Package ‘dfrr’

May 31, 2023

Type Package

Title Dichotomized Functional Response Regression

Version 0.1.5

Maintainer Fatemeh Asgari <fatemeh.asgari@medisin.uio.no>

Description Implementing Function-on-Scalar Regression model in which the response function is dichotomized and observed sparsely. This package provides smooth estimations of functional regression coefficients and principal components for the dichotomized functional response regression (dfrr) model.

License GPL-3

LazyData TRUE

Encoding UTF-8

Depends R (>= 3.5.0), tmvtnorm (>= 1.4-10), fda (>= 5.1.4)

Imports MASS, ggplot2, plotly

Suggests car

URL https://github.com/asgari-fatemeh/dfrr

BugReports https://github.com/asgari-fatemeh/dfrr/issues

RoxygenNote 7.2.3

NeedsCompilation no

Author Fatemeh Asgari [aut, cre],
Saeed Hayati [aut, ctb]

Repository CRAN

Date/Publication 2023-05-31 08:30:02 UTC

R topics documented:

`dfrr-package` *Dichotomizd functional response regression (dfrr) model*

Description

Implementing Function-on-Scalar Regression model, in which the response function is dichotomized and observed sparsely. This function fits the dichotomized functional response regression (dfrr) model as

$$Y_i(t) = I(\beta_0(t) + \beta_1(t) * x_{1i} + \ldots + \beta_{q-1}(t) * x_{(q-1)i} + \varepsilon_i(t) + \epsilon_i(t) \times \sigma^2 > 0),$$

where $I(.)$ is the indicator function, ε_i is a Gaussian random function, and $\epsilon_i(t)$ are iid standard normal for each i and t independent of ε_i. β_k and x_k for $k = 0, 1, \ldots, q-1$ are the functional regression coefficients and scalar covariates, respectively.

Details

@details Implementing Function-on-Scalar Regression model in which the response function is dichotomized and observed sparsely. This package provides smooth estimations of functional regression coefficients and principal components for the dichotomized funtional response regression (dfrr) model. The main function in the dfrr-package is dfrr().

Author(s)

Maintainer: Fatemeh Asgari <fatemeh.asgari@medisin.uio.no>

Authors:

- Saeed Hayati <saeed.hayati@medisin.uio.no> [contributor]

See Also

Useful links:

- https://github.com/asgari-fatemeh/dfrr
- Report bugs at https://github.com/asgari-fatemeh/dfrr/issues

Examples

```
set.seed(2000)
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)

coefs<-coef(dfrr_fit)
  plot(coefs)

fitteds<-fitted(dfrr_fit)
  plot(fitteds)

resids<-residuals(dfrr_fit)
plot(resids)

fpcs<-fpca(dfrr_fit)
plot(fpcs,plot.contour=TRUE,plot.3dsurface = TRUE)

newdata<-data.frame(X=c(1,0))
  preds<-predict(dfrr_fit,newdata=newdata)
  plot(preds)


newdata<-data.frame(X=c(1,0))
newydata<-data.frame(.obs=rep(1,5),.index=c(0.0,0.1,0.2,0.3,0.7),.value=c(1,1,1,0,0))
preds<-predict(dfrr_fit,newdata=newdata,newydata = newydata)
plot(preds)
```

---

basis *Get the basis functions from a dfrr-object*

Description

Returns the basis functions employed in fitting a dfrr-object.

Usage

```
basis(object)
```

Arguments

`object`	a fitted `dfrr`-object obtained from invoking the function `dfrr`.

Value

a basis object used in fitting the functional parameters. The basis object is the one created by the functions `create.*.basis` of the 'fda' package.

