Package 'coactivity'

November 26, 2024

Type Package

Title What the Package Does (Title Case)

Version 0.1.0

Author Who wrote it

Maintainer The package maintainer <yourself@somewhere.net>

Description More about what it does (maybe more than one line)
Use four spaces when indenting paragraphs within the Description.

License What license is it under?

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Contents

`bassfunc2bass` *Convert functional BASS model to BASS model*

Description

The argument to this function is the output of a `bass()` call when a single functional variable is specified using the `xx.func` argument. Note that the resulting model may not be a valid bass object for some applications, but the resulting model can be passed to `concordance::C_bass()` and related functions.

Usage

```
bassfunc2bass(bfm)
```

Arguments

bfm	an object of class bass, where a functional variable has been specified.

Examples

```
# simulate data (Friedman function with first variable as functional)
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
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_func <- bass(x,y,xx.func=xfunc)

# convert to standard BASS model
mod <- bassfunc2bass(mod_func)

# Estimate C matrix (augmentation approach)
C <- C_bass(mod)
```

---

bassfunc2bass_fixed_t *Extract scalar BASS model from a functional BASS model for fixed functional variable values*

Description

Extracts the BASS model(s) corresponding to fixed values of the functional variable. For each specified \(t \)-value, calculates the contribution of the functional variable to the tensor product basis functions and returns a modified BASS model.

Usage

```
bassfunc2bass_fixed_t(bassfunc, func.use, verbose = FALSE)
```

Arguments

bassfunc	A BASS model object (with functional variables).
func.use	A numeric vector of fixed values for the functional variable \(t \).
verbose	logical; should progress be displayed?

Value

A list of BASS models for each \(t \)-value (or a single model if length(func.use) == 1).

Examples

```
# Simulate bass models for each of the three cases
f<-function(x){
  10*sin(pi*x[,1]*x[,2])+20*(x[,3]-.5)^2+10*x[,4]+5*x[,5]
}
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)

# Scalar response
X <- lhs::randomLHS(500, 5)
y <- f(X)
bassfunc <- bass(x, y, xx.func=xfunc)

# Extract the BASS model for t = 0.5
mod_fixed <- funcbass2bass_fixed_t(mod_func, func.use = 0.5)

# Extract models for multiple t values
mods_fixed <- funcbass2bass_fixed_t(mod_func, func.use = c(0.5, 0.75))
```

`bassPCA2bass_fixed_t` *Extract scalar BASS model from bassPCA model for fixed functional variable values*

Description

Extracts the BASS model(s) corresponding to fixed values of the functional variable. For each specified \(t \)-value, calculates the contribution of the functional variable to the tensor product basis functions and returns a modified BASS model.

Usage

```
bassPCA2bass_fixed_t(bassPCA, func.use, func.true = NULL)
```

Arguments

`bassPCA`	A bassBasis model object (from bassPCA() function).
`func.use`	A numeric vector of fixed values for the functional variable \(t \).
`func.true`	An optional vector of values for the functional variable in bassPCA. Should have length equal to nrow(bassPCAdatbasis).

Details

Since bassPCA doesn't

Value

A list of BASS models for each \(t \)-value (or a single model if length(func.use) == 1).

Examples

```
f<-function(x, t){
10*sin(pi*t*x[1])+20*(x[2]-.5)^2+10*x[3]+5*x[4]
}
XX <- lhs::randomLHS(500, 5)
y1 <- apply(XX, 1, f, t=0.5)
xfunc <- seq(0, 1, length.out=20)
yfunc <- t(apply(XX, 1, f, t=xfunc))

# Fit a bassPCA model
mod3_full <- bassPCA(XX, yfunc)

# Extract univariate model at t = 0.5
mod3 <- bassPCA2bass_fixed_t(mod3_full, 0.5)
```

`build_prior` *Build Prior Method for C_bass and Cfg_bass*

Description

A method for building priors of the form needed in 'C_bass', 'Cfg_bass' and similar functions. For mixture distributions, all arguments (except for lower and upper) should be matrices (with nrow equal to the number of mixture components) rather than vectors.

