Package ‘dfrr’

July 20, 2020

Type Package

Title Discritizded Functional Response Regression

Version 0.5.1

Maintainer Fatemeh Asgari <ft.asgari@sci.ui.ac.ir>

Description Dichotomized functional response regression model

License GPL-3

LazyData TRUE

Encoding UTF-8

Depends R, tmvtnorm, fda

Imports stats, MASS

Suggests ggplot2, plotly

VignetteBuilder knitr

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

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

RoxygenNote 7.1.1

NeedsCompilation no

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

Date/Publication 2020-07-20 15:48:56 UTC

R topics documented:

`dfrr-package` *dfrr: Discritizded Functional Response Regression*

Description

Dichotomized functional response regression model

Details

The only function you're likely to need from dfrr-package is [dfrr()].

Author(s)

Maintainer: Fatemeh Asgari <ft.asgari@sci.ui.ac.ir>

Authors:

- Saeed Hayati <s.hayati@sci.ui.ac.ir> [contributor]

References

Fatemeh Asgari, Alamatsaz Mohammad Hossein, Hayati Saeed (2021). Dichotomized Functional Response Regression Model. <http://arxive.org/adress_to_paper>

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)
Y<-simulate.simple.dfrr(beta0=function(t){cos(pi*t+pi)},
                        beta1=function(t){2*t},
                        X=X,M=M)

dfrr_fit<-dfrr(Y~X,yind=time)

coefs<-coef(dfrr_fit)
  plot(coefs)

fitteds<-fitted(dfrr_fit)
  plot(fitteds)
```

```
resids<-residuals(dfrr_fit)

eig<-eigen(dfrr_fit)
plot(eig,plot.contour=TRUE,plot.3d.surface = TRUE)

newdata<-data.frame(X=c(1,0))
  preds<-predict(dfrr_fit,newdata=newdata)
  plot(preds,conf.level = 0.9)

newdata<-data.frame(X=c(1,0))
  preds<-predict(dfrr_fit,newdata=newdata)
  plot(preds,conf.level = 0.9)

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(madras_dfrr,newdata=newdata,newydata = newydata)
plot(preds,conf.level = 0.9)
```

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

Description

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

Usage

```
basis(dfrr_fit)
```

Arguments

`dfrr_fit`	a fitted `dfrr`-object obtained from invoking the function `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)
dfrr_fit<-dfrr(Y~X,yind=time)
coefs<-coef(dfrr_fit,return.fourier.coefs=TRUE)

basis<-basis(dfrr_fit)
evaluated_coefs<-coefs\

#Plotting the regression coefficients
par(mfrow=c(1,2))
plot(time,evaluated_coefs[1,],l,main="Intercept")
plot(time,evaluated_coefs[2,],l,main="X")
```

`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

```
coef.dfrr(
  dfrr_fit,
  standardized = NULL,
  unstandardized = !standardized,
  return.fourier.coefs = NULL,
  return.evaluations = !return.fourier.coefs,
  time_to_evaluate = NULL
)
```

Arguments

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

`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 fitted latent functions, for the case of `return.evaluations=TRUE`.

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.

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)
dfrr_fit<-dfrr(Y~X,yind=time)
coefs<-coef(dfrr_fit)
plot(coefs)
```

---

dfrr *Dichotomized Functional Response Regression*

Description

Implementing Function-on-Scalar Regression model, in which the response function is dichotomized and observed sparsely.

Usage

```
dfrr(
  formula,
  yind = NULL,
  data = NULL,
  ydata = NULL,
  method = c("REML", "ML"),
  rangeval = NULL,
  basis = 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.
...	other arguments that can be passed to the inner function AMCEM.

Details

The output is a dfrr-object, which then can be injected into other methods/functions to post-process the fitted model, including: coefs.dfrr,fitted.dfrr, residuals.dfrr, predict.dfrr, eigen.dfrr, summary.dfrr, qq.dfrr, model.matrix.dfrr, plot.coefs.dfrr, plot.fitted.dfrr, plot.residuals.dfrr, plot.predict.dfrr, plot.eigen.dfrr, plot.residuals.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)
dfrr_fit<-dfrr(Y~X,yind=time)
summary(dfrr_fit)

##### Fitting dfrr model to the Madras Longitudinal Schizophrenia data
data(madras)
ydata<-data.frame(.obs=madras$id,.index=madras$month,.value=madras$y)
ids<-unique(madras$id)
q<-4
N<-length(ids)
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

madras_dfrr<-dfrr(Y~Age+Gender+Age*Gender, data=xData, ydata=ydata, J=11,T_E=5)
coefs<-coef(madras_dfrr)
plot(coefs)

fpcs<-fpca(madras_dfrr)
plot(fpcs)
```

---

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(
  dfrr_fit,
  return.fourier.coefs = NULL,
  return.evaluations = !return.fourier.coefs,
  time_to_evaluate = NULL,
  standardized = NULL,
  unstandardized = !standardized
)
```

Arguments

`dfrr_fit` 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`.

