Package ‘bliss’

July 17, 2024

Title  Bayesian Functional Linear Regression with Sparse Step Functions
Version  1.1.1
Author  Paul-Marie Grollemund [aut, cre],
        Isabelle Sanchez [ctr],
        Meili Baragatti [ctr]
Maintainer  Paul-Marie Grollemund <paul_marie.grollemund@uca.fr>
Description  A method for the Bayesian functional linear regression model (scalar-on-function),
             including two estimators of the coefficient function and an estimator of its support.
             A representation of the posterior distribution is also available. Grollemund P-M., Abraham C.,
License  GPL-3
Depends  R (>= 3.5.0)
LinkingTo  Rcpp, RcppArmadillo, RcppProgress
Encoding  UTF-8
LazyData  TRUE
URL  https://github.com/pmgrollemund/bliss
BugReports  https://github.com/pmgrollemund/bliss/issues
Imports  Rcpp, MASS, ggplot2, RcppArmadillo
Suggests  rmarkdown, knitr, RColorBrewer
RoxygenNote  7.3.1
VignetteBuilder  knitr
NeedsCompilation  yes
Repository  CRAN
Date/Publication  2024-07-17 12:00:02 UTC

Contents

BIC_model_choice ............................................. 2
bliss .......................................................... 3
BIC_model_choice

Description

Model selection with BIC criterion.

Usage

BIC_model_choice(Ks, iter, data, verbose = T)
Arguments

Ks a numerical vector containing the K values.
iter an integer, the number of iteration for each run of fit_Bliss.
data a list containing required options to run the function fit_Bliss.
verbose write stuff if TRUE (optional).

Value

A numerical vector, the BIC values for the Bliss model for different K value.

Examples

```r
param_sim <- list(Q=1,n=100,p=c(50),grids_lim=list(c(0,1)))
data <- sim(param_sim,verbose=TRUE)
iter = 1e2
Ks <- 1:5

res_BIC <- BIC_model_choice(Ks,iter,data)
plot(res_BIC,xlab="K",ylab="BIC")
```

bliss

### bliss: Bayesian functional Linear regression with Sparse Step functions

Description

A method for the Bayesian Functional Linear Regression model (functions-on-scalar), including two estimators of the coefficient function and an estimator of its support. A representation of the posterior distribution is also available.

Author(s)

Maintainer: Paul-Marie Grollemund <paul_marie.grollemund@uca.fr>

Other contributors:

- Isabelle Sanchez <isabelle.sanchez@inra.fr> [contractor]
- Meili Baragatti <meili.baragatti@supagro.fr> [contractor]

See Also

Useful links:

- [https://github.com/pmgrollemund/bliss](https://github.com/pmgrollemund/bliss)
- Report bugs at [https://github.com/pmgrollemund/bliss/issues](https://github.com/pmgrollemund/bliss/issues)
Bliss_Gibbs_Sampler

Description
A Gibbs Sampler algorithm to sample the posterior distribution of the Bliss model.

Usage
Bliss_Gibbs_Sampler(data, param, verbose = FALSE)

Arguments
data a list containing:
   y a numerical vector, the outcome values y_i.
x a list of matrices, the qth matrix contains the observations of the qth functional
covariate at time points given by grids.
grids a list of numerical vectors, the qth vector is the grid of time points for the
qth functional covariate.

param a list containing:
   Q an integer, the number of functional covariates.
   iter an integer, the number of iterations of the Gibbs sampler algorithm.
   K a vector of integers, corresponding to the numbers of intervals for each co-
   variate.
   p an integer, the number of time points.
   basis a character (optional). The possible values are "uniform" (default), "epanech-
   nikov", "gauss" and "triangular" which correspond to different basis func-
   tions to expand the coefficient function and the functional covariates
   phi_1 a numerical (optional). An hyperparameters related to the exponential
   prior on the length of the intervals. Lower values promotes wider intervals.
   verbose_cpp a boolean value (optional). Write stuff from the Rcpp scripts if
   TRUE.
   verbose write stuff if TRUE (optional).

Value
a list containing :
   trace a matrix, the trace of the Gibbs Sampler.
   param a list containing parameters used to run the function.
Examples

```r
param_sim <- list(Q=1,n=25,p=50,grids_lim=list(c(0,1)),iter=2e2,K=2)
data_sim <- sim(param_sim,verbose=FALSE)
res_Bliss_Gibbs_Sampler <- Bliss_Gibbs_Sampler(data_sim,param_sim)
theta_1 <- res_Bliss_Gibbs_Sampler$trace[1,]
theta_1
```

Bliss_Simulated_Annealing

**Bliss_Simulated_Annealing**

Description

A Simulated Annealing algorithm to compute the Bliss estimate.

