

Package ‘MJMbamlss’

November 27, 2023

Type Package

Title Multivariate Joint Models with 'bamlss'

Version 0.1.0

Description Multivariate joint models of longitudinal and time-to-event data based on functional principal components implemented with 'bamlss'. Implementation for Volkmann, Umlauf, Greven (2023) <[arXiv:2311.06409](https://arxiv.org/abs/2311.06409)>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 3.5), mgcv, bamlss

LinkingTo Rcpp, RcppEigen

Imports stats, funData, statmod, mvtnorm, zoo, coda, gamm4, Matrix, refund, utils, fdapace, sparseFLMM, MFPCA, foreach

Suggests testthat (>= 3.0.0), splines, tidyverse

Config/testthat/edition 3

NeedsCompilation yes

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

Maintainer Alexander Volkmann <alexander.volkmann@hu-berlin.de>

Repository CRAN

Date/Publication 2023-11-27 17:30:05 UTC

R topics documented:

attach_wfpc	2
fpca	3
MFPCA_cov	6
mjm_bamlss	7
MJM_predict	8

pb_subset	10
Predict.matrix.unc_pcre.random.effect	10
preproc_MFPCA	11
simMultiJM	13
sim_bamlss_predict	16
sim_jmbamlss_eval	17
sim_jmbayes_eval	17
sim_jmb_predict	18
smooth.construct.unc_pcre.smooth.spec	18
survint_C	19
varbinq	20

Index 21

attach_wfpc	<i>Attach Weighted Functional Principal Components to the Data</i>
-------------	--

Description

Attach Weighted Functional Principal Components to the Data

Usage

```
attach_wfpc(
  mfpca,
  data,
  n = NULL,
  obstime = "obstime",
  marker = "marker",
  eval_weight = FALSE
)
```

Arguments

mfpca	MFPCA object from which to extract the weighted FPCS.
data	Data set to which the weighted FPCS are to be attached.
n	Number of FPCs to attach. Defaults to NULL which corresponds to all FPCs in mfpca.
obstime	Name of the time variable in data set at which points to evaluate.
marker	Name of the marker variable in the data set which separates the data.
eval_weight	Weight the FPC by the square root of its eigenvalue (then variance comparable throughout all FPCs). Defaults to FALSE.

Value

Data set supplied as argument data with additional columns corresponding to the evaluations of the MFPC basis.

Examples

```
# Small example based on subset of PBC data
data(pbc_subset)

# Estimate MFPC basis and attach to data
mfpc <- preproc_MFPCA(pbc_subset, uni_mean = paste0(
  "logy ~ 1 + sex + drug + s(obstime, k = 5, bs = 'ps') + ",
  "s(age, k = 5, bs = 'ps')"),
  pve_uni = 0.99, nbasis = 5, weights = TRUE, save_uniFPCA = TRUE)
pbc_subset <- attach_wfpc(mfpc, pbc_subset, n = 2)
```

fpca

*Functional principal components analysis by smoothed covariance***Description**

Decomposes functional observations using functional principal components analysis. A mixed model framework is used to estimate scores and obtain variance estimates. This function is a slightly adapted copy of the [fpca.sc](#) function in package `refund` (version 0.1-30).

Usage

```
fpca(
  Y = NULL,
  ydata = NULL,
  Y.pred = NULL,
  argvals = NULL,
  argvals_obs = FALSE,
  argvals_pred = seq(0, 1, by = 0.01),
  random.int = FALSE,
  nbasis = 10,
  nbasis_cov = nbasis,
  bs_cov = "symm",
  pve = 0.99,
  npc = NULL,
  useSymm = FALSE,
  makePD = FALSE,
  center = TRUE,
  cov.est.method = 1,
  integration = "trapezoidal"
)
```

Arguments

`Y`, `ydata` the user must supply either `Y`, a matrix of functions observed on a regular grid, or a data frame `ydata` representing irregularly observed functions. See Details.

