Package 'gmfamm'

July 22, 2025

Type Package

Title Generalized Multivariate Functional Additive Models

Version 0.1.0

Description Supply implementation to model generalized multivariate functional data using Bayesian additive mixed models of R package 'bamlss' via a latent Gaussian process (see Umlauf, Klein, Zeileis (2018) <doi:10.1080/10618600.2017.1407325>).

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.5), bamlss

Imports mgcv, stats, MASS, splines, Matrix

Suggests testthat (>= 3.0.0), tidyverse, JMbayes2, registr, funData, MFPCA, MJMbamlss, refund

Config/testthat/edition 3

RoxygenNote 7.3.1

NeedsCompilation no

Author Nikolaus Umlauf [aut] (ORCID: <https://orcid.org/0000-0003-2160-9803>), Alexander Volkmann [aut, cre]

Maintainer Alexander Volkmann <alexandervolkmann8@gmail.com>

Repository CRAN

Date/Publication 2024-06-18 14:30:05 UTC

Contents

`apply_respfun_outcome` *Apply link functions based on outcome information*

Description

This is an internal function for the extreme case that a vector is plugged into a response function depending on outcome information.

Usage

```
apply_respfun_outcome(x, outcome, links)
```

Arguments

`x`	Vector of additive predictors.
`outcome`	Factor vector containing information on the outcome of the corresponding element of vector x.
`links`	Vector containing the names of the respective links for the mu outcomes.

Value

Vector of lenght x where different response functions have been applied.

compress_outcomes *Compress the outcome list of predictions into single vectors*

Description

This is an internal function combining all mu and sigma outcomes, respectively, taking into account the outcome information.

Usage

```
compress_outcomes(pred_list, mus, sigmas, outcome)
```

Arguments

pred_list	List of predictions for each outcome.
mus	Character vector with names of included mu models.
sigmas	Character vector with names of included sigma models.
outcome	Factor vector containing the information of which row corresponds to which outcome.

Value

List with two elements containing predictions for mu and sigma model terms. If a some model parameters are missing (such as sigma for binomial distributional assumption) NA elements are contained.

fam *First draft of new family*

Description

Fix three distributional assumptions and do not supply any derivatives.

Usage

```
fam(...)
```

Arguments

...	Not used.

Value

A bamlss family object.

fam2 *Next draft of new family*

Description

Fix three distributional assumptions but supply derivatives.

Usage

```
fam2(...)
```

Arguments

...	Not used.

Value

A bamlss family object.

famg *Draft of new family for gamlss2*

Description

Fix three distributional assumptions but supply derivatives.

Usage

```
famg(...)
```

Arguments

...	Not used.

Value

A gamlss2 family object.

gm *Indicate Generalized Multivariate Model*

Description

This function is used in the formula call of a generalized multivariate functional additive mixed model to supply the information of the outcome and factor variables to bamlss.

Usage

```
gm(y, outcome, ...)
```

Arguments

y	Name of variable in data set which contains the values of the longitudinal outcome.
outcome	Name of variable in data set which is the factor variable indicating which outcome the value is from. Note that only the ordering not the factor levels are used in the estimation process.
...	Additional arguments not used at the moment.

Value

Matrix combining y and outcomes of class 'matrix' and 'gm'.

Examples

```
set.seed(123)
# Number of subjects
n <- 10

# Number of observations
ni <- 3

# Covariate vector
x <- rep(rnorm(n), each = ni)
t <- rep(c(0, 0.5, 1), times = n)

# Additive predictor
eta_1 <- t + 0.5*x
eta_2 <- t + 0.5*x

# Outcomes
y1 <- rnorm(n*ni, eta_1, 0.3)
y2 <- rbinom(n*ni, 1, 1/(1 + exp(-eta_2)))

# Data format
dat <- data.frame(
```

```
   id = factor(rep(seq_len(n), each = ni)),
   y = c(y1, y2),
   dim = factor(rep(c(1, 2), each = n*ni)),
   t = t,
   x = x,
   fpc = 1
)

# Specify formula
f <- list(
  gm(y, dim) ~ t + x,
  sigma1 ~ 1,
  mu2 ~ t + x,
  Lambda ~ -1 + s(id, by = fpc, bs = "re")
)
```

`gmfamm` *Family object for bamlss for Generalized Multivariate Functional Additive Mixed Models*

Description

Family object for bamlss for Generalized Multivariate Functional Additive Mixed Models

Usage

```
gmfamm(family, ...)
```

Arguments

`family`	Vector of bamlss family names to construct the full family.
`...`	Not used at the moment.