Usage

```r
Bliss_Simulated_Annealing(
  beta_sample,
  posterior_sample,
  param,
  verbose_cpp = FALSE
)
```

Arguments

- `beta_sample` a matrix. Each row is a coefficient function computed from the posterior sample.
- `posterior_sample` a list resulting from the `Bliss_Gibbs_Sampler` function.
- `param` a list containing:
  - `grids` a list of numerical vectors, the qth vector is the grid of time points for the qth functional covariate.
  - `basis` a character (optional). The possible values are "uniform" (default), "epanechnikov", "gauss" and "triangular" which correspond to different basis functions to expand the coefficient function and the functional covariates.
  - `burnin` an integer (optional), the number of iteration to drop from the posterior sample.
  - `iter_sann` an integer (optional), the number of iteration of the Simulated Annealing algorithm.
  - `k_max` an integer (optional), the maximal number of intervals for the Simulated Annealing algorithm.
  - `l_max` an integer (optional), the maximal interval length for the Simulated Annealing algorithm.
  - `Temp_init` a nonnegative value (optional), the initial temperature for the cooling function of the Simulated Annealing algorithm.
Q an integer, the number of functional covariates.
p a vector of integers, the numbers of time point of each functional covariate.
verbose write stuff if TRUE (optional).
verbose_cpp Rcpp writes stuff if TRUE (optional).

Value

a list containing:

Bliss_estimate a numerical vector, corresponding to the Bliss estimate of the coefficient function.
Smooth_estimate a numerical vector, which is the posterior expectation of the coefficient function for each time points.
trace a matrix, the trace of the algorithm.

Examples

data(data1)
data(param1)
data(res_bliss1)
param1$Q <- length(data1$x)
param1$grids <- data1$grids
param1$p <- sapply(data1$grids,length)

posterior_sample <- res_bliss1$posterior_sample
beta_sample <- compute_beta_sample(posterior_sample,param1)

res_sann <- Bliss_Simulated_Annealing(beta_sample,posterior_sample,param1)

Description

Define a Fourier basis to simulate functional covariate observations.

Usage

build_Fourier_basis(grid, dim, per = 2 * pi)

Arguments

grid a numerical vector.
dim a numerical value. It corresponds to dim(basis)/2.
per a numerical value which corresponds to the period of the sine and cosine functions.
**change_grid**

**Details**
See the `sim_x` function.

**Value**
a matrix. Each row is an functional observation evaluated on the grid time points.

**Examples**

```r
# See the function \code{sim_x}.
```

---

**Description**
Compute a function (evaluated on a grid) on a given (finer) grid.

**Usage**

`change_grid(fct, grid, new_grid)`

**Arguments**

- `fct`: a numerical vector, the function to evaluate on the new grid.
- `grid`: a numerical vector, the initial grid.
- `new_grid`: a numerical vector, the new grid.

**Value**
a numerical vector, the approximation of the function on the new grid.

**Examples**

```r
grid <- seq(0,1,l=1e1)
new_grid <- seq(0,1,l=1e2)
fct <- 3*grid^2 + sin(grid*2*pi)
plot(grid,fct,type="o",lwd=2,cex=1.5)
lines(new_grid,change_grid(fct,grid,new_grid),type="o",col="red",cex=0.8)
```
choose_beta

Description

Compute a coefficient function for the Function Linear Regression model.

Usage

choose_beta(param)

Arguments

param 

A list containing:

grid 

A numerical vector, the time points.

p 

A numerical value, the length of the vector grid.

shape 

A character vector: "smooth", "random_smooth", "simple", "simple_bis", "random_simple", "sinusoid", "flat_sinusoid" and "sharp"

Details

Several shapes are available.

Value

A numerical vector which corresponds to the coefficient function at given times points (grid).

Examples

### smooth

param <- list(p=100, grid=seq(0,1,length=100), shape="smooth")

beta_function <- choose_beta(param)

plot(param$grid, beta_function, type="l")

### random_smooth

param <- list(p=100, grid=seq(0,1,length=100), shape="random_smooth")

beta_function <- choose_beta(param)

plot(param$grid, beta_function, type="l")

### simple

param <- list(p=100, grid=seq(0,1,length=100), shape="simple")

beta_function <- choose_beta(param)

plot(param$grid, beta_function, type="s")

### simple_bis

param <- list(p=100, grid=seq(0,1,length=100), shape="simple_bis")

beta_function <- choose_beta(param)

plot(param$grid, beta_function, type="s")