Examples

```
set.seed(2000)
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)
coefs<-coef(dfrr_fit,return.fourier.coefs=TRUE)

basis<-basis(dfrr_fit)
evaluated_coefs<-coefs%*%t(fda::eval.basis(time,basis))

#Plotting the regression coefficients
oldpar<-par(mfrow=c(1,2))

plot(time,evaluated_coefs[1,],'l',main="Intercept")
plot(time,evaluated_coefs[2,],'l',main="X")

par(oldpar)
```

coef.dfrr *Get estimated coefficients from a dfrr fit*

Description

Returns estimations of the smooth functional regression coefficients $\beta(t)$. The result is a matrix of either Fourier coefficients or evaluations. See Details.

Usage

```
## S3 method for class 'dfrr'
coef(
  object,
  standardized = NULL,
  unstandardized = !standardized,
  return.fourier.coefs = NULL,
  return.evaluations = !return.fourier.coefs,
  time_to_evaluate = NULL,
  ...
)
```

Arguments

`object` a `dfrr`-object

`standardized, unstandardized`
: a boolean indicating whether stanadrdized/unstandardized regression coefficients are reported. Only standardized regression coefficients are identifiable, thus the arugment is defaults to `standardized=TRUE`.

`return.fourier.coefs, return.evaluations`
: a boolean indicating whether the Fourier coefficients of regression coefficients are returned (`return.fourier.coefs=TRUE`), or evaluations of the regression coefficients (`return.evaluations=TRUE`). Defaults to `return.fourier.coefs=TRUE`.

`time_to_evaluate`
: a numeric vector indicating the set of time points for evaluating the functional regression coefficients, for the case of `return.evaluations=TRUE`.

`...` dot argument, just for consistency with the generic function

Details

This function will return either the Fourier coefficients or the evaluation of estimated coefficients. Fourier coefficients which are reported are based on the a set of basis which can be determined by `basis(dfrr_fit)`. Thus the evaluation of regression coefficients on the set of time points specified by vector `time`, equals to `fitted(dfrr_fit)%*%t(eval.basis(time,basis(dfrr_fit)))`.

Consider that the unstandardized estimations are not identifiable. So, it is recommended to extract and report the standardized estimations.

Value

This function returns a `matrix` of dimension NxM or NxJ, depending the argument 'return.evaluations'. If `return.evaluations=FALSE`, the returned matrix is NxJ, where N denotes the number of functional regression coefficients, (the number of rows of the argument 'newData'), and J denotes the number of basis functions. Then, the NxJ matrix is the fourier coefficients of functional regression coefficients. If `return.evaluations=TRUE`, the returned matrix is NxM, where M is the length of the argument `time_to_evaluate`. Then, the NxM matrix is the functional regression coefficients evaluated at time points given in `time_to_evaluate`.

See Also

`plot.coef.dfrr`

Examples

```
set.seed(2000)
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)
coefs<-coef(dfrr_fit)
plot(coefs)
```

`dfrr` *Dichotomized Functional Response Regression (dfrr)*

Description

Implementing Function-on-Scalar Regression model, in which the response function is dichotomized and observed sparsely. This function fits the dichotomized functional response regression (dfrr) model as

$$Y_i(t) = I(\beta_0(t) + \beta_1(t) * x_{1i} + \ldots + \beta_{q-1}(t) * x_{(q-1)i} + \varepsilon_i(t) + \epsilon_i(t) \times \sigma^2 > 0),$$

where $I(.)$ is the indicator function, ε_i is a Gaussian random function, and $\epsilon_i(t)$ are iid standard normal for each i and t independent of ε_i. β_k and x_k for $k = 0, 1, \ldots, q-1$ are the functional regression coefficients and scalar covariates, respectively.