Value

An object of class `family.bamlss`

Examples

```
# Short example to see how a family can be specified.
gmfamm(family = c("binomial", "poisson", "gaussian"))

# Long example to see how an analysis can be done.

library(tidyverse)
library(registr)
library(funData)
library(MFPCA)
```

```
library(MJMbamlss)
library(refund)

# Take only three outcomes (normal, binary, poisson)
# Log-transformation of serBilir to get normal distribution
pbc <- pbc_gmfamm %>%
  filter(outcome %in% c("serBilir", "hepatomegaly", "platelets")) %>%
  droplevels() %>%
  mutate(y = case_when(outcome == "serBilir" ~ log(y),
                       outcome != "serBilir" ~ y),
         year = ifelse(year > 9.99, 9.99, year))

pbc_list <- split(pbc, pbc$outcome) %>%
  lapply(function (dat) {
    dat <- dat %>%
      mutate(value = y, index = year) %>%
      select(id, value, index) %>%
      arrange(id, index)
  })

# Fit separate univariate GPFCAs
# Two numbers (x, y) in npc criterion indicate x% total variance but each pc
# hast to contribute at least y%
gfpcs <- mapply(function (data, fams) {
  gfpca_twoStep(Y = data, family = fams, npc_criterion = c(0.99, 0.001),
                verbose = FALSE)
}, data = pbc_list, fams = list("binomial", "poisson", "gaussian"),
SIMPLIFY = FALSE)

# Convert fitted values to funData
mfdata <- multiFunData(lapply(gfpcs, function (x) {
  funData(argvals = x$t_vec,
          X = matrix(x$Yhat$value, ncol = length(x$t_vec), byrow = TRUE))
}))

# Convert estimated eigenfunctions to funData
uniexpansions <- lapply(gfpcs, function (x) {
  list(type = "given",
       functions =  funData(argvals = x$t_vec, X = t(x$efunctions)))
})

# Calculate the maximal number of MFPCs
m <- sum(sapply(gfpcs, "[[", "npc"))

# Estimate the MFPCs with weights 1
mfpca <- MFPCA(mFData = mfdata, M = m, uniExpansions = uniexpansions)

# Choose number of MFPCs based on threshold
nfpc <- min(which(cumsum(mfpca$values) / sum(mfpca$values) > 0.95))

# Attach estimated MFPCs
pbc <- attach_wfpc(mfpca, pbc, n = nfpc, marker = "outcome", obstime = "year")
```

```
# Specify formula
f <- list(
  gm(y, outcome) ~ year + drug + sex, # hepatomegaly
  mu2 ~ year, # platelets
  mu3 ~ year + age, # serBilir
  sigma3 ~ 1, # serBilir sd
  Lambda ~ -1 + s(id, fpc.1, bs = "pcre") +
    s(id, fpc.2, bs = "pcre") + s(id, fpc.3, bs = "pcre") +
    s(id, fpc.4, bs = "pcre")
)

b <- bamlss(f,
            family = gmfamm(c("binomial", "poisson", "gaussian")),
            data = pbc)
```

---

gmfamm_predict *Prediction of Generalized Multivariate Functional Additive Mixed model*

Description

Note: FPC basis has to be evaluated for newdata before the predict function.

Usage

```
gmfamm_predict(
  object,
  newdata,
  model = NULL,
  term = NULL,
  match.names = TRUE,
  intercept = TRUE,
  type = c("link", "parameter"),
  compress = TRUE,
  FUN = function(x) {
     mean(x, na.rm = TRUE)
 },
  trans = NULL,
  what = c("samples", "parameters"),
  nsamps = NULL,
  verbose = FALSE,
  drop = TRUE,
  cores = NULL,
  chunks = 1,
  ...
)
```