Usage

```
build_prior(
  dist,
  lower = -Inf,
  upper = Inf,
  mu = NULL,
  sigma = NULL,
  shape1 = NULL,
  shape2 = NULL,
  shape = NULL,
  scale = NULL,
  weights = NULL
)
```

Arguments

`dist`	A vector of length p. Valid entries include "uniform", "normal", "beta", "gamma".
`lower`	A p-vector of lower truncation bounds. '-Inf' is a valid entry.
`upper`	A p-vector of lower truncation bounds. 'Inf' is a valid entry.
`mu`	A p-vector of means (used for normal/truncated normal only)
`sigma`	A p-vector of sds (used for normal/truncated normal only)
`shape1`	A p-vector of shape1 parameters for beta prior
`shape2`	A p-vector of shape2 parameters for beta prior
`shape`	A p-vector of shape parameters for gamma prior
`scale`	A p-vector of scale parameters for gamma prior
`weights`	A vector of mixture weights of the same dimension as dist.

Details

Builds a list for passing to the `coactivity` functions. List contains one component per input variable. The `dist` argument must be passed in full, but all other values can be scalars (and will be reshaped accordingly). Truncation bounds cannot vary by mixture component. See examples below.

Value

a list which can be passed into C_bass or Cfg_bass as a prior.

Examples

```
# standard uniform priors for 5 inputs
build_prior(rep("uniform", 5), lower=0, upper=1)

# truncated normals with different means for each input
mu_vec <- c(0.4, 0.5, 0.3, 0.7, 0.5)
build_prior(rep("normal", 5), lower=0, upper=1,
            mean=mu_vec, sd=0.1)

# A mixture of normals (p=4)
mu_mat = matrix(c(0.4, 0.5, NA,
                  0.5, NA, NA,
                  0.25, 0.5, 0.75,
                  0.4, 0.5, 0.8),
                  ncol=3, byrow=TRUE)
weights_mat = matrix(c(2, 3, 0,
                  1, 0, 0,
                  1, 1, 1,
                  1, 1, 10),
                  ncol=3, byrow=TRUE)
build_prior(matrix("normal", nrow=4, ncol=3),
            lower=-Inf, upper=Inf,
            mean=mu_mat, sd=0.1,
            weights=weights_mat)
```

Cfg_bass *Estimate the Constantine Matrix with BASS*

Description

Closed form estimator of the C matrix using a BASS model

Usage

```
Cfg_bass(
  mod1,
  mod2,
  prior = NULL,
  mcmc.use = NULL,
  use_native_scale = FALSE,
  func.use = NULL,
```

```
  pca_method = 2,
  verbose = FALSE
)
```

Arguments

mod1	a fitted BASS model for the first function.
mod2	a fitted BASS model for the second function.
prior	a list, like one returned by the build_prior() function. See the documentation for details.
mcmc.use	a two-column matrix of MCMC indices corresponding to mod1 and mod2 respectively.
use_native_scale	logical (default 'TRUE'). Determines the scale of the inputs for computing the \(C \)-matrix. When 'TRUE', the \(C \)-matrix is computed on the original (native) scale of the input variables. When 'FALSE', the \(C \)-matrix corresponds to the inputs normalized to the \([0, 1]\) range, as used internally by BASS. This also affects derived quantities, such as activity scores..
func.use	a vector indicating which values of the functional variable to compute C for, if applicable
pca_method	takes value 1 or 2 to indicate which method should be used for estimating C. See details.
verbose	logical; should progress be displayed?

Details

The C matrices are computed using inputs which are scaled to (0, 1). The use_native_scale flag indicates whether the C matrix should be transformed to the native space before returning.

The func.use argument is used only if mod is fit to functional response data (using bass() with xx.func specified or by using bassPCA). In the latter case, the functional input is assumed to be between 0 and 1.

The pca_method argument is used only when mod has class bassBasis. When set to 1 (the default), the decomposition theorem is used to estimate C. Otherwise, the model list is converted to a single model using lcbass2bass and C is found directly. Method 1 is typically faster (especially for multiple func.use), but method 2 gives more flexibility in the choice of prior.

Value

A list representing the posterior distribution of the Constantine matrix.

