Package ‘denseFLMM’

April 19, 2018

Type Package

Title Functional Linear Mixed Models for Densely Sampled Data

Version 0.1.2

Author Sonja Greven, Jona Cederbaum

Maintainer Jona Cederbaum <Jona.Cederbaum@gmail.com>

Description Estimation of functional linear mixed models for densely sampled data based on functional principal component analysis.

License GPL-2

LazyData TRUE

Depends R (>= 3.3), mgcv (>= 1.8-12)

Imports methods, parallel, MASS, Matrix, mvtnorm

Suggests

Collate ‘denseFLMM.R’

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-04-19 03:25:24 UTC

R topics documented:

denseFLMM ....................................................... 2

Index 8
Functional Linear Mixed Models for Densely Sampled Data

Description

Estimation of functional linear mixed models (FLMMs) for functional data sampled on equal grids based on functional principal component analysis. The implemented models are special cases of the general FLMM

\[ Y_i(t_d) = \mu(t_d, x_i) + z_i^T U(t_d) + \epsilon_i(t_d), \quad i = 1, \ldots, n, \quad d = 1, \ldots, D, \]

with \( Y_i(t_d) \) the value of the response of curve \( i \) at observation point \( t_d \), \( \mu(t_d, x_i) \) is a mean function, which may depend on covariates \( x_i \) and needs to be estimated beforehand. \( z_i \) is a covariate vector, which is multiplied with the vector of functional random effects \( U(t_d) \). Usually, the functional random effects will additionally include a smooth error term which is a functional random intercept with a special structure that captures deviations from the mean which are correlated along the support of the functions. In this case, the last block of \( z_i \) corresponds to an indicator vector of indicators for each curve and the last block in \( U(t) \) consists of curve-specific functional random effects. \( \epsilon_i(t_d) \) is independent and identically distributed white noise measurement error with homoscedastic, constant variance.

The vector-valued functional random effects can be subdivided into \( H \) independent blocks of functional random effects

\[ U(t_d) = (U_1(t_d)^T, \ldots, U_H(t_d)^T)^T, \]

with \( U_g(t_d) \) and \( U_h(t_d) \) independent for \( g \neq h \). Each block \( U_h(t_d) \) further contains \( L_{U_h} \) independent copies \( U_{gl}(t_d), \quad l = 1, \ldots, L_{U_h} \), of a vector-valued stochastic process with \( \rho_{U_h} \) vector components \( U_{gls}(t_d), \quad s = 1, \ldots, \rho_{U_h} \). The total number of functional random effects then amounts to \( q = \sum_{h=1}^H L_{U_h} \rho_{U_h} \).

The code implements a very general functional linear mixed model for \( n \) curves observed at \( D \) grid points. Grid points are assumed to be equidistant and so far no missings are assumed. The curves are assumed to be centered. The functional random effects for each grouping factor are assumed to be correlated (e.g., random intercept and slope curves). The code can handle group-specific functional random effects including group-specific smooth errors. Covariates are assumed to be standardized.

Usage

denseFLMM(Y, gridpoints = 1:ncol(Y), Zlist = NA, G = NA, Lvec = NA,
          groups = matrix(1, nrow(Y), 1), Zvars, L = NA, NPC = NA,
          smooth = FALSE, bf = 10, smoothalg = "gamm")

Arguments

\( Y \) \( n \times D \) matrix of \( n \) curves observed on \( D \) grid points. \( Y \) is assumed to be centered by its mean function.
gridpoints vector of grid points along curves, corresponding to columns of $Y$. Defaults to matrix(1, nrow(Y), 1).

$Z_{list}$ list of length $H$ of $p^{Ug}$ design matrices $Z_{s^{Ug}}$, $g = 1, \ldots, H$, $s = 1, \ldots, p^{Ug}$. Needed instead of $Z_{vars}$ and groups if group-specific functional random effects are present. Defaults to NA, then $Z_{vars}$ and groups needed.

$G$ number of grouping factors not used for estimation of error variance. Needed if $Z_{list}$ is used instead of $Z_{vars}$ and groups. Defaults to NA.

$L_{vec}$ vector of length $H$ containing the number of levels for each grouping factor. Needed if $Z_{list}$ is used instead of $Z_{vars}$ and groups. Defaults to NA.

$groups$ $n \times G$ matrix with $G$ grouping factors for the rows of $Y$, where $G$ denotes the number of random grouping factors not used for the estimation of the error variance. Defaults to groups = matrix(1, nrow(Y), 1). Set to NA when $Z_{list}$ is used to specify group-specific functional random effects.

$Z_{vars}$ list of length $G$ with $n \times p^{Ug}$ matrices of random variables for grouping factor $g$, where $G$ denotes the number of random grouping factors not used for the estimation of the error variance. Random curves for each grouping factor are assumed to be correlated (e.g., random intercept and slope). Set to NA when $Z_{list}$ is used to specify group-specific functional random effects.