Arguments

object	bamlss-model object to be predicted.
newdata	Dataset for which to create predictions. Not needed for conditional survival probabilities.
model	Character or integer, specifies the model for which predictions should be computed.
term	Character or integer, specifies the model terms for which predictions are required. Note that, e.g., term = c("s(x1)", "x2") will compute the combined prediction s(x1) + x2.
match.names	Should partial string matching be used to select the terms for prediction. Note that, e.g., term = "x1" will select all terms including "x1" if match.names = TRUE.
intercept	Should the intercept be included?
type	Character string indicating which type of predictions to compute. link returns the predictors of the corresponding model., "parameter" returns the estimates for all pedictors, "probabilities" returns the survival probabilities conditional on the survival up to the last longitudinal measurement, and "cumhaz" return the cumulative hazard up to the survival time or for a time window after the last longitudinal measurement. If type is set to "loglik", the log-likelihood of the joint model is returned. Note that types "probabilities" and "cumhaz" are not yet implemented.
compress	TRUE if the
FUN	A function that should be applied on the samples of predictors or parameters, depending on argument type.
trans	A transformer function or named list of transformer functions that computes transformed predictions. If trans is a list, the list names must match the names of the parameters of the bamlss.family.
what	Predictions can be computed from samples or estimated parameters of optimizer functions. If no samples are available the default is to use estimated parameters.
nsamps	If the fitted bamlss object contains samples of parameters, computing predictions may take quite some time. Therefore, to get a first feeling it can be useful to compute predictions only based on nsamps samples, i.e., nsamps specifies the number of samples which are extracted on equidistant intervals.
verbose	Print information during runtime of the algorithm.
drop	If predictions for only one model are returned, the list structure is dropped.
cores	Specifies the number of cores that should be used for prediction. Note that this functionality is based on the parallel package.
chunks	Should computations be split into chunks? Prediction is then processed sequentially.
...	Currently not used.

Details

Functionality of some arguments are restricted.

`incorporate_outcome` *Incorporate outcome information into*

Description

This is an internal function multiplying all outcome predictions with 0 if the respective row is not part of the outcome.

Usage

```
incorporate_outcome(pred_list, mus, sigmas, outcome_ids, outcome_levels)
```

Arguments

`pred_list`	List of predictions for each outcome.
`mus`	Integer vector for numbering available mus. Can be NULL but shouldn't.
`sigmas`	Integer vector for numbering available sigmas. Can be NULL.
`outcome_ids`	Numeric matrix resulting from model.matrix call containing the info about the outcomes. Column names are hard coded.
`outcome_levels`	Character string containing the outcome names.

Value

List but now with 0 elements where the rows are not corresponding to outcomes.

`mface_cyc` *Multilevel functional principal components analysis with fast covariance estimation*

Description

Decompose dense or sparse multilevel functional observations using multilevel functional principal component analysis with the fast covariance estimation approach.

Usage

```
mface_cyc(
  Y,
  id,
  visit = NULL,
  twoway = TRUE,
  weight = "obs",
  argvals = NULL,
  pve = 0.99,
```

```
  npc = NULL,
  p = 3,
  m = 2,
  knots = 35,
  silent = TRUE
)
```

Arguments

Y	A multilevel functional dataset on a regular grid stored in a matrix. Each row of the data is the functional observations at one visit for one subject. Missingness is allowed and need to be labeled as NA. The data must be specified.
id	A vector containing the id information to identify the subjects. The data must be specified.
visit	A vector containing information used to identify the visits. If not provided, assume the visit id are 1,2,... for each subject.
twoway	Logical, indicating whether to carry out twoway ANOVA and calculate visit-specific means. Defaults to TRUE.
weight	The way of calculating covariance. weight = "obs" indicates that the sample covariance is weighted by observations. weight = "subj" indicates that the sample covariance is weighted equally by subjects. Defaults to "obs".
argvals	A vector containing observed locations on the functional domain.
pve	Proportion of variance explained. This value is used to choose the number of principal components for both levels.
npc	Pre-specified value for the number of principal components. If given, this overrides pve.
p	The degree of B-splines functions to use. Defaults to 3.
m	The order of difference penalty to use. Defaults to 2.
knots	Number of knots to use or the vectors of knots. Defaults to 35.
silent	Logical, indicating whether to not display the name of each step. Defaults to TRUE.