### random_simple

param <- list(p=100, grid=seq(0,1,length=100), shape="random_simple")

beta_function <- choose_beta(param)

plot(param$grid, beta_function, type="s")
compute_beta_posterior_density

### sinusoid
param <- list(p=100, grid=seq(0,1,length=100), shape="sinusoid")
beta_function <- choose_beta(param)
plot(param$grid, beta_function, type="l")

### flat_sinuoid
param <- list(p=100, grid=seq(0,1,length=100), shape="flat_sinuoid")
beta_function <- choose_beta(param)
plot(param$grid, beta_function, type="l")

### sharp
param <- list(p=100, grid=seq(0,1,length=100), shape="sharp")
beta_function <- choose_beta(param)
plot(param$grid, beta_function, type="l")

compute_beta_posterior_density

compute_beta_posterior_density

---

Description

Compute the posterior density of the coefficient function.

Usage

compute_beta_posterior_density(beta_sample, param)

Arguments

- **beta_sample**: a matrix. Each row is a coefficient function computed from the posterior sample.
- **param**: a list containing:
  - **grid**: a numerical vector, the time points.
  - **lims_estimate**: a numerical vector, the time points.
  - **burnin**: an integer (optional), the number of iteration to drop from the Gibbs sample.
  - **lims_kde**: an integer (optional), correspond to the lims option of the kde2d function.
  - **new_grid**: a numerical vector (optional) to compute beta sample on a different grid.
  - **thin**: an integer (optional) to thin the posterior sample.

Details

The posterior densities corresponds to approximations of the marginal posterior distributions (of beta(t) for each t). The sample is thinned in order to reduce the correlation and the computational time of the function kde2d.
compute_beta_sample

Value
An approximation of the posterior density on a two-dimensional grid (corresponds to the result of the kde2d function).

Examples
library(RColorBrewer)
data(data1)
data(param1)
data(res_bliss1)
param1$grids <- data1$grids
param1$p <- sapply(data1$grids,length)
param1$Q <- length(data1$x)

density_estimate <- compute_beta_posterior_density(res_bliss1$beta_sample,param1)

compute_beta_sample compute_beta_sample

Description
Compute the posterior coefficient function from the posterior sample.

Usage
compute_beta_sample(posterior_sample, param)

Arguments
posterior_sample
a list provided by the function Bliss_Gibbs_Sampler.

param
a list containing:
  K a vector of integers, corresponding to the numbers of intervals for each covariate.
  grids a numerical vector, the observation time points.
  basis a character (optional) among : "uniform" (default), "epanechnikov", "gauss" and "triangular" which correspond to different basis functions to expand the coefficient function and the functional covariates.
  Q an integer, the number of functional covariates.
  p a vector of integers, the numbers of time points of each functional covariate.

Value
a matrix containing the coefficient function posterior sample.
**Examples**

```r
data(data1)
data(param1)
data(res_bliss1)
param1$grids <- data1$grids
param1$p <- sapply(data1$grids, length)
param1$Q <- length(data1$x)
beta_sample <- compute_beta_sample(posterior_sample=res_bliss1$posterior_sample,
                                   param=param1)
```

**Description**

Compute summaries of Gibbs Sampler chains.

**Usage**

```r
compute_chains_info(chain, param)
```

**Arguments**

- `chain`: a list given by the `Bliss_Gibbs_Sampler` function.
- `param`: a list containing:
  - `K`: a vector of integers, corresponding to the numbers of intervals for each covariate.
  - `grids`: a numerical vector, the observation time points.
  - `basis`: a vector of characters (optional) among: "uniform" (default), "epanechnikov", "gauss" and "triangular" which correspond to different basis functions to expand the coefficient function and the functional covariates.

**Value**

Return a list containing the estimates of mu and sigma_sq, the Smooth estimate and the chain autocorrelation for mu, sigma_sq and beta.

**Examples**

```r
a=1
```
compute_random_walk

Description
Compute a (Gaussian) random walk.

Usage
compute_random_walk(n, p, mu, sigma, start = rep(0, n))

Arguments
- \( n \) an integer, the number of random walks.
- \( p \) an integer, the length of the random walks.
- \( mu \) a numerical vector, the mean of the random walks.
- \( sigma \) a numerical value which is the standard deviation of the gaussian distribution used to compute the random walks.
- \( start \) a numerical vector (optional) which is the initial value of the random walks.

Details
See the \texttt{sim.x} function.

Value
a matrix where each row is a random walk.

Examples
# see the \texttt{sim.x()} function.

compute_starting_point_sann

Description
Compute a starting point for the Simulated Annealing algorithm.

Usage
compute_starting_point_sann(beta_expe)
Arguments

**beta_expe** a numerical vector, the expectation of the coefficient function posterior sample.

Value

a matrix with 3 columns: "m", "l" and "b". The two first columns define the begin and the end of the intervals and the third gives the mean values of each interval.

