncation bounds, taken to be the data bounds if omitted), and for "normal" or "student" priors, mean (scalar mean of the Normal/Student, or a vector of means for a mixture of Normals or Students), sd (scalar standard deviation of the Normal/Student, or a vector of standard deviations for a mixture of Normals or Students), df (scalar degrees of freedom of the Student, or a vector of degrees of freedom for a mixture of Students), and weights (a vector of weights that sum to one for the mixture components, or the scalar 1). If unspecified, a uniform is assumed with the same bounds as are represented in the input to mod.
mcmc.use	an integer indicating which MCMC iteration to use for sensitivity analysis. Defaults to the last iteration.
nind	number of Sobol indices to keep (will keep the largest nind).
n.cores	number of cores to use (nearly linear speedup for adding cores).
parType	either "fork" or "socket". Forking is typically faster, but not compatible with Windows. If n.cores==1, parType is ignored.
plot	logical; whether to plot results.
verbose	logical; print progress.

Details

Performs analytical Sobol' decomposition for each MCMC iteration in mcmc.use (each corresponds to a MARS model), yeilding a posterior distribution of sensitivity indices. Can obtain Sobol' indices as a function of one functional variable.

Value

If non-functional (func.var = NULL), a list with two elements:

S	a data frame of sensitivity indices with number of rows matching the length of mcmc.use. The columns are named with a particular main effect or interaction. The values are the proportion of variance in the model that is due to each main effect or interaction.

T	a data frame of total sensitivity indices with number of rows matching the length of mcmc.use. The columns are named with a particular variable.

Otherwise, a list with four elements:

S	an array with first dimension corresponding to MCMC samples (same length as mcmc.use), second dimension corresponding to different main effects and interactions (labeled in names.ind), and third dimension corresponding to the grid used for the functional variable. The elements of the array are sensitivity indices.
S.var	same as S, but scaled in terms of total variance rather than percent of variance.
names.ind	a vector of names of the main effects and interactions used.

See Also

bassPCA and bassBasis for model fitting and predict.bassBasis for prediction.

Examples

```
# See examples in bass documentation.
```

---

summary.bass *Summarize BASS Details*

Description

Summarize some of the details of a BASS model.

Usage

```
## S3 method for class 'bass'
summary(object, ...)
```

Arguments

object	a bass object, returned from bass.
...	further arguments passed to or from other methods.

summary.bassBasis *Summarize BASS Details*

Description

Summarize some of the details of a BASS model.

Usage

```
## S3 method for class 'bassBasis'
summary(object, ...)
```

Arguments

object	a bassBasis object, returned from bassPCA or bassBasis.
...	further arguments passed to or from other methods.

Index