Package ‘FREG’

May 9, 2022

Title Functional Regression Models
Version 1.1
License GPL-3
Encoding UTF-8
LazyData True
LazyDataCompression xz
RoxygenNote 7.1.1
Suggests knitr, rmarkdown
Imports fda, graphics
Depends R (>= 2.10)
NeedsCompilation no
Author Sanja Samardzic [aut],
    Julien Jacques [cre, ctb]
Maintainer Julien Jacques <julien.jacques@univ-lyon2.fr>
Repository CRAN
Date/Publication 2022-05-09 11:50:02 UTC

R topics documented:

CanadianWeather ................................. 2
cycling .............................................. 3
freg ............................................... 3
gradient .......................................... 5
Hessian ............................................ 5
lfreg .............................................. 6
loglik ............................................. 7
CanadianWeather

A dataset containing information about weather in Canadian weather stations

Usage

CanadianWeather

Format

A list that contains 8 variables:

- **dailyAvg** average daily temperature and precipitation and log of precipitation for 35 Canadian weather stations
- **place** places where Canadian weather stations are located
- **province** provinces where Canadian weather stations are located
- **coordinates** geographical coordinates
- **region** regions where Canadian weather stations are located
- **monthlyTemp** monthly temperature data
- **monthlyPrecip** monthly precipitation data
- **geogindex** geographic index
**cycling**

**Cycling dataset**

**Description**

A dataset containing information about cycling sessions

**Usage**

cycling

**Format**

A list of 9 elements. The first element of the list is time whereas the other 8 elements are functional variables. The performance of 216 cyclists is recorded during one hour. Thus, all functional variables are expressed in a form of 216 x 3600 matrix

- SECS 3600 seconds (1 hour)
- KM kilometers
- WATTS power
- CAD cadence
- KPH kilometers per hour
- HR heart rate
- ALT altitude
- SLOPE slope
- TEMP temperature

**freg**

**Functional linear regression model**

**Description**

Functional linear regression model in which the response variable is a scalar variable whereas the independent variables are functional variables. Independent variables could also be scalar variables.

**Usage**

freg(formula, betalist = NULL)
Arguments

- **formula**: a formula expression of the form response ~ predictors. On the left side of the formula, y is a numeric variable whereas on the right side, X can be either functional data object of class `fd` or a scalar variable of class `numeric`. The length of a scalar variable must equal the length of a response variable. Similarly, the number of observations of a functional covariate must equal the length of a response variable.

- **betalist**: an optional argument. A list which contains beta regression coefficient functions for independent variables. If betalist is not provided, the number of estimated beta regression coefficient functions for one functional covariate would equal the number of basis functions used to represent that functional covariate. For a scalar variable, beta regression coefficient function is also a functional object whose basis is constant. Needless to say, for a scalar variable, there will be one beta regression coefficient.

Value

- **call**: call of the `freg` function
- **x.count**: number of predictors
- **xfdlist**: a list of functional data objects. The length of the list is equal to the number of predictors
- **betalist**: a list of beta regression coefficient functions
- **coefficients**: estimated beta regression coefficient functions

Examples

```r
library(fda)
y = log10(apply(daily$precav,2,sum))
x = daily$tempav
xbasis = create.fourier.basis(c(1,365),5) # 5 basis functions
# smoothing of the data and extraction of functional data object
xfd=smooth.basis(c(1:365),x,xbasis)$fd
formula = y ~ xfd
# betalist is an optional argument
bbasis = create.fourier.basis(c(1,365),5) # 5 basis functions
betalist = list(bbasis)
freg.model = freg(formula = formula, betalist = betalist)
# Functional variable and two scalar variables
latitude = CanadianWeather$coordinates[,1]
longitude = CanadianWeather$coordinates[,2]
xfdlist = list(xfd, latitude, longitude)
cbasis = create.constant.basis(c(1,365))
#betalist = list(bbasis, cbasis, cbasis)
formula = y ~ xfd + latitude + longitude
freg.model = freg(formula = formula, betalist = betalist)
print(freg.model$coefficients)
```
gradient

---

Estimation of gradient of log-likelihood function

**Description**

First derivative of log-likelihood function

**Usage**

\texttt{gradient(x, y, beta)}

**Arguments**

- **x**: a design matrix which is a product of inner product of basis functions and basis coefficients of functional covariate \(X\)
- **y**: a response variable of class \texttt{factor}
- **beta**: initial values for beta regression coefficients and intercepts