Examples

data(res_bliss1)
mystart <- compute_starting_point_sann(apply(res_bliss$beta_sample[[1]], 2, mean))

corr_matrix

diagonal ksi

Description

Compute an autocorrelation matrix.

Usage

corr_matrix(diagonal, ksi)

Arguments

diagonal a numerical vector corresponding to the diagonal.

ksi a numerical value, related to the correlation.

Value

a symmetric matrix.

Examples

### Test 1: weak autocorrelation
ksi <- 1
diagVar <- abs(rnorm(100, 50, 5))
Sigma <- corr_matrix(diagVar, ksi^2)
persp(Sigma)

### Test 2: strong autocorrelation
ksi <- 0.2
diagVar <- abs(rnorm(100, 50, 5))
Sigma <- corr_matrix(diagVar, ksi^2)
persp(Sigma)
data1

\textit{data1} \hspace{1cm} \textit{a list of data}

**Description**

A data object for bliss model

**Usage**

```
data1
```

**Format**

- \textit{a list of data}
- \textit{y} y coordinate
- \textit{x} x coordinate
- \textit{betas} the coefficient function used to generate the data
- \textit{grids} the grid of the observation times

**determine_intervals**

\textit{determine_intervals} \hspace{1cm} \textit{determine_intervals}

**Description**

Determine for which intervals a function is nonnull.

**Usage**

```
determine_intervals(beta_fct)
```

**Arguments**

- \textit{beta_fct} a numerical vector.

**Value**

a matrix with 3 columns: "begin", "end" and "value". The two first columns define the begin and the end of the intervals and the third gives the mean values of each interval.
Examples

data(data1)
data(param1)
# result of res_bliss1<-fit_Bliss(data=data1,param=param1)
data(res_bliss1)
intervals <- determine_intervals(res_bliss1$Bliss_estimate[1])
plot(data1$grids[1],res_bliss1$Bliss_estimate[1],type="s")
for(k in 1:nrow(intervals)){
  segments(data1$grids[1][intervals[k,1]],intervals[k,3],
          data1$grids[1][intervals[k,2]],intervals[k,3],col=2,lwd=4)
}

do_need_to_reduce

do_need_to_reduce

Description

Determine if it is required to reduce the size of the grid time points for each functional covariate.

Usage

do_need_to_reduce(param)

Arguments

param a list containing p_threshold the maximum number of time points and p the actual number of time points for each functional covariate.

Value

a boolean value.

Examples

data(param1)
param1$p <- sapply(data1$grids,length)
do_need_to_reduce(param1)
**dposterior**

---

**Description**

Compute (non-normalized) posterior densities for a given parameter set.

**Usage**

```r
dposterior(posterior_sample, data, theta = NULL)
```

**Arguments**

- `posterior_sample`: a list given by the `Bliss_Gibbs_Sampler` function.
- `data`: a list containing
  - `y`: a numerical vector, the outcomes.
  - `x`: a list of matrices, the qth matrix contains the observations of the qth functional covariate at time points given by grids.
- `theta`: a matrix or a vector which contains the parameter set.

**Details**

If the `theta` is NULL, the posterior density is computed from the MCMC sample given in the `posterior_sample`.

**Value**

Return the (log) posterior density, the (log) likelihood and the (log) prior density for the given parameter set.

**Examples**

```r
data(data1)
data(param1)
# result of res_bliss1<-fit_Bliss(data=data1,param=param1)
data(res_bliss1)
# Compute the posterior density of the MCMC sample :
res_poste <- dposterior(res_bliss1$posterior_sample,data1)
```
**Description**

Fit the Bayesian Functional Linear Regression model (with Q functional covariates).

**Usage**

```r
fit_Bliss(
  data,
  param,
  sann = TRUE,
  compute_density = TRUE,
  support_estimate = TRUE,
  sann_trace = FALSE,
  verbose = TRUE
)
```

**Arguments**

- **data**
  - a list containing:
    - `y` a numerical vector, the outcomes.
    - `x` a list of matrices, the qth matrix contains the observations of the qth functional covariate at time points given by `grids`.
    - `grids` a list of numerical vectors, the qth vector is the grid of time points for the qth functional covariate.