Usage

```
dfrr(
  formula,
  yind = NULL,
  data = NULL,
  ydata = NULL,
  method = c("REML", "ML"),
  rangeval = NULL,
  basis = NULL,
  times_to_evaluate = NULL,
  ...
)
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class with as.formula: a symbolic description of the model to be fitted.
yind	a vector with length equal to the number of columns of the matrix of functional responses giving the vector of evaluation points $(t_1, ..., t_G)$. If not supplied, yind is set to 1:ncol(<response>).
data	an (optional) data.frame containing the covariate data. the variable terms will be searched from the columns of data, covariates also can be read from the workspace if it is not available in data.
ydata	an (optional) data.frame consists of three columns .obs, .index and .value, supplying the functional responses that are not observed on a regular grid. ydata must be provided if the sampling design is irregular.
method	detrmines the estimation method of functional parameters. Defaults to "REML" estimation.
rangeval	an (optional) vector of length two, indicating the lower and upper limit of the domain of latent functional response. If not specified, it will set by minimum and maximum of yind or .index column of ydata.
basis	an (optional) object of class 'basisfd'. Defaults to cubic bspline basis.
times_to_evaluate	a numeric vector indicating the set of time points for evaluating the functional regression coefficients and principal components.
...	other arguments that can be passed to the inner function AMCEM.

Value

The output is a dfrr-object, which then can be injected into other methods/functions to postprocess the fitted model, including: coef.dfrr,fitted.dfrr, basis, residuals.dfrr, predict.dfrr, fpca, summary.dfrr, model.matrix.dfrr, plot.coef.dfrr, plot.fitted.dfrr, plot.residuals.dfrr, plot.predict.dfrr, plot.fpca.dfrr

Examples

```
set.seed(2000)
N<-50;M<-24
X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)
#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)
plot(dfrr_fit)

#Fitting dfrr model to the Madras Longitudinal Schizophrenia data
data(madras)
ids<-unique(madras$id)
N<-length(ids)

ydata<-data.frame(.obs=madras$id,.index=madras$month,.value=madras$y)

xdata<-data.frame(Age=rep(NA,N),Gender=rep(NA,N))
for(i in 1:N){
  dt<-madras[madras$id==ids[i],]
  xdata[i,]<-c(dt$age[1],dt$gender[1])
}
rownames(xdata)<-ids

#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
#J is the number of basis functions that will be used in estimating the functional parameters.
madras_dfrr<-dfrr(Y~Age+Gender+Age*Gender, data=xdata, ydata=ydata, J=11,T_E=1)


coefs<-coef(madras_dfrr)
plot(coefs)

fpcs<-fpca(madras_dfrr)
plot(fpcs)
plot(fpcs,plot.eigen.functions=FALSE,plot.contour=TRUE,plot.3dsurface = TRUE)

oldpar<-par(mfrow=c(2,2))

fitteds<-fitted(madras_dfrr) #Plot first four fitted functions
  plot(fitteds,id=c(1,2,3,4))
par(oldpar)

resids<-residuals(madras_dfrr)
plot(resids)
```

```
newdata<-data.frame(Age=c(1,1,0,0),Gender=c(1,0,1,0))
  preds<-predict(madras_dfrr,newdata=newdata)
  plot(preds)


newdata<-data.frame(Age=c(1,1,0,0),Gender=c(1,0,1,0))
newydata<-data.frame(.obs=rep(1,5),.index=c(0,1,3,4,5),.value=c(1,1,1,0,0))
preds<-predict(madras_dfrr,newdata=newdata,newydata = newydata)
plot(preds)
```

`fitted.dfrr` *Obtain fitted curves for a dfrr model*

Description

Fitted curves refer to the estimations of latent functional response curves. The results can be either the Fourier coefficients or evaluation of the fitted functions. See Details.

Usage