Details

The fast MFPCA approach (Cui et al., 2023) uses FACE (Xiao et al., 2016) to estimate covariance functions and mixed model equations (MME) to predict scores for each level. As a result, it has lower computational complexity than MFPCA (Di et al., 2009) implemented in the mfpca.sc function, and can be applied to decompose data sets with over 10000 subjects and over 10000 dimensions.

This code is a direct copy of the function mfpca.face in the refund package (version 0.1-35) and slightly adapted to allow cyclical splines in the estimation of the eigenfunctions.

Value

A list containing:

`Yhat`	FPC approximation (projection onto leading components) of `Y`, estimated curves for all subjects and visits
`Yhat.subject`	Estimated subject specific curves for all subjects
`Y.df`	The observed data
`mu`	estimated mean function (or a vector of zeroes if `center==FALSE`).
`eta`	The estimated visit specific shifts from overall mean.
`scores`	A matrix of estimated FPC scores for level1 and level2.
`efunctions`	A matrix of estimated eigenfunctions of the functional covariance, i.e., the FPC basis functions for levels 1 and 2.
`evalues`	Estimated eigenvalues of the covariance operator, i.e., variances of FPC scores for levels 1 and 2.
`pve`	The percent variance explained by the returned number of PCs.
`npc`	Number of FPCs: either the supplied `npc`, or the minimum number of basis functions needed to explain proportion `pve` of the variance in the observed curves for levels 1 and 2.
`sigma2`	Estimated measurement error variance.

Author(s)

Ruonan Li <rli20@ncsu.edu>, Erjia Cui <ecui@umn.edu>, adapted by Alexander Volkmann

References

Cui, E., Li, R., Crainiceanu, C., and Xiao, L. (2023). Fast multilevel functional principal component analysis. *Journal of Computational and Graphical Statistics*, 32(3), 366-377.

Di, C., Crainiceanu, C., Caffo, B., and Punjabi, N. (2009). Multilevel functional principal component analysis. *Annals of Applied Statistics*, 3, 458-488.

Xiao, L., Ruppert, D., Zipunnikov, V., and Crainiceanu, C. (2016). Fast covariance estimation for high-dimensional functional data. *Statistics and Computing*, 26, 409-421.

Examples

```
require(refund)
data(DTI)
mfpca.DTI <- mfpca.face(Y = DTI$cca, id = DTI$ID, twoway = TRUE)
```

pbc_gmfamm *Subset of PBC data set for GMFAMM*

Description

A subset of data from the pbc2 data set which is the Mayo Clinic Primary Biliary Cirrhosis Data, where only patients who survived at least 10 years since they entered the study and were alive and had not had a transplant at the end of the 10th year.

Usage

```
pbc_gmfamm
```

Format

'pbc_gmfamm' A data frame with 5,943 rows and 10 columns:

id patients identifier; in the subset, there are 50 patients included.

years number of years in the study without event

status a factor with levels `alive`, `transplanted`, and `dead`.

drug a factor with levels `placebo` and `D-penicilin`.

age at registration in years.

sex a factor with levels `male` and `female`.

year number of years between enrollment and this visit date.

status2 a numeric vector with value `1` denotign if the patient was dead, and `0` if the patient was alive or transplanted.

outcome a factor with levels `albumin`, `alkaline`, `ascites`, `edema`, `hepatomegaly`, `histologic`, `platelets`, `prothrombin`, `serBilir`, `serChol`, `SGOT`, `spiders`.

y value of the corresponding outcome at the visit date.

Details

Additionally, subject 124 is excluded as it has only one longitudinal measurement per outcome. Function gfpca_twoStep, however, assumes at least two longitudinal observations per subject.

Source

pbc2

References

Hall et al. (2008): Modelling sparse generalized longitudinal observations with latent gaussian processes. Journal of the Royal Statistical Society Series B: Statistical Methodology, 70(4), 703-723.

simMuFu *Simulate multivariate functional data*

Description

This function provides a unified simulation structure for multivariate functional data $f_1, \ldots, f_N$ on one- or two-dimensional domains, based on a truncated multivariate Karhunen-Loeve representation:

$$f_i(t) = \sum_{m=1}^{M} \rho_{i,m} \psi_m(t).$$

The multivariate eigenfunctions (basis functions) ψ_m are constructed from univariate orthonormal bases. There are two different concepts for the construction, that can be chosen by the parameter `type`: A split orthonormal basis (`split`, only one-dimensional domains) and weighted univariate orthonormal bases (`weighted`, one- and two-dimensional domains). The scores $\rho_{i,m}$ in the Karhunen-Loeve representation are simulated independently from a normal distribution with zero mean and decreasing variance. See Details.