- **param**
  - a list containing:
    - `iter` an integer, the number of iterations of the Gibbs sampler algorithm.
    - `K` a vector of integers, corresponding to the numbers of intervals for each covariate.
    - `basis` a character vector (optional). The possible values are "uniform" (default), "epanechnikov", "gauss" and "triangular" which correspond to different basis functions to expand the coefficient function and the functional covariates.
    - `burnin` an integer (optional), the number of iteration to drop from the posterior sample.
    - `iter_sann` an integer (optional), the number of iteration of the Simulated Annealing algorithm.
    - `k_max` an integer (optional), the maximal number of intervals for the Simulated Annealing algorithm.
    - `l_max` an integer (optional), the maximal interval length for the Simulated Annealing algorithm.
    - `lims_kde` an integer (optional), correspond to the `lims` option of the `kde2d` function.
new_grids a list of Q vectors (optional) to compute beta samples on different grids.

Temp_init a nonnegative value (optional), the initial temperature for the cooling function of the Simulated Annealing algorithm.

thin an integer (optional) to thin the posterior sample.

times_sann an integer (optional), the number of times the algorithm will be executed

times_sann an integer (optional), the number of times the algorithm will be executed

allow_reducing a boolean value (optional), indicate if the function is allowed to reduce the number of sample times of each functional covariate.

verbose_cpp a boolean value (optional). Write stuff from the Rcpp scripts if TRUE.

sann a logical value. If TRUE, the Bliss estimate is computed with a Simulated Annealing Algorithm. (optional)

compute_density a logical value. If TRUE, the posterior density of the coefficient function is computed. (optional)

support_estimate a logical value. If TRUE, the estimate of the coefficient function support is computed. (optional)

sann_trace a logical value. If TRUE, the trace of the Simulated Annealing algorithm is included into the result object. (optional)

verbose write stuff if TRUE (optional).

Value

return a list containing:

alpha a list of Q numerical vector. Each vector is the function alpha(t) associated to a functional covariate. For each t, alpha(t) is the posterior probabilities of the event "the support covers t".

beta_posterior_density a list of Q items. Each item contains a list containing information to plot the posterior density of the coefficient function with the image function.

grid_t a numerical vector: the x-axis.

grid_beta_t a numerical vector: the y-axis.

density a matrix: the z values.

new_beta_sample a matrix: beta sample used to compute the posterior densities.

beta_sample a list of Q matrices. The qth matrix is a posterior sample of the qth functional covariates.

Bliss_estimate a list of numerical vectors corresponding to the Bliss estimates of each functional covariates.

data a list containing the data.

posterior_sample a list of information about the posterior sample: the trace matrix of the Gibbs sampler, a list of Gibbs sampler parameters and the posterior densities.
**support_estimate** a list of support estimates of each functional covariate.

**support_estimate_fct** another version of the support estimates.

**trace_sann** a list of Q matrices which are the trace of the Simulated Annealing algorithm.

Examples

```r
# see the vignette BlissIntro.
```

## Description

Plot an approximation of the posterior density.

## Usage

```r
image_Bliss(beta_posterior_density, param = list(), q = 1, to_print = TRUE)
```

## Arguments

- `beta_posterior_density` a list. The result of the function `compute_beta_posterior_density`.
- `param` an optional list containing arguments: `col_low`, `col_mid`, `col_high`, `ylim`, `xlab`, `ylab`, `title`.
- `q` an integer (optional), the index of the functional covariate to plot.
- `to_print` display the plot if TRUE.

## Examples

```r
data(data1)
data(param1)
data(res_bliss1)
image_Bliss(res_bliss1$beta_posterior_density, param1, q=1)
```
**Description**

Trapezoidal rule to approximate an integral.

**Usage**

```r
integrate_trapeze(x, y)
```

**Arguments**

- `x`: a numerical vector, the discretization of the domain.
- `y`: a numerical value, the discretization of the function to integrate.

**Value**

a numerical value, the approximation.

**Examples**

```r
x <- seq(0,1,le=1e2)
integrate_trapeze(x,x^2)
integrate_trapeze(data1$grids[[1]],t(data1$x[[1]]))
```

---

**Description**

Provide a graphical representation of the functional data with a focus on the detected periods with the Bliss method.

**Usage**

```r
interpretation_plot(data, Bliss_estimate, q = 1, centered = FALSE, cols = NULL)
```
**Arguments**

- **data** a list containing:
  - **y** a numerical vector, the outcomes.
  - **x** a list of matrices, the qth matrix contains the observations of the qth functional covariate at time points given by grids.
  - **grids** a list of numerical vectors, the qth vector is the grid of time points for the qth functional covariate.

- **Bliss_estimate** a numerical vector, the Bliss estimate.

- **q** an integer (optional), the index of the functional covariate to plot.

- **centered** a logical value (optional). If TRUE, the functional data are centered.

- **cols** a numerical vector of colours (optional).