```
## S3 method for class 'dfrr'
fitted(
  object,
  return.fourier.coefs = NULL,
  return.evaluations = !return.fourier.coefs,
  time_to_evaluate = NULL,
  standardized = NULL,
  unstandardized = !standardized,
  ...
)
```

Arguments

`object` a fitted `dfrr`-object obtained from invoking the function `dfrr`.

`return.fourier.coefs, return.evaluations`
a boolean indicating whether the Fourier coefficients of the fitted curves are returned (`return.fourier.coefs=TRUE`), or evaluations of the fitted curves (`return.evaluations=TRUE`). Defaults to `return.fourier.coefs=TRUE`.

`time_to_evaluate`
a numeric vector indicating the set of time points for evaluating the fitted latent functions, for the case of `return.evaluations=TRUE`.

`standardized, unstandardized`
a boolean indicating whether stanadrdized/unstandardized fitted latent curves is reported. Only standardized fitted curves are identifiable, thus the arugment is defaults to `standardized=TRUE`.

`...` dot argument, just for consistency with the generic function

Details

This function will return either the Fourier coefficients or the evaluation of fitted curves to the binary sequences. Fourier coefficients which are reported are based on the a set of basis which can be determined by `basis(dfrr_fit)`. Thus the evaluation of fitted latent curves on the set of time points specified by vector `time`, equals to `fitted(dfrr_fit)%*%t(eval.basis(time,basis(dfrr_fit)))`.

Consider that the unstandardized estimations are not identifiable. So, it is recommended to extract and report the standardized estimations.

Value

This function returns a `matrix` of dimension NxM or NxJ, depending the argument `return.evaluations`. If `return.evaluations=FALSE`, the returned matrix is NxJ, where N denotes the sample size (the number of rows of the argument 'newData'), and J denotes the number of basis functions. Then, the NxJ matrix is the fourier coefficients of the fitted curves. If `return.evaluations=TRUE`, the returned matrix is NxM, where M is the length of the argument `time_to_evaluate`. Then, the NxM matrix is the fitted curves evaluated at time points given in `time_to_evaluate`.

See Also

`plot.fitted.dfrr`

Examples

```
set.seed(2000)
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)
fitteds<-fitted(dfrr_fit)
plot(fitteds)
```

`fpca` *Functional principal component analysis (fpca) of a dfrr fit*

Description

`fpca()` returns estimations of the smooth principal components/eigen-functions and the corresponding eigen-values of the residual function in the `dfrr` model. The result is a named list containing the vector of eigen-values and the matrix of Fourier coefficients. See Details.

Usage

```
fpca(object, standardized = NULL, unstandardized = !standardized)
```

Arguments

`object` a fitted `dfrr`-object obtained from invoking the function `dfrr`.

`standardized, unstandardized`
a `boolean` indicating whether stanadrdized/unstandardized pricipal components/eigen-functions are reported. Only standardized pricipal components/eigen-functions are identifiable, thus the arugment is defaults to `standardized=TRUE`.

Details

Fourier coefficients which are reported are based on the a set of basis which can be determined by `basis(dfrr_fit)`. Thus the evaluation of pricipal component/eigen-function on the set of time points specified by vector `time`, equals to `fpca(dfrr_fit)%*%t(eval.basis(time,basis(dfrr_fit)))`.

Consider that the unstandardized estimations are not identifiable. So, it is recommended to extract and report the standardized estimations.

Value

`fpca(dfrr_fit)` returns a list containtng the following components:

`values` a vector containing the eigen-values of the standaridized/unstandardized covariance operator of the residual function term in `dfrr` model, sorted in decreasing order.

`vectors` a matrix whose columns contain the Fourier coefficients of the principal components/eigen-functions of the standaridized/unstandardized covariance operator of the residual function term in `dfrr` model, sorted based on the corresponding eigen-values.

See Also

`plot.fpca.dfrr`

Examples

```
set.seed(2000)
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)
```

```
fpcs<-fpca(dfrr_fit)
plot(fpcs,plot.eigen.functions=TRUE,plot.contour=TRUE,plot.3dsurface = TRUE)
```

---

madras *Madras Longitudinal Schizophrenia Study.*

Description

Monthly records of presence/abscence of psychiatric symptom 'thought disorder' of 86 patients over the first year after initial hospitalisation for disease.

Usage