<code>Y.pred</code>	if desired, a matrix of functions to be approximated using the FPC decomposition.
<code>argvals</code>	the argument values of the function evaluations in <code>Y</code> , defaults to a equidistant grid from 0 to 1.
<code>argvals_obs</code>	Should the timepoints of the original observations be used to evaluate the FPCs. Defaults to FALSE.
<code>argvals_pred</code>	Vector of timepoints on which to evaluate the FPCs. Defaults to a sequence from 0 to 1.
<code>random.int</code>	If TRUE, the mean is estimated by <code>gamm4</code> with random intercepts. If FALSE (the default), the mean is estimated by <code>gam</code> treating all the data as independent.
<code>nbasis</code>	number of B-spline basis functions used for estimation of the mean function.
<code>nbasis_cov</code>	number of basis functions used for the bivariate smoothing of the covariance surface.
<code>bs_cov</code>	type of spline for the bivariate smoothing of the covariance surface. Default is symmetric fast covariance smoothing proposed by Cederbaum.
<code>pve</code>	proportion of variance explained: used to choose the number of principal components.
<code>npc</code>	prespecified value for the number of principal components (if given, this overrides <code>pve</code>).
<code>useSymm</code>	logical, indicating whether to smooth only the upper triangular part of the naive covariance (when <code>cov.est.method==2</code>). This can save computation time for large data sets, and allows for covariance surfaces that are very peaked on the diagonal.
<code>makePD</code>	logical: should positive definiteness be enforced for the covariance surface estimate?
<code>center</code>	logical: should an estimated mean function be subtracted from <code>Y</code> ? Set to FALSE if you have already demeaned the data using your favorite mean function estimate.
<code>cov.est.method</code>	covariance estimation method. If set to 1 (the default), a one-step method that applies a bivariate smooth to the $y(s_1)y(s_2)$ values. This can be very slow. If set to 2, a two-step method that obtains a naive covariance estimate which is then smoothed.
<code>integration</code>	quadrature method for numerical integration; only 'trapezoidal' is currently supported.

Details

This function computes a FPC decomposition for a set of observed curves, which may be sparsely observed and/or measured with error. A mixed model framework is used to estimate curve-specific scores and variances.

FPCA via kernel smoothing of the covariance function, with the diagonal treated separately, was proposed in Staniswalis and Lee (1998) and much extended by Yao et al. (2005), who introduced the 'PACE' method. `fpca.sc` uses penalized splines to smooth the covariance function, as developed by Di et al. (2009) and Goldsmith et al. (2013). This implementation uses REML and Cederbaum et al. (2018) for smoothing the covariance function.

The functional data must be supplied as either

- an $n \times d$ matrix Y , each row of which is one functional observation, with missing values allowed; or
- a data frame `ydata`, with columns `' .id '` (which curve the point belongs to, say i), `' .index '` (function argument such as time point t), and `' .value '` (observed function value $Y_i(t)$).

Value

An object of class `fpca` containing:

<code>Yhat</code>	FPC approximation (projection onto leading components) of <code>Y.pred</code> if specified, or else of <code>Y</code> .
<code>Y</code>	the observed data
<code>scores</code>	$n \times npc$ matrix of estimated FPC scores.
<code>mu</code>	estimated mean function (or a vector of zeroes if <code>center==FALSE</code>).
<code>efunctions</code>	$d \times npc$ matrix of estimated eigenfunctions of the functional covariance, i.e., the FPC basis functions.
<code>evalues</code>	estimated eigenvalues of the covariance operator, i.e., variances of FPC scores.
<code>npc</code>	number of FPCs: either the supplied <code>npc</code> , or the minimum number of basis functions needed to explain proportion <code>pve</code> of the variance in the observed curves.
<code>argvals</code>	argument values of eigenfunction evaluations
<code>sigma2</code>	estimated measurement error variance.
<code>diag.var</code>	diagonal elements of the covariance matrices for each estimated curve.
<code>VarMats</code>	a list containing the estimated covariance matrices for each curve in <code>Y</code> .
<code>crit.val</code>	estimated critical values for constructing simultaneous confidence intervals.

Author(s)

Jeff Goldsmith <jeff.goldsmith@columbia.edu>, Sonja Greven <sonja.greven@stat.uni-muenchen.de>, Lan Huo <Lan.Huo@nyumc.org>, Lei Huang <huangracer@gmail.com>, and Philip Reiss <phil.reiss@nyumc.org>, Alexander Volkman

References

- Cederbaum, J. Scheipl, F. and Greven, S. (2018). Fast symmetric additive covariance smoothing. *Computational Statistics & Data Analysis*, 120, 25–41.
- Di, C., Crainiceanu, C., Caffo, B., and Punjabi, N. (2009). Multilevel functional principal component analysis. *Annals of Applied Statistics*, 3, 458–488.
- Goldsmith, J., Greven, S., and Crainiceanu, C. (2013). Corrected confidence bands for functional data using principal components. *Biometrics*, 69(1), 41–51.
- Staniswalis, J. G., and Lee, J. J. (1998). Nonparametric regression analysis of longitudinal data. *Journal of the American Statistical Association*, 93, 1403–1418.
- Yao, F., Mueller, H.-G., and Wang, J.-L. (2005). Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 100, 577–590.