**Examples**

```r
data(data1)
data(param1)
# result of res_bliss1 <- fit_Bliss(data=data1,param=param1,verbose=TRUE)
data(res_bliss1)
interpretation_plot(data=data1,Bliss_estimate=res_bliss1$Bliss_estimate,q=1)
interpretation_plot(data=data1,Bliss_estimate=res_bliss1$Bliss_estimate,q=1,centered=TRUE)
```

**Description**

Add a line to a plot obtained with image_Bliss.

**Usage**

```r
lines_bliss(x, y, col = "black", lty = "solid")
```

**Arguments**

- **x** the coordinates of points in the plot.

- **y** the y coordinates of points in the plot.

- **col** a color.

- **lty** option corresponding to "linetype" of geom_line.
Examples

data(data1)
data(param1)
data(res_bliss1)

image_Bliss(res_bliss1$beta_posterior_density,param1,q=1) +
lines_bliss(res_bliss1$data$grids[[1]],res_bliss1$sSmooth_estimate[[1]])+
lines_bliss(res_bliss1$data$grids[[1]],res_bliss1$Bliss_estimate[[1]],col="purple")

param1  

Description

A list of param for bliss model

Usage

param1

Format

a list of param for bliss model

Q  the number of functional covariates
n  the sample size
p  the number of observation times
beta_shapes  the shapes of the coefficient functions
grids_lim  the range of the observation times
grids  the grids of the observation times
K  the number of intervals for the coefficient function

pdexp  

Description

Probability function of a discretized Exponentiel distribution.

Usage

pdexp(a, l_values)
Arguments

a           a positive value, the mean of the Exponential prior.
1_values    a numerical value, the discrete support of the parameter l.

Value

a numerical vector, which is the probability function on 1_values.

Examples

```r
dexp(10, seq(0, 1, 1))

x <- seq(0, 10, len=1e3)
plot(x, dexp(x, 0.5), lty=2, type="l")
lines(pexp(0.5, 1:10), type="p")
```

Description

Compute the post treatment values.

Usage

```r
post_treatment_bliss(posterior_sample, param, data)
```

Arguments

- `posterior_sample` 
  a list provided by the function `Bliss_Gibbs_Sampler`.
- `param` 
  a list containing:
  - **K**  
    a vector of integers, corresponding to the numbers of intervals for each 
    covariate.
- `data` 
  a list containing required options to run the function `dposterior`.

Value

A list of important post treatment value: BIC, the maximum of the log likelihood and the number of parameters.

Examples

```r
data(data1)
data(param1)
data(res_bliss1)

post_treatment_bliss(res_bliss1$posterior_sample, param1, data1)
```
predict_bliss

Description
Compute predictions.

Usage
predict_bliss(x, grids, burnin, posterior_sample, Smooth_estimate)

Arguments
- x: a list containing the design matrices related to the functional covariates. Must be similar to the result of the function sim_x.
- grids: a list of numerical vectors, the qth vector is the grid of time points for the qth functional covariate.
- burnin: an integer (optional), the number of iteration to drop from the posterior sample.
- posterior_sample: a list provided by the function Bliss_Gibbs_Sampler.
- Smooth_estimate: one of the objects resulting from Bliss_Simulated_Annealing.

Value
A vector of predictions for each individual data x.

Examples
```r
data(data1)
data(param1)
data(res_bliss1)
predict_bliss(data1$x, data1$grids, 50, res_bliss1$posterior_sample, res_bliss1$smooth_estimate)
```

predict_bliss_distribution

Description
Compute the distribution of the predictions.
Usage
predict_bliss_distribution(x, grids, burnin, posterior_sample, beta_sample)

Arguments
- **x**: a list containing the design matrices related to the functional covariates. Must be similar to the result of the function `sim_x`.
- **grids**: a list of numerical vectors, the qth vector is the grid of time points for the qth functional covariate.
- **burnin**: an integer (optional), the number of iteration to drop from the posterior sample.
- **posterior_sample**: a list provided by the function Bliss_Gibbs_Sampler.
- **beta_sample**: a list provided by the function compute_beta_sample.

Value
A matrix containing predictions for each individual data x.

Examples
```
data(data1)
data(param1)
data(res_bliss1)
predict_bliss_distribution(data1$x, data1$grids, 50, res_bliss1$posterior_sample, res_bliss1$beta_sample)
```

---

**printbliss**

Print a bliss Object

Description
Print a bliss Object

Usage
printbliss(x, ...)

Arguments
- **x**: input bliss Object
- **...**: further arguments passed to or from other methods

Examples
```
# See fit_Bliss() function
```
reduce_x

Description
Reduce the number of time points.

Usage
reduce_x(data, param)

Arguments
- data: similar to fit_Bliss.
- param: a list containing values Q, p and p

Value
a numerical value, the approximation.