```
madras
```

Format

A data frame with 1032 observations and 5 variables

id identification number of a patient

y response 'thought disorder': 0 = absent, 1 = present

month month since hospitalisation

age age indicator: 0 = less than 20 years, 1 = 20 or over

gender sex indicator: 0 = male, 1 = female

Source

Diggle PJ, Heagerty P, Liang KY, Zeger SL (2002). The analysis of Longitudinal Data, second ed., pp. 234-43. Oxford University Press, Oxford.
<http://faculty.washington.edu/heagerty/Books/AnalysisLongitudinal/datasets.html>

References

Jokinen J. Fast estimation algorithm for likelihood-based analysis of repeated categorical responses. *Computational Statistics and Data Analysis* 2006; 51:1509-1522.

model.matrix.dfrr *Obtain model matrix for a dfrr fit*

Description

Obtain model matrix for a dfrr fit

Usage

```
## S3 method for class 'dfrr'
model.matrix(object, ...)
```

Arguments

object	a dfrr-object
...	dot argument, just for consistency with the generic function

Details

'@return This function returns the model matrix.

plot.coef.dfrr *Plot dfrr coefficients*

Description

Plot a `coef.dfrr` object. The output is the plot of regression coefficients.

Usage

```
## S3 method for class 'coef.dfrr'
plot(x, select = NULL, ask.hit.return = TRUE, ...)
```

Arguments

x	a `coef.dfrr`-object.
select	a vector of length one or more of indices of regression coefficients to plot.
ask.hit.return	a boolean indicating whether to wait for interaction of the user between any two plots.
...	graphical parameters passed to `plot`.

Value

This function generates the plot of functional regression coefficients.

Examples

```
set.seed(2000)
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)
coefs<-coef(dfrr_fit)
plot(coefs)
```

`plot.dfrr` *Plot a dfrr fit*

Description

Plot the regression coefficients, principal components, kernel function and residuals of a `dfrr`-object.

Usage

```
## S3 method for class 'dfrr'
plot(x, plot.kernel = TRUE, ...)
```

Arguments

`x`	the output of the function fitted.dfrr
`plot.kernel`	a boolean indicating whether plots the kernel function or not. `ggplot2`-package and `plotly`-package is required to plot contour and 3d surface of kernel function.
`...`	graphical parameters passed to `plot.coef.dfrr`

Details

The contour plot of the kernel function is produced if the package `ggplot2` is installed. Plotting the 3d surface of the kernel function is also depends on the package `plotly`. To produce the qq-plot, the package `car` must be installed.

Value

This function generates a set of plots, including functional regression coefficients, principal components, 2-d contour and 3d-surface of kernel function, and QQ-plot of residuals.

Examples

```
set.seed(2000)
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)
plot(dfrr_fit)
```

`plot.fitted.dfrr` *Plot dfrr fitted latent functions*

Description

Plot a `fitted.dfrr` object.

Usage

```
## S3 method for class 'fitted.dfrr'
plot(
  x,
  id = NULL,
  main = NULL,
  col = "blue",
  lwd = 2,
  lty = "solid",
  cex.circle = 1,
  col.circle = "black",
  ylim = NULL,
  ...
)
```

Arguments

x	the output of the function fitted.dfrr
id	a vector of length one or more containing subject ids to plot. Must be matched with `rownames(<response>)` or the `.obs` column of `ydata`. Defaults to all subject ids.

`main` a vector of length one or `length(id)` containing the title of plots.

`col, lwd, lty, ...`
graphical parameters passed to `plot`

`cex.circle, col.circle`
size and color of circles and filled circles.

`ylim` a vector of length two indicating the range of y-axis of the plot.

Details

The output is the plot of latent curves over the observed binary sequence. The binary sequence is illustrated with circles and filled circles for the values of zero and one, respectively.

Value

This function generates plot of fitted curves.

Examples

```
set.seed(2000)
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)
fitteds<-fitted(dfrr_fit)
plot(fitteds)
```

`plot.fpca.dfrr` *Plot dfrr functional principal components*

Description

Plot a `fpca.dfrr` object.

Usage

```
## S3 method for class 'fpca.dfrr'
plot(
  x,
  plot.eigen.functions = TRUE,
```