**Value**

- \texttt{grd}: a vector of gradient values at the estimated optimum

---

Hessian

---

Estimation of Hessian matrix

**Description**

Second derivative of log-likelihood function

**Usage**

\texttt{Hessian(x, y, beta)}

**Arguments**

- **x**: a design matrix which is a product of inner product of basis functions and basis coefficients of functional covariate \(X\)
- **y**: a response variable of class \texttt{factor}
- **beta**: initial values for beta regression coefficients and intercepts

**Value**

- \texttt{Hess}: a Hessian matrix at the estimated optimum
### Description

Functional logistic regression model in which the response variable is a factor variable whereas the independent variables are functional variables. Independent variables could also be scalar variables.

### Usage

`lfreg(formula, betalist = NULL)`

### Arguments

- **formula**: a formula expression of the form `response ~ predictors`. On the left side of the formula, `y` is a factor variable whereas on the right side, `X` can be either functional data object of class `fd` or a scalar variable of class `numeric`. The length of a scalar variable must equal the length of a response variable. Similarly, the number of observations of a functional covariate must equal the length of a response variable.

- **betalist**: an optional argument. A list which contains beta regression coefficient functions for independent variables. If betalist is not provided, the number of estimated beta regression coefficient functions for one functional covariate would equal the number of basis functions used to represent that functional covariate. For a scalar variable, beta regression coefficient function is also a functional object whose basis is constant. Needless to say, for a scalar variable, there will be one beta regression coefficient.

### Value

- **call**: call of the `lfreg` function
- **x.count**: number of predictors
- **xfdlist**: a list of functional data objects. The length of the list is equal to the number of predictors
- **betalist**: a list of beta regression coefficient functions
- **coefficients**: estimated beta regression coefficient functions
- **fitted.values**: predicted values of a response variable `y`
- **loglik**: a value of log-likelihood function at optimum
- **df**: degrees of freedom
- **AIC**: Akaike information criterion
- **iteration**: number of iterations needed for convergence criterion to be met
loglik

Description

Log-likelihood function

Usage

loglik(x, y, beta)

Arguments

x a design matrix which is a product of inner product of basis functions and basis coefficients of functional covariate X
y a response variable of class factor
beta initial values for beta regression coefficients and intercepts

Value

11 a value of the log-likelihood function at the estimated optimum
Functional ordinal logistic regression model

Description

Functional ordinal logistic regression model in which the response variable is a factor variable whereas the independent variables are functional variables. Independent variables could also be scalar variables.

Usage

olfreg(formula, betalist = NULL)

Arguments

formula a formula expression of the form response ~ predictors. On the left side of the formula, y is a factor variable whereas on the right side, X can be either functional data object of class fd or a scalar variable of class numeric. The length of a scalar variable must equal the length of a response variable. Similarly, the number of observations of a functional covariate must equal the length of a response variable.

betalist an optional argument. A list which contains beta regression coefficient functions for independent variables. If betalist is not provided, the number of estimated beta regression coefficient functions for one functional covariate would equal the number of basis functions used to represent that functional covariate. For a scalar variable, beta regression coefficient function is also a functional object whose basis is constant. Needless to say, for a scalar variable, there will be one beta regression coefficient.

Value

call call of the olfreg function
x.count number of predictors
xfdlist a list of functional data objects. The length of the list is equal to the number of predictors
betalist a list of beta regression coefficient functions
coefficients estimated beta regression coefficient functions
alpha estimated intercepts which represent boudaries of categories of dependent factor variable y
ylev a number of categories of a response variable
fitted.values fitted probabilities of a dependent factor variable y
loglik a value of log-likelihood function at optimum
grd a vector of gradient values at optimum
Hess Hessian matrix at optimum
**optimization**

<table>
<thead>
<tr>
<th>df</th>
<th>degrees of freedom</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>Akaike information criterion</td>
</tr>
<tr>
<td>iteration</td>
<td>number of iterations of Fisher Scoring algorithm needed for convergence</td>
</tr>
</tbody>
</table>