Usage

```
simMuFu(
  type,
  argvals,
  M,
  eFunType,
  ignoreDeg = NULL,
  eValType,
  N,
  seed,
  seed_funs = 8
)
```

Arguments

`type`	A character string, specifying the construction method for the multivariate eigenfunctions (either `"split"` or `"weighted"`). See Details.
`argvals`	A list, containing the observation points for each element of the multivariate functional data that is to be simulated. The length of `argvals` determines the number of elements in the resulting simulated multivariate functional data. See Details.
`M`	An integer (`type = "split"`) or a list of integers (`type = "weighted"`), giving the number of univariate basis functions to use. See Details.
`eFunType`	A character string (`type = "split"`) or a list of character strings (`type = "weighted"`), specifying the type of univariate orthonormal basis functions to use. See Details.
`ignoreDeg`	A vector of integers (`type = "split"`) or a list of integer vectors (`type = "weighted"`), specifying the degrees to ignore when generating the univariate orthonormal bases. Defaults to `NULL`. See Details.

eValType	A character string, specifying the type of eigenvalues/variances used for the simulation of the multivariate functions based on the truncated Karhunen-Loeve representation. See eVal for details.
N	An integer, specifying the number of multivariate functions to be generated.
seed	A random seed for the score generation.
seed_funs	A random seed to make the eigenfunction creation reproducible.

Details

The parameter type defines how the eigenfunction basis for the multivariate Karhunen-Loeve representation is constructed:

- type = "split": The basis functions of an underlying 'big' orthonormal basis are split in M parts, translated and possibly reflected. This yields an orthonormal basis of multivariate functions with M elements. This option is implemented only for one-dimensional domains.
- type = "weighted": The multivariate eigenfunction basis consists of weighted univariate orthonormal bases. This yields an orthonormal basis of multivariate functions with M elements. For data on two-dimensional domains (images), the univariate basis is constructed as a tensor product of univariate bases in each direction (x- and y-direction).

Depending on type, the other parameters have to be specified as follows:

Split 'big' orthonormal basis: The parameters M (integer), eFunType (character string) and ignoreDeg (integer vector or NULL) are passed to the function eFun to generate a univariate orthonormal basis on a 'big' interval. Subsequently, the basis functions are split and translated, such that the j-th part of the split function is defined on the interval corresponding to argvals[[j]]. The elements of the multivariate basis functions are given by these split parts of the original basis functions multiplied by a random sign $\sigma_j \in \{-1, 1\}, j = 1, \ldots, p$.

Weighted orthonormal bases: The parameters argvals, M, eFunType and ignoreDeg are all lists of a similar structure. They are passed element-wise to the function eFun to generate orthonormal basis functions for each element of the multivariate functional data to be simulated. In case of bivariate elements (images), the corresponding basis functions are constructed as tensor products of orthonormal basis functions in each direction (x- and y-direction).

If the j-th element of the simulated data should be defined on a one-dimensional domain, then

- argvals[[j]] is a list, containing one vector of observation points.
- M[[j]] is an integer, specifying the number of basis functions to use for this entry.
- eFunType[[j]] is a character string, specifying the type of orthonormal basis functions to use for this entry (see eFun for possible options).
- ignoreDeg[[j]] is a vector of integers, specifying the degrees to ignore when constructing the orthonormal basis functions. The default value is NULL.

If the j-th element of the simulated data should be defined on a two-dimensional domain, then

- argvals[[j]] is a list, containing two vectors of observation points, one for each direction (observation points in x-direction and in y-direction).
- M[[j]] is a vector of two integers, giving the number of basis functions for each direction (x- and y-direction).

- `eFunType[[j]]` is a vector of two character strings, giving the type of orthonormal basis functions for each direction (x- and y-direction, see `eFun` for possible options). The corresponding basis functions are constructed as tensor products of orthonormal basis functions in each direction.
- `ignoreDeg[[j]]` is a list, containing two integer vectors that specify the degrees to ignore when constructing the orthonormal basis functions in each direction. The default value is `NULL`.