```
  select = NULL,
  plot.contour = FALSE,
  plot.3dsurface = FALSE,
 plot.contour.pars = list(breaks = NULL, minor_breaks = NULL, n.breaks = NULL, labels =
   NULL, limits = NULL, colors = NULL, xlab = NULL, ylab = NULL, title = NULL),
 plot.3dsurface.pars = list(xlab = NULL, ylab = NULL, zlab = NULL, title = NULL, colors
   = NULL),
  ask.hit.return = TRUE,
  ...
)
```

Arguments

`x`	a `fpca.dfrr`-object to be plotted. It is the output of the function `fpca()`
`plot.eigen.functions`	a `boolean` indicating whether to print the principal components/eigen-functions. Defaults to TRUE.
`select`	a vector of length one or more of indices of eigenfunctions to be plotted.
`plot.contour`	a `boolean` indicating whether to print the contour plot of the kernel function. It requires `ggplot2-package` to be installed. Defaults to FALSE.
`plot.3dsurface`	a `boolean` indicating whether to print the 3d surface plot of the kernel function. It requires the package `plotly` to be installed. Defaults to FALSE.
`plot.contour.pars`	a named list of graphical parameters passed to the function `ggplot`.
`plot.3dsurface.pars`	a named list of graphical parameters passed to the function `plot_ly`.
`ask.hit.return`	a boolean indicating whether to wait for interaction of the user between any two plots.
`...`	graphical parameters passed to `plot` function in drawing 2D eigenfunctions.

Details

This function plots the functional principal components, contour plot and 3d surface of the kernel function.

If `ggplot2-package` is installed, the contour plot of the kernel function is produced by setting the argument `plot.contour=TRUE`. Some graphical parameters of the contour plot can be modified by setting the (optional) argument `plot.contour.pars`.

If the package `plotly` is installed, the 3d surface of the kernel function is produced by setting the argument `plot.3dsurface=TRUE`. Some graphical parameters of the 3d surface can be modified by setting the (optional) argument `plot.3dsurface.pars`.

Value

This function generates the plot of principal components.

Examples

```
set.seed(2000)
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

 #The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)
fpcs<-fpca(dfrr_fit)
plot(fpcs,plot.eigen.functions=TRUE,plot.contour=TRUE,plot.3dsurface=TRUE)
```

---

plot.predict.dfrr *Plot dfrr predictions*

Description

Plot a `predict.dfrr` object.

Usage

```
## S3 method for class 'predict.dfrr'
plot(
  x,
  id = NULL,
  main = id,
  col = "blue",
  lwd = 2,
  lty = "solid",
  cex.circle = 1,
  col.circle = "black",
  ylim = NULL,
  ...
)
```

Arguments

x	a `predict.dfrr`-object
id	a vector of length one or more containing subject ids to plot. Must be matched with `rownames(newdata)`. Defaults to all subject ids.
main	a vector of length one or `length(id)` containing the title of plots.

`col, lwd, lty, ...`
: graphical parameters passed to `plot`

`cex.circle, col.circle`
: size and color of circles and filled circles.

`ylim`
: a vector of length two indicating the range of y-axis of the plot.

Details

The output is the plot of predictions of latent functions given the new covariates. For the case in which `newydata` is also given, the predictions are plotted over the observed binary sequence. The binary sequence is illustrated with circles and filled circles for the values of zero and one, respectively.

Value

This function generates the plot of predictions.

References

Choi, H., & Reimherr, M. A geometric approach to confidence regions and bands for functional parameters . *Journal of the Royal Statistical Society, Series B Statistical methodology* 2018; 80:239-260.

Examples

```
set.seed(2000)
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

 #The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)

newdata<-data.frame(X=c(1,0))
  preds<-predict(dfrr_fit,newdata=newdata)
  plot(preds)

newdata<-data.frame(X=c(1,0))
newydata<-data.frame(.obs=rep(1,5),.index=c(0.0,0.1,0.2,0.3,0.7),.value=c(1,1,1,0,0))
preds<-predict(dfrr_fit,newdata=newdata,newydata = newydata)
plot(preds)
```

plot.residuals.dfrr *QQ-plot for dfrr residuals*

Description

The output gives the qq-plot of estimated measurment error.

Usage

```
## S3 method for class 'residuals.dfrr'
plot(x, ...)