Examples

```
# FRIEDMAN FUNCTION AND SECONDARY FUNCTION
# First input is treated as functional
# Use p=5, so there is one inert variable
f <- function(x, t) {
  10 * sin(pi * t * x[1]) + 20 * (x[2] - 0.5)^2 + 10 * x[3] + 5 * x[4]
}
g <- function(x, t) {
  5 * sin(2 * pi * t * x[1]) + 15 * (x[3] - 0.3)^2 + 8 * x[4] + 3 * x[5]
}
```

```
# ==========================================
#        GENERATE DATA
# ==========================================
XX <- lhs::randomLHS(500, 5)
y1_f <- apply(XX, 1, f, t = 0.5)
y1_g <- apply(XX, 1, g, t = 0.5)
xfunc <- seq(0, 1, length.out = 20)
yfunc_f <- t(apply(XX, 1, f, t = xfunc))
yfunc_g <- t(apply(XX, 1, g, t = xfunc))

Xtest <- lhs::randomLHS(100, 5)
ytest_f <- apply(Xtest, 1, f, t = 0.5)
ytest_g <- apply(Xtest, 1, g, t = 0.5)

# ==========================================
#        CASE 1: Univariate BASS
# ==========================================
mod1_f <- bass(XX, y1_f)
mod1_g <- bass(XX, y1_g)
C1_fg <- Cfg_bass(mod1_f, mod1_g)

# ==========================================
#      CASE 2: Augmented BASS (fixed t)
# ==========================================
mod2_full_f <- bass(XX, yfunc_f, xx.func = xfunc)
mod2_full_g <- bass(XX, yfunc_g, xx.func = xfunc)
mod2_f <- bassfunc2bass_fixed_t(mod2_full_f, 0.5)
mod2_g <- bassfunc2bass_fixed_t(mod2_full_g, 0.5)
C2_fg <- Cfg_bass(mod2_f, mod2_g)
C2b_fg <- Cfg_bass(mod2_full_f, mod2_full_g, func.use = 0.5)

# ==========================================
#      CASE 3: PCA BASS (fixed t)
# ==========================================
mod3_full_f <- bassPCA(XX, yfunc_f)
mod3_full_g <- bassPCA(XX, yfunc_g)
mod3_f <- bassPCA2bass_fixed_t(mod3_full_f, 0.5)
mod3_g <- bassPCA2bass_fixed_t(mod3_full_g, 0.5)
C3_fg <- Cfg_bass(mod3
```

C_bass *Estimate the Constantine Matrix with BASS*

Description

Closed form estimator of the C matrix using a BASS model

Usage

```
C_bass(
  mod,
  prior = NULL,
  mcmc.use = NULL,
```

```
  use_native_scale = FALSE,
  func.use = NULL,
  pca_method = 2,
  verbose = FALSE
)
```

Arguments

mod	a fitted BASS model. The output of the bass() or bassPCA() functions.
prior	a list, like one returned by the build_prior() function. See the documentation for details.
mcmc.use	a vector of indices telling which mcmc draws to use
use_native_scale	logical (default 'TRUE'). Determines the scale of the inputs for computing the \(C \)-matrix. When 'TRUE', the \(C \)-matrix is computed on the original (native) scale of the input variables. When 'FALSE', the \(C \)-matrix corresponds to the inputs normalized to the \([0, 1]\) range, as used internally by BASS. This also affects derived quantities, such as activity scores..
func.use	a vector indicating which values of the functional variable to compute C for, if applicable
pca_method	takes value 1 or 2 to indicate which method should be used for estimating C. See details.
verbose	logical; should progress be displayed?

Details

The C matrices are computed using inputs which are scaled to (0, 1). The use_native_scale flag indicates whether the C matrix should be transformed to the native space before returning.

The func.use argument is used only if mod is fit to functional response data (using bass() with xx.func specified or by using bassPCA). In the latter case, the functional input is assumed to be between 0 and 1.

The pca_method argument is used only when mod has class bassBasis. When set to 1 (the default), the decomposition theorem is used to estimate C. Otherwise, the model list is converted to a single model using lcbass2bass and C is found directly. Method 1 is typically faster (especially for multiple func.use), but method 2 gives more flexibility in the choice of prior.

Value

A list representing the posterior distribution of the Constantine matrix.