MFPCA_cov	<i>Function to calculate the multivariate FPCA for a given covariance matrix and univariate basis functions</i>
-----------	---

Description

Function to calculate the multivariate FPCA for a given covariance matrix and univariate basis functions

Usage

```
MFPCA_cov(cov, basis_funs, scores = NULL, weights = NULL)
```

Arguments

cov	Covariance matrix of the basis functions coefficients.
basis_funs	List with basis functions on each dimension. The basis functions are funData objects
scores	Matrix (n rows, B columns) containing the basis functions coefficients. Defaults to NULL which does not calculate the multivariate scores.
weights	Vector of weights, defaults to 1 for each element

Value

List mimicking an MFPCAfit object containing the following components:

values A vector of eigenvalues.

functions A multiFunData object containing the multivariate functional principal components.

scores A matrix containing the scores (if applicable).

vectors A matrix representing the eigenvectors associated with the combined univariate score vectors.

normFactors The normalizing factors used for calculating the multivariate eigenfunctions and scores.

Examples

```
library(funData)
# Covariance matrix for the data generation in simulation scenario I
auto <- matrix(c(0.08, -0.07, -0.07, 0.9), ncol = 2)
cross <- matrix(rep(0.03, 4), ncol = 2)
cor <- matrix(c(0, 1, 0.75, 0.5, 0, 0,
               1, 0, 1, 0.75, 0.5, 0,
               0.75, 1, 0, 1, 0.75, 0.5,
               0.5, 0.75, 1, 0, 1, 0.75,
               0, 0.5, 0.75, 1, 0, 1,
               0, 0, 0.5, 0.75, 1, 0),
```

```
ncol = 6)
cov <- kronecker(cor, cross) + kronecker(diag(c(1, 1.2, 1.4, 1.6, 1.8, 2)),
                                         auto)

# Basis functions on each dimension
seq1 <- seq(0, 1, by = 0.01)
b_funs <- rep(list(funData(argvals = seq1,
                          X = matrix(c(rep(1, length(seq1))), seq1),
                          byrow = TRUE, ncol = length(seq1))), 6)

# Prepare objects for the model on different data sets
mfpca_tru <- MFPCA_cov(cov = cov, basis_funs = b_funs)
```

mjm_bamlss

Family for Flexible Multivariate Joint Model

Description

This function specifies the different predictors and link functions as well as the corresponding transform/updating/sampling functions as well as the predict function.

Usage

```
mjm_bamlss(...)
```

Arguments

... All arguments are actually hard coded as needed by the implementation.

Details

Family object to fit a flexible additive joint model for multivariate longitudinal and survival data under a Bayesian approach using multivariate functional principal components as presented in Volkman, Umlauf, Greven (2023).

Value

An object of class `family.bamlss`.

References

Volkman, A., Umlauf, N., Greven, S. (2023). Flexible joint models for multivariate longitudinal and time-to-event data using multivariate functional principal components. <arXiv:2311.06409>

Examples

```

library(mgcv)
library(bamlss)
data(pbc_subset)
mfPCA <- preproc_MFPCA(pbc_subset, uni_mean = paste0(
  "logy ~ 1 + sex + drug + s(obstime, k = 5, bs = 'ps') + ",
  "s(age, k = 5, bs = 'ps')"),
  pve_uni = 0.99, nbasis = 5, weights = TRUE, save_uniFPCA = TRUE)
pbc_subset <- attach_wfpc(mfPCA, pbc_subset, n = 2)
mfPCA_list <- list(
  list(functions = funData::extractObs(mfPCA$functions, 1),
    values = mfPCA$values[1]),
  list(functions = funData::extractObs(mfPCA$functions, 2),
    values = mfPCA$values[2]))

# Model formula
f <- list(
  Surv2(survtime, event, obs = logy) ~ -1 +
    s(survtime, k = 5, bs = "ps", xt = list("scale" = FALSE)),
  gamma ~ 1 + sex + drug + s(age, k = 5, bs = 'ps'),
  mu ~ -1 + marker + sex:marker + drug:marker +
    s(obstime, by = marker, xt = list("scale" = FALSE), k = 5, bs = "ps") +
    s(age, by = marker, xt = list("scale" = FALSE), k = 5, bs = "ps") +
    s(id, fpc.1, bs = "unc_pcre",
      xt = list("mfpc" = mfPCA_list[[1]], scale = "FALSE")) +
    s(id, fpc.2, bs = "unc_pcre",
      xt = list("mfpc" = mfPCA_list[[2]], scale = "FALSE")),
  sigma ~ -1 + marker,
  alpha ~ -1 + marker
)

# Model fit
b <- bamlss(f, family = mjm_bamlss, data = pbc_subset,
  timevar = "obstime", maxit = 15, n.iter = 15, burnin = 2,
  thin = 2)