$L$ pre-specified level of variance explained (between 0 and 1), determines number of functional principal components.

$NPC$ vector of length $H$ with number of functional principal components to keep for each functional random effect. Overrides $L$ if not NA. Defaults to NA.

$smooth$ TRUE to add smoothing of the covariance matrices, otherwise covariance matrices are estimated using least-squares. Defaults to FALSE.

$bf$ number of marginal basis functions used for all smooths. Defaults to $bf = 10$.

$smoothalg$ smoothing algorithm used for covariance smoothing. Available options are "gamm", "gamGCV", "ggamREML", "bamGCV", "bamREML", and "bamfREML". "gamm" uses REML estimation based on function gamm in R-package mgcv. "gamGCV" and "ggamREML" use GCV and REML estimation based on function gam in R-package mgcv, respectively. "bamGCV", "bamREML", and "bamfREML" use GCV, REML, and a fast REML estimation based on function bam in R-package mgcv, respectively. Defaults to "gamm".

Details

The model fit for centered curves $Y_i(\cdot)$ is

$$Y = ZU + \epsilon,$$

with $Y = [Y_i(t_d)]_{i=1,\ldots,n,d=1,\ldots,D}$, $Z$ consisting of $H$ blocks $Z_{Ug}$ for $H$ grouping factors, $Z = [Z^{U_1} \ldots Z^{U_H}]$, and each $Z_{Ug} = [Z_1^{Ug} \ldots Z_{p^{Ug}}^{Ug}]$. $U$ is row-wise divided into blocks $U_1, \ldots, U_H$, corresponding to $Z$.

In case no group-specific functional random effects are specified, column $j$ in $Z_{s^{Ug}}$, $s = 1, \ldots, p^{Ug}$, is assumed to be equal to the $s$th variable (column) in $Z_{vars}[g]$ times an indicator for the $j$th level of grouping factor $g$, $g = 1, \ldots, G$.

Note that $G$ here denotes the number of random grouping factors not used for the estimation of the
error variance, i.e., all except the smooth error term(s). The total number of grouping factors is denoted by $H$.

The estimated eigenvectors and eigenvalues are rescaled to ensure that the approximated eigenfunctions are orthonormal with respect to the $L^2$-inner product.

The estimation of the error variance takes place in two steps. In case of smoothing (smooth = TRUE), the error variance is first estimated as the average difference of the raw and the smoothed diagonal values. In case no smoothing is applied, the estimated error variance is zero. Subsequent to the eigen decomposition and selection of the eigenfunctions to keep for each grouping factor, the estimated error variance is recalculated in order to capture the left out variability due to the truncation of the infinite Karhunen-Loeve expansions.

**Value**

The function returns a list containing the input arguments $Y$, gridpoints, groups, Zvars, L, smooth, bf, and smoothalg. It additionally contains:

- $Zlist$ either the input argument $Zlist$ or if set to NA, the computed list of list of design matrices $Z^U_g$, $g = 1, \ldots, H$, $s = 1, \ldots, \rho^U_s$.
- $G$ either the input argument $G$ or if set to NA, the computed number of random grouping factors $G$ not used for the estimation of the error variance.
- $lvec$ either the input argument $lvec$ or if set to NA, the computed vector of length $H$ containing the number of levels for each grouping factor (including the smooth error(s)).
- $np$ vector of length $H$ of number of random effects for each grouping factor (including the smooth error(s)).
- $phi$ list of length $H$ of $D_x N^U_g$ matrices containing the $N^U_g$ functional principal components kept per grouping factor (including the smooth error(s)).
- $sigma2$ estimated measurement error variance $\sigma^2$.
- $nu$ list of length $H$ of $N^U_g x 1$ vectors of estimated eigenvalues $\nu^U_g$.
- $xi$ list of length $H$ of $L^U_g x N^U_g$ matrices containing the predicted random basis weights. Within matrices, basis weights are ordered corresponding to the ordered levels of the grouping factors, e.g., a grouping factor with levels 4, 2, 3, 1 ($L^U_g = 4$) will result in rows in $xi[[g]]$ corresponding to levels 1, 2, 3, 4.
- $totvar$ total average variance of the curves.
- $exvar$ level of variance explained by the selected functional principal components (+ error variance).