## S3 method for class 'dfrr'
qq(x, ...)
```

Arguments

x	a residuals.dfrr-object.
...	graphical parameters passed to car::qqPlot

Value

This function generates the QQ-plot of residuals.

Examples

```
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)
resid<-residuals(dfrr_fit)
plot(resid)
# We can also use the qq function to draw the QQ-plot.
```

`predict.dfrr` *Prediction for dichotomized function-on-scalar regression*

Description

Takes a `dfrr`-object created by `dfrr()` and returns predictions given a new set of values for a model covariates and an optional `ydata`-like `data.frame` of observations for the dichotomized response.

Usage

```
## S3 method for class 'dfrr'
predict(
  object,
  newdata,
  newydata = NULL,
  standardized = NULL,
  unstandardized = !standardized,
  return.fourier.coefs = NULL,
  return.evaluations = !return.fourier.coefs,
  time_to_evaluate = NULL,
  ...
)
```

Arguments

`object` a fitted `dfrr`-object obtained from invoking the function `dfrr`.

`newdata` a `data.frame` containing the values of all of the model covariates at which the latent functional response is going to be predicted.

`newydata` (optional) a `ydata`-like `data.frame` containing the values of dichotomized response sparsly observed in the domain of function.

`standardized, unstandardized`
a `boolean` indicating whether stanadrdized/unstandardized predictions are reported. Defaults to `standardized=TRUE`.

`return.fourier.coefs, return.evaluations`
a `boolean` indicating whether the Fourier coefficients of predictions are returned (`return.fourier.coefs=TRUE`), or evaluations of the predictions (`return.evaluations=TRUE`). Defaults to `return.evaluations=TRUE`.

`time_to_evaluate`
a numeric vector indicating the set of time points for evaluating the predictions, for the case of `return.evaluations=TRUE`.

`...` dot argument, just for consistency with the generic function

Details

This function will return either the Fourier coefficients or the evaluation of predictions. Fourier coefficients which are reported are based on the a set of basis which can be determined by basis(dfrr_fit). Thus the evaluation of predictions on the set of time points specified by vector time, equals to fitted(dfrr_fit,return.fourier.coefs=T)%*%t(eval.basis(time,basis(dfrr_fit))).

Value

This function returns a matrix of dimension NxM or NxJ, depending the argument 'return.evaluations'. If return.evaluations=FALSE, the returned matrix is NxJ, where N denotes the sample size (the number of rows of the argument 'newData'), and J denotes the number of basis functions. Then, the NxJ matrix is the fourier coefficients of the predicted curves. If return.evaluations=TRUE, the returned matrix is NxM, where M is the length of the argument time_to_evaluate. Then, the NxM matrix is the predicted curves evaluated at time points given in time_to_evaluate.

See Also

plot.predict.dfrr

Examples

```
set.seed(2000)
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)

newdata<-data.frame(X=c(1,0))
  preds<-predict(dfrr_fit,newdata=newdata)
  plot(preds)

newdata<-data.frame(X=c(1,0))
newydata<-data.frame(.obs=rep(1,5),.index=c(0.0,0.1,0.2,0.3,0.7),.value=c(1,1,1,0,0))
preds<-predict(dfrr_fit,newdata=newdata,newydata = newydata)
plot(preds)
```

qq — *qq-plot Generic function*

Description

This is a generic function for qq() method.

Usage