```

MJM_predict

*Prediction of MJM model***Description**

Note: Writing a predict function is a bit tricky. For longitudinal prediction, if subject specific predictions are wanted, then the PCRE terms must be attached to newdata and already evaluated. If the model uses standardized survival matrices, the different linear predictors should be predicted using different data sets.

Usage

```
MJM_predict(
```



```

    object,
    newdata,
    type = c("link", "parameter", "probabilities", "cumhaz"),
    dt,
    id,
    FUN = function(x) {
      mean(x, na.rm = TRUE)
    },
    subdivisions = 7,
    cores = NULL,
    chunks = 1,
    verbose = FALSE,
    ...
  )

```

Arguments

object	bamlss-model object to be predicted.
newdata	Dataset for which to create predictions. Not needed for conditional survival probabilities.
type	Character string indicating which type of predictions to compute. <code>link</code> returns estimates for all predictors with the respective link functions applied, <code>parameter</code> returns the estimates for all predictors, <code>probabilities</code> returns the survival probabilities conditional on the survival up to the last longitudinal measurement, and <code>cumhaz</code> return the cumulative hazard up to the survival time or for a time window after the last longitudinal measurement. If <code>type</code> is set to <code>loglik</code> , the log-likelihood of the joint model is returned. Note that types <code>probabilities</code> and <code>cumhaz</code> are not yet implemented.
dt	The time window after the last observed measurement for which predictions should be computed.
id	Integer or character, that specifies the individual for which the plot should be created.
FUN	A function that should be applied on the samples of predictors or parameters, depending on argument <code>type</code> .
subdivisions	Number of Gaussian quadrature points for survival integral calculation.
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.
verbose	Print information during runtime of the algorithm.
...	Currently not used.

pbcs_subset *PBC Subset*

Description

A subset of the pbc data provided by package JMbayes2 used only to illustrate the functions.

Usage

```
pbcs_subset
```

Format

```
## 'pbcs_subset' A data frame with 336 observations and 10 columns:
```

id Subject id.

survtime Survival time of composite endpoint.

event Composite endpoint.

sex Male or female.

drug Placebo or D-penicil.

age Age.

marker Name of longitudinal biomarker (albumin, SerBilir, serChol, SGOT)

obstime Longitudinal time.

y Longitudinal outcome value.

logy Log-transformed longitudinal outcome value.

Source

<<https://cran.r-project.org/web/packages/JMbayes2/index.html>>

```
Predict.matrix.unc_pcre.random.effect
      mgcv-style constructor for prediction of PC-basis functional random
      effects
```

Description

This predictor function uses time-information saved in the object. This is handled within the bamlss-transform function, so this function is not exported.

Usage

```
## S3 method for class 'unc_pcre.random.effect'
Predict.matrix(object, data)
```

Arguments

object a smooth specification object, see [smooth.construct](#)
 data see [smooth.construct](#)

Value

design matrix for PC-based functional random effects

Author(s)

Alexander Volkman, adapted from 'Predict.matrix.pcre.random.effect' by F. Scheipl (adapted from 'Predict.matrix.random.effect' by S.N. Wood).

Examples

```
data(pbc_subset)
mf pca <- preproc_MFPCA(pbc_subset, uni_mean = paste0(
  "logy ~ 1 + sex + drug + s(obstime, k = 5, bs = 'ps') + ",
  "s(age, k = 5, bs = 'ps')"),
  pve_uni = 0.99, nbasis = 5, weights = TRUE, save_uniFPCA = TRUE)
pbc_subset <- attach_wfpc(mf pca, pbc_subset, n = 2)
mf pca_list <- list(
  list(functions = funData::extractObs(mf pca$functions, 1),
    values = mf pca$values[1]),
  list(functions = funData::extractObs(mf pca$functions, 2),
    values = mf pca$values[2]))
sm <- smoothCon(s(id, fpc.1, bs = "unc_pcre",
  xt = list("mfpc" = mf pca_list[[1]], scale = "FALSE")), pbc_subset)[[1]]
sm$timevar <- "obstime"
sm$term <- c(sm$term, "obstime")
pm <- PredictMat(sm, pbc_subset, n = 4*nrow(pbc_subset))
```

```
preproc_MFPCA
```