The total number of basis functions (i.e. the product of `M[[j]]` for all `j`) must be equal!

This code is a direct copy of the function `simMultiFunData` in the `funData` package (version 1.3-9) and slightly adapted. It also returns the simulated scores and needs the additional argument `seed` to generate reproducible eigenvalues and eigenfunctions.

Value

`simData`	A `multiFunData` object with `N` observations, representing the simulated multivariate functional data.
`trueFuns`	A `multiFunData` object with `M` observations, representing the multivariate eigenfunction basis used for simulating the data.
`trueVals`	A vector of numerics, representing the eigenvalues used for simulating the data.
`score`	A matrix containing the simulated scores.

References

C. Happ, S. Greven (2018): Multivariate Functional Principal Component Analysis for Data Observed on Different (Dimensional) Domains. Journal of the American Statistical Association, 113(522): 649-659.

Examples

```
oldPar <- par(no.readonly = TRUE)
library(funData)

# split
split <- simMuFu(type = "split", argvals = list(seq(0,1,0.01),
                                                seq(-0.5,0.5,0.02)),
                 M = 5, eFunType = "Poly", eValType = "linear", N = 7,
                 seed = 2)

par(mfrow = c(1,2))
plot(split$trueFuns, main = "Split: True Eigenfunctions", ylim = c(-2,2))
plot(split$simData, main = "Split: Simulated Data")

# weighted (one-dimensional domains)

par(oldPar)
```

trafficfam *Draft of family for traffic example*

Description

Fix four distributional assumptions and supply derivatives. Use BCCGo for speed data. Use negative binomial for count data.

Usage

```
trafficfam(...)
```

Arguments

...	Not used.

Value

A bamlss family object.

trafficfam2 *Draft 2 of family for traffic example*

Description

Fix four distributional assumptions and supply derivatives. Use BCCGo for speed data. Use zero-truncated negative binomial for count data (no second derivatives available).

Usage

```
trafficfam2(...)
```

Arguments

...	Not used.

Value

A bamlss family object.

`trafficfam3` *Draft of family for traffic example*

Description

Fix four distributional assumptions and supply derivatives. Use gamma for speed data. Use negative binomial for count data.

Usage

```
trafficfam3(...)
```

Arguments

`...`	Not used.

Value

A bamlss family object.

Examples

```
# Construct data
set.seed(123)
# Number of subjects
n <- 10

# Number of observations
ni <- 3

# Covariate vector
x <- rep(rnorm(n), each = ni)
t <- rep(c(0, 0.5, 1), times = n)

# Additive predictor
eta_1 <- eta_2 <- eta_3 <- eta_4 <- t + 0.5*x

# Outcomes
y1 <- rnbinom(n*ni, exp(eta_1), 0.3)
y2 <- rnbinom(n*ni, exp(eta_2), 0.4)
y3 <- rgamma(n*ni, shape = 0.3, scale = exp(eta_3) / 0.3)
y4 <- rgamma(n*ni, shape = 0.4, scale = exp(eta_4) / 0.4)

# Data format
dat <- data.frame(
   id = factor(rep(seq_len(n), each = ni)),
   y = c(y1, y2, y3, y4),
   dim = factor(rep(c(1:4), each = n*ni)),
   t = t,
   x = x,
```

```
   fpc = 1
)

# Specify formula
f <- list(
  gm(y, dim) ~ t + x,
  sigma1 ~ 1,
  mu2 ~ t + x,
  sigma2 ~ 1,
  mu3 ~ t + x,
  sigma3 ~ 1,
  mu4 ~ t + x,
  sigma4 ~ 1,
  Lambda ~ -1 + s(id, by = fpc, bs = "re")
)

# Model
b <- bamlss(f, family = trafficfam3, n.iter = 20, burnin = 10,
            data = dat)
```

---

trafficfam4 *Draft of family for traffic example*

Description

Fix four distributional assumptions and supply derivatives. Use lognormal for speed data. Use Poisson for count data.

Usage

```
trafficfam4(...)
```

Arguments

...	Not used.

Value

A bamlss family object.

---

varbinq *Generalized Multivariate Functional Additive Models*

Description

This package does things. _PACKAGE

Index