```
qq(x, ...)
```

Arguments

`x`	an `dfrr-fit` object
`...`	extra parameters passed to S3 methods

Value

This function generates the QQ-plot of residuals.

residuals.dfrr — *Obtain residuals for a dfrr model*

Description

Returns the residuals of a fitted `dfrr` model. A `dfrr` model is of the form:

$$Y_i(t) = I(W_i(t) > 0),$$

in which $I(.)$ is the indicator function and $W_i(t) = Z_i(t) + \epsilon_i(t) \times \sigma^2$, where $Z_i(t)$ is the functional part of the model and $epsilon_i(t) \times \sigma^2$ is the measurement error. The functional part of the model, consisting a location and a residual function of the form:

$$Z_i(t) = \sum_{j=1}^{q} \beta_j(t) * x_{ji} + \varepsilon_i(t),$$

and $\epsilon_i(t)$ are iid standard normal for each i and t. The residuals reported in the output of this functions is the estimation of the measurement error of the model i.e. $\epsilon_i(t) \times \sigma^2$, which is estimated by:

$$E(W_i(t) - Z_i(t) \mid Y_i(t)).$$

Usage

```
## S3 method for class 'dfrr'
residuals(object, standardized = NULL, unstandardized = !standardized, ...)
```

Arguments

`object` a fitted `dfrr`-object obtained from invoking the function `dfrr`.

`standardized, unstandardized`
a `boolean` indicating whether stanadrdized/unstandardized residuals are reported. Defaults to `standardized=TRUE`.

`...` dot argument, just for consistency with the generic function

Value

This function returns either a `matrix` or a `data.frame`. If the argument ydata is specified, the return value is 'ydata' with a column added, namely 'residual'. Otherwise, the return value is a matrix of residuals of dimension NxM where N is the number of sample curves, and M is the length of argument 'yind' passed to the function `dfrr`.

See Also

`plot.residuals.dfrr`, `qq.dfrr`

Examples

```
set.seed(2000)
N<-50;M<-24

X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)

#The argument T_E indicates the number of EM algorithm.
#T_E is set to 1 for the demonstration purpose only.
#Remove this argument for the purpose of converging the EM algorithm.
dfrr_fit<-dfrr(Y~X,yind=time,T_E=1)
resid<-residuals(dfrr_fit)


plot(resid)
# We can also use the qq function to draw the QQ-plot.
```

`simulate_simple_dfrr` *Simulating a Simple* `dfrr` *Model*

Description

Simulation from a simple dfrr model:

$$Y_i(t) = I(\beta_0(t) + \beta_1(t) * x_i + \varepsilon_i(t) + \epsilon_i(t) \times \sigma^2 > 0),$$

where $I(.)$ is the indicator function, ε_i is a Gaussian random function, and $\epsilon_i(t)$ are iid standard normal for each i and t independent of ε_i. For demonstration purpose only.

Usage

```
simulate_simple_dfrr(
  beta0 = function(t) {
     cos(pi * t + pi)
 },
  beta1 = function(t) {
     2 * t
 },
  X = rnorm(50),
  time = seq(0, 1, length.out = 24),
  sigma2 = 0.2
)
```

Arguments

`beta0, beta1`	(optional) functional intercept and slope parameters
`X`	an (optional) vector consists of scalar covariate
`time`	an (optional) vector of time points for which, each sample curve is observed at.
`sigma2`	variance of the measurement error in the `dfrr` model

Value

This function returns a martix of binary values of dimension NxM where N denotes the length of X and M stands for the length of time.

Examples

```
N<-50;M<-24
X<-rnorm(N,mean=0)
time<-seq(0,1,length.out=M)
Y<-simulate_simple_dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,time=time)
```

`summary.dfrr` *Summary for a dfrr fit*

Description

Summarise a fitted `dfrr`-object. Not implemented.

Usage

```
## S3 method for class 'dfrr'
summary(object, ...)
```

Arguments

`object`	a `dfrr`-object
`...`	dot argument, just for consistency with the generic function

Value

The function `summary.dfrr` computes and returns a list of summary statistics of the fitted dfrr model given in `dfrr`-object. Not implemented.

Index