Examples
```r
param <- list(Q=1, n=10, p=c(150), grids_lim=list(c(0,1)))
data <- sim(param)
data(param1)
param1$n <- nrow(data$x[[1]])
param1$p <- sapply(data$grids,length)
param1$Q <- length(data$x)
data <- reduce_x(data,param1)
```

res_bliss1

A result of the BliSS method

Description
A result of the BliSS method

Usage
res_bliss1
sigmoid

Format

- a Bliss object (list)
  - **alpha** a list of Q numerical vector. Each vector is the function \( \alpha(t) \) associated to a functional covariate. For each \( t \), \( \alpha(t) \) is the posterior probabilities of the event "the support covers \( t \)."
  - **beta_posterior_density** a list of Q items. Each item contains a list containing information to plot the posterior density of the coefficient function with the `image` function.
    - `grid_t` a numerical vector: the x-axis.
    - `grid_beta_t` a numerical vector: the y-axis.
    - `density` a matrix: the z values.
    - `new_beta_sample` a matrix: beta sample used to compute the posterior densities.
  - **beta_sample** a list of Q matrices. The \( q \)th matrix is a posterior sample of the \( q \)th functional covariates.
  - **Bliss_estimate** a list of numerical vectors corresponding to the Bliss estimates of each functional covariates.
  - **data** see the description of the object `data1`.
  - **posterior_sample** a list containing (for each chain) the result of the `Bliss_Gibbs_Sampler` function.
  - **Smooth_estimate** a list containing the Smooth estimates of the coefficient functions.
  - **support_estimate** a list containing the estimations of the support.
  - **support_estimate_fct** a list containing the estimation of the support.
  - **trace_sann** a list containing (for each chain) the trace of the Simulated Annealing algorithm.

---

### sigmoid

Description

Compute a sigmoid function.

Usage

```r
sigmoid(x, asym = 1, v = 1)
```

Arguments

- `x` a numerical vector, time points.
- `asym` a numerical value (optional), the asymptote of the sigmoid function.
- `v` a numerical value (optional), related to the slope at the origin.

Details

see the function `sim_x`.
Value

a numerical vector.

Examples

```r
## Test 1 :
x <- seq(-7,7,0.1)
y <- sigmoid(x)
plot(x,y,type="l",main="Sigmoid function")
## Test 2 :
x <- seq(-7,7,0.1)
y <- sigmoid(x)
y2 <- sigmoid(x,asym=0.5)
y3 <- sigmoid(x,v = 5)
plot(x,y,type="l",main="Other sigmoid functions")
lines(x,y2,col=2)
lines(x,y3,col=3)
```

Description

Compute a sharp sigmoid function.

Usage

`sigmoid_sharp(x, loc = 0, ...)`

Arguments

- `x`  
a numerical vector, time points.
- `loc`  
a numerical value (optional), the time of the sharp.
- `...`  
Arguments (optional) for the function `sigmoid`.

Details

see the function `sim.x`.

Value

a numerical vector.
Examples

```r
## Test 1:
x <- seq(-7,7,0.1)
y <- sigmoid_sharp(x)
plot(x,y,type="l",main="Sharp sigmoid")
## Test 2:
x <- seq(-7,7,0.1)
y <- sigmoid_sharp(x,loc=3)
y2 <- sigmoid_sharp(x,loc=3,asym=0.5)
y3 <- sigmoid_sharp(x,loc=3,v = 5)
plot(x,y,type="l",main="Other sharp sigmoids")
lines(x,y2,col=2)
lines(x,y3,col=3)
```

Description

Simulate a dataset for the Function Linear Regression model.

Usage

```r
sim(param, verbose = FALSE)
```

Arguments

- **param**: a list containing:
  - `beta_shapes`: a character vector. The qth item indicates the shape of the coefficient function associated to the qth functional covariate.
  - `beta_functions`: a list containing numerical vectors to define the beta functions
  - `n`: an integer, the sample size.
  - `p`: a vector of integers, the qth component is the number of times for the qth covariate.
  - `Q`: an integer, the number of functional covariates.
  - `autocorr_diag`: a list of numerical vectors (optional), the qth vector is the diagonal of the autocorrelation matrix of the qth functional covariate.
  - `autocorr_spread`: a vector of numerical values (optional) which are related to the autocorrelation of the functional covariates.
  - `grids`: a list of numerical vectors (optional), the qth vector is the grid of time points for the qth functional covariate.
  - `grids_lim`: a list of numerical vectors (optional), the qth item is the lower and upper boundaries of the domain for the qth functional covariate.
  - `link`: a function (optional) to simulate data from the Generalized Functional Linear Regression model.
  - `mu`: a numerical value (optional), the 'true' intercept of the model.
r a nonnegative value (optional), the signal to noise ratio.
x_shapes a character vector (optional). The qth item indicates the shape of the functional covariate observations.
verbose write stuff if TRUE.