Preprocessing step to create MFPCA object

Description

This function takes the data and uses the residuals of marker-specific additive models to estimate the covariance structure for a MFPCA

Usage

```
preproc_MFPCA(
  data,
  uni_mean = "y ~ s(obstime) + s(x2)",
  time = "obstime",
  id = "id",
  marker = "marker",
```

```

M = NULL,
weights = FALSE,
remove_obs = NULL,
method = c("fpca", "fpca.sc", "FPCA", "PACE"),
nbasis = 10,
nbasis_cov = nbasis,
bs_cov = "symm",
npc = NULL,
fve_uni = 0.99,
pve_uni = 0.99,
fit = FALSE,
max_time = NULL,
save_uniFPCA = FALSE,
save_uniGAM = FALSE
)

```

Arguments

<code>data</code>	Data.frame such as returned by function <code>simMultiJM</code> .
<code>uni_mean</code>	String to crate a formula for the univariate addtive models.
<code>time</code>	String giving the name of the longitudinal time variable.
<code>id</code>	String giving the name of the identifier.
<code>marker</code>	String giving the name of the longitudinal marker variable.
<code>M</code>	Number of mFPCs to compute in the MFPCA. If not supplied, it defaults to the maximum number of computable mFPCs.
<code>weights</code>	TRUE if inverse sum of univariate eigenvals should be used as weights in the scalar product of the MFPCA. Defaults to FALSE (weights 1).
<code>remove_obs</code>	Minimal number of observations per individual and marker to be included in the FPC estimation. Defaults to NULL (all observations). Not removing observations can lead to problems if the univariate variance estimate is negative and has to be truncated, then the scores for IDs with few observations cannot be estimated.
<code>method</code>	Which package to use for the univariate FPCA. Either function adapted function 'fpca', 'FPCA' from package <code>fdapace</code> , 'fpca.sc' from package <code>refund</code> , or function 'PACE' from package <code>MFPCA</code> .
<code>nbasis</code>	Number of B-spline basis functions for mean estimate for methods <code>fpca</code> , <code>fpca.sc</code> , <code>PACE</code> . For <code>fpca.sc</code> , <code>PACE</code> also bivariate smoothing of covariance estimate.
<code>nbasis_cov</code>	Number of basis functions used for the bivariate smoothing of the covariance surface for method <code>fpca</code> .
<code>bs_cov</code>	Type of spline for the bivariate smoothing of the covariance surface for method <code>fpca</code> . Default is symmetric fast covariance smoothing proposed by Cederbaum.
<code>npc</code>	Number of univariate principal components to use in <code>fpca.sc</code> , <code>PACE</code> .
<code>fve_uni</code>	Fraction of univariate variance explained for method <code>FPCA</code> .
<code>pve_uni</code>	Proportion of univariate variance explained for methods <code>fpca</code> , <code>fpca.sc</code> , <code>PACE</code> .

fit	MFPCA argument to return a truncated KL fit to the data. Defaults to FALSE.
max_time	If supplied, forces the evaluation of the MFPCs up to maxtime. Only implemented for method = 'fpca'.
save_uniFPCA	TRUE to attach list of univariate FPCAs as attribute to output. Defaults to FALSE.
save_uniGAM	TRUE to attach list of univariate additive models used to calculate the residuals. Defaults to FALSE.

Value

An object of class MFPCAfit with additional attributes depending on the arguments save_uniFPCA, save_uniGAM, fit.

Examples

```
data(pbc_subset)
mf pca <- preproc_MFPCA(pbc_subset, uni_mean = paste0(
  "logy ~ 1 + sex + drug + s(obstime, k = 10, bs = 'ps') + ",
  "s(age, k = 10, bs = 'ps')"),
  pve_uni = 0.99, nbasis = 5, weights = TRUE, save_uniFPCA = TRUE)
```

 simMultiJM

New Simulation Function For Multivariate JMs Based On FPCs

Description

Adapt the structure given by simJM function in bamlss.