Examples

```
# FRIEDMAN FUNCTION
# First input is treated as functional
# Use p=5, so there is one inert variable
f <- function(x, t) {
  10 * sin(pi * t * x[1]) + 20 * (x[2] - 0.5)^2 + 10 * x[3] + 5 * x[4]
}

# ========================================
#        GENERATE DATA
# ========================================
```

```
XX <- lhs::randomLHS(500, 5)
y1 <- apply(XX, 1, f, t = 0.5)
xfunc <- seq(0, 1, length.out = 20)
yfunc <- t(apply(XX, 1, f, t = xfunc))

Xtest <- lhs::randomLHS(100, 5)
ytest <- apply(Xtest, 1, f, t = 0.5)

# ========================================
#        CASE 1: Univariate BASS
# ========================================
mod1 <- bass(XX, y1)
C1 <- C_bass(mod1)

# ========================================
#      CASE 2: Augmented BASS (fixed t)
# ========================================
mod2_full <- bass(XX, yfunc, xx.func = xfunc)
mod2 <- bassfunc2bass_fixed_t(mod2_full, 0.5)
C2 <- C_bass(mod2)
C2b <- C_bass(mod2_full, func.use=0.5)

# ========================================
#      CASE 3: PCA BASS (fixed t)
# ========================================
mod3_full <- bassPCA(XX, yfunc)
mod3 <- bassPCA2bass_fixed_t(mod3_full, 0.5)
C3 <- C_bass(mod3)
C3b <- C_bass(mod3_full, func.use=0.5)
```

`gradient_bass` *Gradient evaluations of a BASS model*

Description

I THINK THIS IS WRONG AT THE MOMENT.

Usage

```
gradient_bass(object, newdata, mcmc.use = NULL, verbose = FALSE)
```

Arguments

`object`	A fitted model, output from the bass function
`newdata`	A matrix of input locations at which to evaluate the gradient. The columns should correspond to the same variables used in the `bass` function.
`mcmc.use`	A vector indexing which MCMC iterations to be used.
`verbose`	logical; should progress be displayed

Details

This function behaves similarly to the predict.bass method, except that it returns predictions for the gradient of f.

Value

An array of gradient evaluations. First dimension indexes MCMC iteration, second dimension indexes input location, third dimension indexes the input variable.

Examples

```
#' # simulate data (Friedman function with first variable as functional)
n <- 500
p <- 2
X <- matrix(runif(n*p), ncol=p)
y <- apply(X, 1, duqling::dms_additive)
mod <- bass(X, y)

Xnew <-  matrix(runif(100*p), ncol=p)
gradients <- gradient_bass(mod, Xnew)
```

---

hello *Hello, World!*

Description

Prints 'Hello, world!'.

Usage

```
hello()
```

Examples

```
hello()
```

K_bass *Estimate the Zahm Matrix with BASS*

Description

Closed form estimator of the Zahm matrix using a BASS model

Usage

```
K_bass(
  mod,
  prior = NULL,
  prior_func = function(tt) dunif(tt),
  mcmc.use = NULL,
  use_native_scale = FALSE,
  func.use = NULL,
  func.true = NULL,
  pca_method = 2,
  verbose = FALSE
)
```

Arguments

mod	a fitted functional BASS model, or a list of univariate bass models. The output of the bass() or bassPCA() functions.
prior	a list, like one returned by the build_prior() function. See the documentation for details.
prior_func	prior for the functional variable. Either a function over 0 to 1 (assumed to integrate to 1) or a vector with the same length as func.use.
mcmc.use	a vector of indices telling which mcmc draws to use
use_native_scale	logical (default 'TRUE'). Determines the scale of the inputs for computing the \(C \)-matrix. When 'TRUE', the \(C \)-matrix is computed on the original (native) scale of the input variables. When 'FALSE', the \(C \)-matrix corresponds to the inputs normalized to the \([0, 1]\) range, as used internally by BASS. This also affects derived quantities, such as activity scores..
func.use	a vector indicating which values of the functional variable to compute C for, if applicable
func.true	An optional vector of values for the functional variable in bassPCA. Should have length equal to nrow(bassPCAdatbasis).
pca_method	takes value 1 or 2 to indicate which method should be used for estimating C. Ignored unless class(mod) == "bassBasis".
verbose	logical; should progress be displayed?

Details

The K matrix is C(t) integrated over t with respect to a prior rho_t. In practice, this is equivalent to the Zahm matrix for multivariate response with a diagonal R.

Value

A list representing the posterior distribution of the Constantine matrix.