Value

a list containing:

Q an integer, the number of functional covariates.
y a numerical vector, the outcome observations.
x a list of matrices, the qth matrix contains the observations of the qth functional covariate at time points given by grids.
grids a list of numerical vectors, the qth vector is the grid of time points for the qth functional covariate.
betas a list of numerical vectors, the qth vector is the 'true' coefficient function associated to the qth covariate on a grid of time points given with grids.

Examples

library(RColorBrewer)
param <- list(Q=2,n=25,p=c(50,50),grids_lim=list(c(0,1),c(-1,2)))
data <- sim(param)
data$y
cols <- colorRampPalette(brewer.pal(9,"YlOrRd"))(10)
q=2
matplot(data$grids[[q]],t(data$x[[q]]),type="l",lty=1,col=cols)
plot(data$grids[[q]],data$betas[[q]],type="l")
abline(h=0,lty=2,col="gray")

Description

Simulate functional covariate observations.

Usage

sim_x(param)
**Arguments**

- **param**
  - a list containing:
    - **grid** a numerical vector, the observation times.
    - **n** an integer, the sample size.
    - **p** an integer, the number of observation times.
    - **diagVar** a numerical vector (optional), the diagonal of the autocorrelation matrix.
    - **dim** a numerical value (optional), the dimension of the Fourier basis, if "shape" is "Fourier" or "Fourier2".
    - **ksi** a numerical value (optional) related to the observations correlation.
    - **x_shape** a character vector (optional), the shape of the observations.

**Details**

Several shape are available for the observations: "Fourier", "Fourier2", "random_walk", "random_sharp", "uniform", "gaussian", "mvgauss", "mvgauss_different_scale", "mvgauss_different_scale2", "mvgauss_different_scale3" and "mvgauss_different_scale4".

**Value**

A matrix which contains the functional covariate observations at time points given by grid.

**Examples**

```r
library(RColorBrewer)
### uniform
param <- list(n=15,p=100,grid=seq(0,1,length=100),x_type="uniform")
x <- sim_x(param)
cols <- colorRampPalette(brewer.pal(9,"YlOrRd"))(15)
matplot(param$grid,t(x),type="l",lty=1,col=cols)
```

**Description**

Compute the support estimate.

**Usage**

`support_estimation(beta_sample, param)`

**Arguments**

- **beta_sample** the result of the function `compute_beta_sample`.
- **param** a list containing the value `Q` and an optional parameter `gamma`. 
Value

a list containing:

*alpha* a numerical vector. The approximated posterior probabilities that the coefficient function support covers \( t \) for each time points \( t \).

*estimate* a numerical vector, the support estimate.

*estimate_fct* a numerical vector, another version of the support estimate.

Examples

```r
data(data1)
data(param1)
data(res_bliss1)
param1$Q <- length(data1$x)

res_support <- support_estimation(res_bliss1$beta_sample,param1)
```

Description

Check if a number belong to a given interval.

Usage

```
value %between% interval
```

Arguments

- `value`: a numerical value.
- `interval`: a numerical vector: \((\text{lower},\text{upper})\).

Value

a logical value.

Examples

1. `1 %between% c(0,2)`
2. `2 %between% c(0,2)`
3. `3 %between% c(0,2)`
Index

* datasets
  data1, 14
  param1, 22
  res_bliss1, 26
  %between%, 32

BIC_model_choice, 2
bliss, 3
bliss-package (bliss), 3
Bliss_Gibbs_Sampler, 4
Bliss_Simulated_Annealing, 5
build_Fourier_basis, 6

change_grid, 7
choose_beta, 8
compute_beta_posterior_density, 9
compute_beta_sample, 10
compute_chains_info, 11
compute_random_walk, 12
compute_starting_point_sann, 12
corr_matrix, 13

data1, 14
determine_intervals, 14
do_need_to_reduce, 15
dposterior, 16

fit_Bliss, 17

image_Bliss, 19
integrate_trapeze, 20
interpretation_plot, 20

lines_bliss, 21

param1, 22
pdexp, 22
post_treatment_bliss, 23
predict_bliss, 24
predict_bliss_distribution, 24
printbliss, 25

reduce_x, 26
res_bliss1, 26
sigmoid, 27
sigmoid_sharp, 28
sim, 29
sim_x, 7, 12, 27, 28, 30
support_estimation, 31

33