Usage

```
simMultiJM(
  nsub = 300,
  times = seq(0, 120, 1),
  probmiss = 0.75,
  max_obs = length(times),
  maxfac = 1.5,
  nmark = 2,
  long_assoc = c("FPC", "splines", "param"),
  M = 6,
  FPC_bases = NULL,
  FPC_evals = NULL,
  mfpc_args = list(type = "split", eFunType = "Poly", ignoreDeg = NULL, eValType =
    "linear", eValScale = 1),
  re_cov_mat = NULL,
  ncovar = 2,
  lambda = function(t, x) {
    1.4 * log((t + 10)/1000)
```

```

},
  gamma = function(x) {
    -1.5 + 0.3 * x[, 1]
  },
  alpha = rep(list(function(t, x) {
    0.3 + 0 * t
  })), nmark),
  mu = rep(list(function(t, x) {
    1.25 + 0.6 * sin(x[, 2]) + (-0.01) * t
  })), nmark),
  sigma = function(t, x) {
    0.3 + 0 * t + I(x$marker == "m2") * 0.2
  },
  tmax = NULL,
  seed = NULL,
  full = FALSE,
  file = NULL
)

```

Arguments

nsub	Number of subjects.
times	Vector of time points.
probmiss	Probability of missingness.
max_obs	Maximal number of observations per individual and marker. Defaults to no upper limit.
maxfac	Factor changing the uniform censoring interval.
nmark	Number of markers.
long_assoc	Longitudinal association between the markers (Defaults to "FPC"). If "splines" or "param", then specify the normal covariance matrix with argument 're_cov_mat' and include the random effects in argument mu. If "FPC", then principal components are used to model the association structure.
M	Number of principal components.
FPC_bases	FunData object. If supplied, use the contained FPC as basis for the association structure.
FPC_evals	Vector of eigenvalues. If supplied, use the provided eigenvalues for the association structure.
mfpc_args	List containing the named arguments "type", "eFunType", "ignoreDeg", "eValType" of function simMultiFunData and "eValScale" for scaling the eigenvalues.
re_cov_mat	If supplied, a covariance matrix to use for drawing the random effects needed for the association structure.
ncovar	Number of covariates.
lambda	Additive predictor of time-varying survival covariates.
gamma	Additive predictor of time-constant survival covariates.

alpha	List of length nmark containing the additive predictors of the association.
mu	List of length nmark containing the additive predictors of the longitudinal part.
sigma	Additive predictor of the variance.
tmax	Maximal time point of observations.
seed	Seed for reproducibility.
full	Create a wide-format data.frame and a short one containing only survival info.
file	Name of the data file the generated data set should be stored into (e.g., "sim-data.RData") or NULL if the dataset should directly be returned in R.

Value

For full = TRUE a list of four data.frames is returned:

data Simulated dataset in long format including all longitudinal and survival covariates.

data_full Simulated dataset on a grid of fixed time points.

data_hypo Simulated dataset on a grid of fixed time points with hypothetical longitudinal outcomes after the event.

fpc_base If applicable, include the FPC basis used for simulation.

data_short Convenience output containing only one observation per subject for easy access to event-times.

For full = FALSE only the first dataset is returned.