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.

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)
dfrr_fit<-dfrr(Y~X,yind=time)
fitteds<-fitted(dfrr_fit)
plot(fitteds)
```

fpca *Functional principal component analysis 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(dfrr_fit, standardized = NULL, unstandardized = !standardized)
```

Arguments

`dfrr_fit` 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)
dfrr_fit<-dfrr(Y~X,yind=time)
fpcs<-fpca(dfrr_fit)
plot(fpcs)
```

`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/data

References

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

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(coefs, select = NULL, ...)
```

Arguments

`coefs`	a `coef.dfrr`-object.
`select`	a vector of length one or more of indices of regression coefficients to plot.
`...`	graphical parameters passed to `plot`.

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)
dfrr_fit<-dfrr(Y~X,yind=time)
coefs<-coef(dfrr_fit)
plot(coefs)
```

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

Description

Plot a `fitted.dfrr` object.

Usage

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

```
  col.circle = "black",
  ...
)
```

Arguments

fitted.dfrr — 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.

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.

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)
dfrr_fit<-dfrr(Y~X,yind=time)
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(
  dfrr_fit,
  plot.eigen.functions = TRUE,
  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)
)
```

Arguments

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

`plot.eigen.functions`
a `boolean` indicating whether to print the principal components/eigen-functions. Defaults to TRUE.

`plot.contour` a `boolean` indicating whether to print the contour plot of the kernel function. It requires the package `ggplot2` 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`.

Details

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

If the package `ggplot2` 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`.

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)
dfrr_fit<-dfrr(Y~X,yind=time)
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(
  predict.dfrr,
  id = NULL,
  conf.band.type = "BEc",
  conf.level = 0.95,
  main = id,
  col = "blue",
  lwd = 2,
  lty = "solid",
  cex.circle = 1,
  col.circle = "black",
  ylim = NULL,
  ...
)
```

Arguments

`predict.dfrr`	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.
`conf.band.type`	a type of confidence band specified in package `fregion`. Can be either NULL for omitting the confidence band from the plot, "BEc" for modified Scheffe style band constructing from a hyper-ellipsoid region, "Bs" for Parametric bootstrap simultaneous confidence band, or any other `conf.band.type` acceptable to package `fregion`. Defaults to NULL. See References.
`conf.level`	confidence levels for the bands to achieve. Defaults to 0.95.
`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.

Details

The output is the plot of predictions of latent functions given the new covariates. For the case in which `newdata` 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. Confidence bands can also be added to the plot if the package `fregion` is installed.

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)
dfrr_fit<-dfrr(Y~X,yind=time)
preds<-predict(dfrr_fit)
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(residuals.dfrr, ...)

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

Arguments

`residuals.dfrr`	a `residuals.dfrr`-object.
`...`	graphical parameters passed to `car::qqPlot`
`dfrr_fit`	a fitted `dfrr`-object obtained from invoking the function `dfrr`.

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)
dfrr_fit<-dfrr(Y~X,yind=time)
resid<-residuals(dfrr_fit)
plot(resid)
#qq(dfrr_fit)
```

---

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(
  dfrr_fit,
  newdata,
  newydata = NULL,
  standardized = NULL,
  unstandardized = !standardized,
  return.fourier.coefs = NULL,
  return.evaluations = !return.fourier.coefs,
  time_to_evaluate = NULL
)
```

Arguments

- `dfrr_fit` — 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.

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))).

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)
dfrr_fit<-dfrr(Y~X,yind=time)
preds<-predict(dfrr_fit)
plot(preds)
```

`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(dfrr_fit, standardized = NULL, unstandardized = !standardized)
```

Arguments

`dfrr_fit` 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`.

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)
dfrr_fit<-dfrr(Y~X,yind=time)
resid<-residuals(dfrr_fit)
plot(resid)
#qq(dfrr_fit)
```

`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, and $\epsilon_i(t)$ is iid standard normal for each i and t. For demonstration purpose only.

Usage

```
## S3 method for class simple.dfrr
simulate(
  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 regression coefficients
`X`	an (optional) vector consists of scalar covariate
`time`	an (optional) vector of time point for which, each sample curve is observed at.

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)
```

Index