Examples

```
# FRIEDMAN FUNCTION
# First input is treated as functional
# Use p=5, so there is one inert variable
f <- function(x, t) {
  10 * sin(pi * t * x[1]) + 20 * (x[2] - 0.5)^2 + 10 * x[3] + 5 * x[4]
}

# ========================================
#        GENERATE DATA
# ========================================
XX <- lhs::randomLHS(500, 5)
y1 <- apply(XX, 1, f, t = 0.5)
xfunc <- seq(0, 1, length.out = 20)
yfunc <- t(apply(XX, 1, f, t = xfunc))

Xtest <- lhs::randomLHS(100, 5)
ytest <- apply(Xtest, 1, f, t = 0.5)

# ========================================
```

```
#        CASE 1: Univariate BASS
# ========================================
mod1 <- bass(XX, y1)
C1 <- C_bass(mod1)

# ========================================
#      CASE 2: Augmented BASS (fixed t)
# ========================================
mod2_full <- bass(XX, yfunc, xx.func = xfunc)
mod2 <- bassfunc2bass_fixed_t(mod2_full, 0.5)
C2 <- C_bass(mod2)
C2b <- C_bass(mod2_full, func.use=0.5)

# ========================================
#      CASE 3: PCA BASS (fixed t)
# ========================================
mod3_full <- bassPCA(XX, yfunc)
mod3 <- bassPCA2bass_fixed_t(mod3_full, 0.5)
C3 <- C_bass(mod3)
C3b <- C_bass(mod3_full, func.use=0.5)
```

`lcbass2bass` *Convert a linear combination of BASS models to a single BASS model*

Description

A linear combination of BASS models is also a BASS model. This function takes a list of BASS models (all with the same data matrix xx.des) and returns the resulting linear combination as a new BASS model. One useful application of this function is to convert bassPCA to bass for a fixed time point. Does not currently work for bass models with functional or categorical inputs.

Usage

```
lcbass2bass(
  mod_list,
  weights = rep(1, length(mod_list)),
  offset = 0,
  yy = NULL,
  mcmc.use = NULL
)
```

Arguments

`mod_list`	A list of bass models.
`weights`	An optional vector of weights.
`offset`	This value is added to the intercept of the resulting model.
`yy`	The data vector. When unspecified, y is taken as the linear combination of responses for the individual bass models.
`mcmc.use`	set of indices telling which mcmc draws to use.

Examples

```
#' # simulate data (Friedman function with first variable as functional)
n <- 500
p <- 2
X <- matrix(runif(n*p), ncol=p)
y1 <- apply(X, 1, duqling::dms_additive)
y2 <- apply(X, 1, duqling::dms_harmonic)
z <- y1 + y2

# Fit bass models
mod1 <- bass(X, y1)
mod2 <- bass(X, y2)
mod_list <- list(mod1, mod2)

# Combine models
mod <- lcbass2bass(mod_list, yy=z)

# Get predictions
yhat1 <- colMeans(predict(mod1, X))
yhat2 <- colMeans(predict(mod2, X))
yhat <- colMeans(predict(mod, X))

plot(yhat1 + yhat2, yhat)
```

Z_bass *Expected gradient with BASS*

Description

Closed form estimator of the expected value of the gradient of a function

Usage

```
Z_bass(
  mod,
  prior = NULL,
  mcmc.use = NULL,
  use_native_scale = FALSE,
  func.use = NULL,
  verbose = FALSE
)
```

Arguments

mod	a fitted BASS model. The output of the bass() or bassPCA() functions.
prior	a list, like one returned by the build_prior() function. See the documentation for details.
mcmc.use	a vector of indices telling which mcmc draws to use

use_native_scale
: logical (default 'TRUE'). Determines the scale of the inputs for computing the \(C \)-matrix. When 'TRUE', the \(C \)-matrix is computed on the original (native) scale of the input variables. When 'FALSE', the \(C \)-matrix corresponds to the inputs normalized to the \([0, 1]\) range, as used internally by BASS. This also affects derived quantities, such as activity scores..

func.use
: a vector indicating which values of the functional variable to compute C for, if applicable

verbose
: Doesn't do anything currently.

Details

Returns the expected value of the gradient of a BASS model.The use_native_scale flag indicates whether the C matrix should be transformed to the native space before returning.

Value

A list representing the posterior distribution of the Constantine matrix.

Examples

```
# FRIEDMAN FUNCTION
# First input is treated as functional
# Use p=5, so there is one inert variable
f <- function(x, t) {
  10 * sin(pi * t * x[1]) + 20 * (x[2] - 0.5)^2 + 10 * x[3] + 5 * x[4]
}

# ========================================
#        GENERATE DATA
# ========================================
XX <- lhs::randomLHS(500, 5)
y1 <- apply(XX, 1, f, t = 0.5)
mod <- bass(XX, y1)
Z <- Z_bass(mod)
```

Index