Examples

```
# Number of individuals
n <- 15
# Covariance matrix for the data generation
auto <- matrix(c(0.08, -0.07, -0.07, 0.9), ncol = 2)
cross <- matrix(rep(0.03, 4), ncol = 2)
cor <- matrix(c(0, 1, 0.75, 0.5, 0, 0,
               1, 0, 1, 0.75, 0.5, 0,
               0.75, 1, 0, 1, 0.75, 0.5,
               0.5, 0.75, 1, 0, 1, 0.75,
               0, 0.5, 0.75, 1, 0, 1,
               0, 0, 0.5, 0.75, 1, 0),
              ncol = 6)
cov <- kronecker(cor, cross) +
      kronecker(diag(c(1, 1.2, 1.4, 1.6, 1.8, 2)), auto)

# Simulate the data
d_rirs <- simMultiJM(
  nsub = n, times = seq(0, 1, by = 0.01), max_obs = 15, probmiss = 0.75,
  maxfac = 1.75, nmark = 6, long_assoc = "param", M = NULL, FPC_bases = NULL,
  FPC_evals = NULL, mfpc_args = NULL, re_cov_mat = cov, ncovar = 2,
  lambda = function(t, x) {1.37 * t^(0.37)},
  gamma = function(x) {-1.5 + 0.48*x[, 3]},
  alpha = list(function(t, x) {1.5 + 0*t}, function(t, x) {0.6 + 0*t},
```

```

      function(t, x) {0.3 + 0*t}, function(t, x) {-0.3 + 0*t},
      function(t, x) {-0.6 + 0*t}, function(t, x) {-1.5 + 0*t}),
mu = list(function(t, x, r){
  0 + 0.2*t - 0.25*x[, 3] - 0.05*t*x[, 3] + r[, 1] + r[, 2]*t
}, function(t, x, r){
  0 + 0.2*t - 0.25*x[, 3] - 0.05*t*x[, 3] + r[, 3] + r[, 4]*t
}, function(t, x, r){
  0 + 0.2*t - 0.25*x[, 3] - 0.05*t*x[, 3] + r[, 5] + r[, 6]*t
}, function(t, x, r){
  0 + 0.2*t - 0.25*x[, 3] - 0.05*t*x[, 3] + r[, 7] + r[, 8]*t
}, function(t, x, r){
  0 + 0.2*t - 0.25*x[, 3] - 0.05*t*x[, 3] + r[, 9] + r[, 10]*t
}, function(t, x, r){
  0 + 0.2*t - 0.25*x[, 3] - 0.05*t*x[, 3] + r[, 11] + r[, 12]*t
}),
sigma = function(t, x) {log(0.06) + 0*t}, tmax = NULL, seed = NULL,
full = TRUE, file = NULL)

```

sim_bamlss_predict

Simulation Helper Function - Predict the Results for bamlss-Models

Description

This function takes all the models listed in a folder and predicts the fit.

Usage

```
sim_bamlss_predict(m, wd, model_wd, data_wd, rds = TRUE, old = FALSE)
```

Arguments

m	Vector containing all the file names of the models.
wd	Path to simulations folder.
model_wd	Simulation setting folder where the models are saved.
data_wd	Simulation data folder.
rds	Objects are saved as .rds files (for backwards compatibility when .Rdata files were used). Defaults to TRUE.
old	Simulated data sets before Version 0.0.3 (samples need to be adapted for standardized survival matrices). Defaults to FALSE.

sim_jmbamlss_eval	<i>Simulation Helper Function - Evaluate the Simulation for JMbamlss Setting</i>
-------------------	--

Description

This function evaluates the results for a given folder of JMbamlss model fits.

Usage

```
sim_jmbamlss_eval(wd, model_wd, data_wd, name, rds = TRUE)
```

Arguments

wd	Path to simulations folder.
model_wd	Simulation setting folder where the models are saved.
data_wd	Simulation data folder.
name	Name for description of the simulation setting.
rds	Objects are saved as .rds files (for backwards compatibility when .Rdata files were used). Defaults to TRUE.

sim_jmbayes_eval	<i>Simulation Helper Function - Evaluate the Simulation for JMbayes Setting</i>
------------------	---

Description

This function evaluates the results for a given folder of JMbayes model fits.

Usage

```
sim_jmbayes_eval(wd, model_wd, data_wd, name, rds = TRUE)
```

Arguments

wd	Path to simulations folder.
model_wd	Simulation setting folder where the models are saved.
data_wd	Simulation data folder.
name	Name for description of the simulation setting.
rds	Objects are saved as .rds files (for backwards compatibility when .Rdata files were used). Defaults to TRUE.

sim_jmb_predict *Simulation Helper Function - Predict the Results for JMbayes-Models*

Description

This function takes all the models listed in a folder and predicts the fit.

Usage

```
sim_jmb_predict(m, wd, model_wd, data_wd, rds = TRUE, gamma_timeconst = TRUE)
```

Arguments

m	Vector containing all the file names of the models.
wd	Path to simulations folder.
model_wd	Simulation setting folder where the models are saved.
data_wd	Simulation data folder.
rds	Objects are saved as .rds files (for backwards compatibility when .Rdata files were used). Defaults to TRUE.
gamma_timeconst	Only implemented for timeconstant gamma predictors. If FALSE a warning message is returned.

smooth.construct.unc_pcre.smooth.spec
mgcv-style constructor for PC-basis functional random effects (no constraint)

Description

Sets up design matrix for functional random effects based on the PC scores of the covariance operator of the random effect process. Note that there is no constraint on the smoother. See [smooth.construct.re.smooth.spec](#) for more details on mgcv-style smoother specification and [smooth.construct.pcre.smooth.spec](#) for the corresponding refund implementation.