**Author(s)**

Sonja Greven, Jona Cederbaum

**See Also**

For the estimation of functional linear mixed models for irregularly or sparsely sampled data based on functional principal component analysis, see function sparseFLMM in package sparseFLMM.
Examples

# fit model with group-specific functional random intercepts for two groups
# and a non group-specific smooth error, i.e., G = 2, H = 1.

set.seed(123)

require(mvtnorm)
require(Matrix)

class_nr <- 2 # number of groups

c <- list(c(0.5, 0.3), c(1, 0.4), c(2)) # eigenvalues

tau_square <- 2.5e-05 # error variance

NPCs <- c(rep(2, 2), 1) # number of eigenfunctions

Lvec <- c(rep(2, 2), 480) # number of levels

G <- 2 # number of functional random effects not used for the estimation of the error variance
g <- 1 # number of functional random effects

grid_points <- seq(from = 0, to = 1, length = 100) # grid points

funB1 <- function(k, t)
{
  if(k == 1)
    out <- sqrt(2) * sin(2 * pi * t)
  if(k == 2)
    out <- sqrt(2) * cos(2 * pi * t)
  out
}

funB2 <- function(k, t)
{
  if(k == 1)
    out <- sqrt(7) * (20 * t^3 - 30 * t^2 + 12 * t - 1)
  if(k == 2)
    out <- sqrt(3) * (2 * t - 1)
  out
}

funE <- function(k, t)
{
  if(k == 1)
    out <- 1 + t - t
  if(k == 2)
    out <- sqrt(5) * (6 * t^2 - 6 * t + 1)
  out
}

# define data generation

c <- c(0.5, 0.3, 1, 0.4, 2)

# generate data

D <- length(gridpoints) # number of grid points
n <- Lvec[3] # number of curves in total

class <- rep(1:class_nr, each = n / class_nr)
group1 <- rep(rep(1:Lvec[1], each = n / (Lvec[1] * class_nr)), class_nr)
group2 <- 1:n
data <- data.frame(class = class, group1 = group1, group2 = group2)

# get eigenfunction evaluations
phis <- list(sapply(1:NPCs[1], FUN = funB1, t = gridpoints),
             sapply(1:NPCs[2], FUN = funB2, t = gridpoints),
             sapply(1:NPCs[3], FUN = funE, t = gridpoints))

# draw basis weights
xis <- vector("list", H)
for(i in 1:H){
  if(NPCs[i] > 0){
    xis[i] <- rmvnorm(Lvec[i], mean = rep(0, NPCs[i]), sigma = diag(NPCs[i]) * nus[[i]])
  }
}

# construct functional random effects
b1 <- xis[[1]] %*% t(phis[[1]])
b2 <- xis[[2]] %*% t(phis[[2]])
e <- xis[[3]] %*% t(phis[[3]])

B1_mat <- B2_mat <- E_mat <- matrix(0, nrow = n, ncol = D)
B1_mat[group1 == 1 & class == 1, ] <- t(replicate(n = n / (Lvec[1] * class_nr),
B1[1, ], simplify = "matrix"))
B1_mat[group1 == 2 & class == 1, ] <- t(replicate(n = n / (Lvec[1] * class_nr),
B1[2, ], simplify = "matrix"))
B2_mat[group1 == 1 & class == 2, ] <- t(replicate(n = n / (Lvec[1] * class_nr),
B2[1, ], simplify = "matrix"))
B2_mat[group1 == 2 & class == 2, ] <- t(replicate(n = n / (Lvec[1] * class_nr),
B2[2, ], simplify = "matrix"))
E_mat <- E

# draw white noise measurement error
eps <- matrix(rnorm(n * D, mean = 0, sd = sqrt(sigmasq)), nrow = n, ncol = D)

# construct curves
Y <- B1_mat + B2_mat + E_mat + eps

# construct Zlist
Zlist <- list()
denseFLMM

Zlist[[1]] <- Zlist[[2]] <- Zlist[[3]] <- list()

d1 <- data.frame(a = as.factor(data$group1[data$class == 1]))
Zlist[[1]][[1]] <- rbind(sparse.model.matrix(~ -1 + a, d1),
   matrix(0, nrow = (1 / class_nr * n), ncol = (Lvec[1])))

d2 <- data.frame(a = as.factor(data$group1[data$class == 2]))
Zlist[[2]][[1]] <- rbind(matrix(0, nrow = (1 / class_nr * n),
   ncol = (Lvec[2])), sparse.model.matrix(~ -1 + a, d2))

d3 <- data.frame(a = as.factor(data$group2))
Zlist[[3]][[1]] <- sparse.model.matrix(~ -1 + a, d3)

# run estimation
results <- denseFLMM(Y = Y, gridpoints = gridpoints, Zlist = Zlist, G = G, Lvec = Lvec,
   groups = NA, Zvars = NA, L = 0.99999, NPC = NA,
   smooth = FALSE)

# plot estimated eigenfunctions
with(results, matplot(gridpoints, phi[[1]], type = "l"))
with(results, matplot(gridpoints, phi[[2]], type = "l"))
with(results, matplot(gridpoints, phi[[3]], type = "l"))
Index

*Topic **FPCA**
  denseFLMM, 2

*Topic **models**,
  denseFLMM, 2

bam, 3

denseFLMM, 2

gam, 3

gamm, 3

mgcv, 3