Usage

```
## S3 method for class 'unc_pcre.smooth.spec'  
smooth.construct(object, data, knots, ...)
```

Arguments

object	a smooth specification object, see smooth.construct
data	see smooth.construct .
knots	see smooth.construct .
...	see smooth.construct .

Details

This is an internal function as the corresponding smooth object and its predict method is primarily used within the `bamlss` call.

Value

An object of class "random.effect". See [smooth.construct](#) for the elements that this object will contain.

Author(s)

Alexander Volkman; adapted from 'pcre' constructor by F. Scheipl (adapted from 're' constructor by S.N. Wood).

Examples

```
data(pbc_subset)
mfPCA <- preproc_MFPCA(pbc_subset, uni_mean = paste0(
  "log ~ 1 + sex + drug + s(obstime, k = 5, bs = 'ps') + ",
  "s(age, k = 5, bs = 'ps')"),
  pve_uni = 0.99, nbasis = 5, weights = TRUE, save_uniFPCA = TRUE)
pbc_subset <- attach_wfpc(mfPCA, pbc_subset, n = 2)
mfPCA_list <- list(
  list(functions = funData::extractObs(mfPCA$functions, 1),
    values = mfPCA$values[1]),
  list(functions = funData::extractObs(mfPCA$functions, 2),
    values = mfPCA$values[2]))
sm <- smoothCon(s(id, fpc.1, bs = "unc_pcre",
  xt = list("mfpc" = mfPCA_list[[1]]), scale = "FALSE"), pbc_subset)
```

survint_C

Survival Integral

Description

This function is a wrapper function for calculating the survival integral in C needed in the calculation of the score vector and Hessian.

Usage

```
survint_C(
  pred = c("lambda", "gamma", "long", "fpc_re"),
  pre_fac,
  pre_vec = NULL,
  omega,
  int_fac = NULL,
  int_vec = NULL,
  weights,
  survtime
)
```

Arguments

pred	String to define for which predictor the survival integral is calculated.
pre_fac	Vector serving as factor before the survival integral. Corresponds to the gamma predictor.
pre_vec	Matrix serving as row vectors before the survival integral. Only needed if pred = "gamma".
omega	Vector serving as additive predictor placeholder within the survival integral. Present for all pred.
int_fac	Vector serving as factor within the survival integral. Only needed for the longitudinal predictors.
int_vec	Matrix serving as row vectors within the survival integral. NULL only if pred = "gamma".
weights	Vector containing the Gaussian integration weights.
survtime	Vector containing the survival times for weighting of the integral.

Details

The survival integral has a similar structure for the different model predictors. It is always a sum over all individuals, followed by the multiplication with a pre-integral factor (pre_fac). For the gamma predictor a pre-integral vector is next. Then, the integral itself consists of a weighted sum (weights) of gauss-quadrature integration points weighted by the survival time of the individuals (survtime). Inside the integral, the current additive predictor (omega) is multiplied with an in-integral vector (int_vec), except for predictor gamma. All longitudinal predictors additionally include an in-integration factor (int_fac).

The difference between predictors "long" and "fpc_re" is that the latter makes efficient use of the block structure of the design matrix for unconstrained functional principal component random effects. The outputs also differ as the Hessian for "fpc_re" is a diagonal matrix, so only the diagonal elements are returned.

varbinq

Flexible Joint Models for Multivariate Longitudinal and Time-to-Event Data

Description

This package contains all functions and implementations of the corresponding paper by Volkman, Umlauf, Greven: "Flexible joint models for multivariate longitudinal and time-to-event data using multivariate functional principal components". Code to reproduce the simulation and analysis as well as additional information on the model fitting process are contained in the "inst" folder.

Index

* datasets

- pbc_subset, 10
- attach_wfpc, 2
- fPCA, 3
- fPCA.sc, 3
- gam, 4
- gamm4, 4
- MFPCA_cov, 6
- mjm_bamlss, 7
- MJM_predict, 8
- parallel, 9
- pbc_subset, 10
- Predict.matrix.unc_pcre.random.effect,
 10
- preproc_MFPCA, 11
- sim_bamlss_predict, 16
- sim_jmb_predict, 18
- sim_jmbamlss_eval, 17
- sim_jmbayes_eval, 17
- simMultiJM, 13
- smooth.construct, 11, 18, 19
- smooth.construct.pcre.smooth.spec, 18
- smooth.construct.re.smooth.spec, 18
- smooth.construct.unc_pcre.smooth.spec,
 18
- survint_C, 19
- varbinq, 20