**Examples**

```r
# cycling dataset
library(fda)
# creation of ordinal variable from HR variable
zoneHR = rep(0, 216)
zoneHR[which(rowMeans(cycling$HR[,1:60])<107)] = 1
zoneHR[which((rowMeans(cycling$HR[,1:60])<125)&(rowMeans(cycling$HR[,1:60])>107))] = 2
zoneHR[which((rowMeans(cycling$HR[,1:60])<142)&(rowMeans(cycling$HR[,1:60])>125))] = 3
zoneHR[which((rowMeans(cycling$HR[,1:60])<160)&(rowMeans(cycling$HR[,1:60])>142))] = 4
zoneHR[which((rowMeans(cycling$HR[,1:60])>160))] = 5

# first functional variable - power (WATTS)
watts = t(cycling$WATTS[,1:60])
# set up a fourier basis system due to its cycling pattern
xbasis = create.fourier.basis(c(1,60),5) # 5 basis functions for example
watts.fd = smooth.basis(c(1:60),watts,xbasis)$fd
zoneHR = as.factor(zoneHR)
formula = zoneHR ~ watts.fd
olfreg.model = olfreg(formula = formula)

# additional functional variable - cadence (CAD)
cad = t(cycling$CAD[,1:60])
# set up a functional variable for cad
xbasis2 = create.bspline.basis(c(1,60), nbasis = 5, norder = 4)
cad.fd = smooth.basis(c(1:60),cad,xbasis2)$fd
formula = zoneHR ~ watts.fd + cad.fd
olfreg.model = olfreg(formula = formula)
```

---

**optimization**  

*Fisher Scoring algorithm*

**Description**

Optimization algorithm for the estimation of beta regression coefficient functions and intercepts

**Usage**

```r
optimization(x, y, beta, loglik, gradient, Hessian)
```

**Arguments**

- **x**: a design matrix which is a product of inner product of basis functions and basis coefficients of functional covariate $X$
- **y**: a response variable of class factor
- **beta**: initial values for beta regression coefficients and intercepts
loglik log-likelihood function
gradient function for the estimation of first derivative of log-likelihood function - gradient
Hessian function for the estimation of second derivative of log-likelihood function - Hessian

Value
beta a vector with estimated beta regression coefficients and intercepts
ll a value of the log-likelihood function at the estimated optimum
grd a vector of gradient values at the estimated optimum
hessian Hessian matrix at the estimated optimum

plot_freg Plot coefficients from FREG model

Description
Plot coefficients from FREG model

Usage
plot_freg(object)

Arguments
object FREG model

Value
plot of the beta coefficient regression functions for each variable

Examples
library(fda)
y = log10(apply(CanadianWeather$dailyAv[1:335,,2],2,sum))
x = CanadianWeather$dailyAv[1:335,,1] # temperature
xbasis = create.fourier.basis(c(1,335),5)
xfd = smooth.basis(c(1:335),x,xbasis)$fd
bbasis = create.fourier.basis(c(1,335),5)
betalist = list(bbasis)
formula = y ~ xfd
freg.model = freg(formula = formula, betalist = betalist)
plot_freg(freg.model)
plot_lfreg

Description
Plot coefficients from LFREG model

Usage
plot_lfreg(object)

Arguments
object
LFREG model

Value
plot of the beta coefficient regression functions for each variable

Examples
library(fda)
precipitation_data = CanadianWeather$daily[1:334, "Precipitation.mm"]
annualprec = apply(precipitation_data, 2, sum) # without December
y = ifelse(annualprec < mean(annualprec), 0, 1)
y = as.factor(y)
x = CanadianWeather$daily[1:334, "Temperature.C"]
xbasis = create.fourier.basis(c(1, 334), 5) # 5 basis functions
xfd = smooth.basis(c(1:334), x, xbasis)$fd
bbasis = create.fourier.basis(c(0, 334), 5)
betalist = list(bbasis)
formula = y ~ xfd
lfreg.model = lfreg(formula, betalist = betalist)
plot_lfreg(lfreg.model)

plot_olfreg

Description
Plot coefficients from OLFREG model

Usage
plot_olfreg(object)
Arguments

object OLFREG model

Value

plot of the beta coefficient regression functions for each intercept and for each variable

Examples

```r
# cycling dataset
library(fda)
# creation of ordinal variable from HR variable
zoneHR=rep(0,216)
zoneHR[which(rowMeans(cycling$HR[,1:1700])<107)]=1
zoneHR[which((rowMeans(cycling$HR[,1:1700])<125)&(rowMeans(cycling$HR[,1:1700])>107))]=2
zoneHR[which((rowMeans(cycling$HR[,1:1700])<142)&(rowMeans(cycling$HR[,1:1700])>125))]=3
zoneHR[which((rowMeans(cycling$HR[,1:1700])<160)&(rowMeans(cycling$HR[,1:1700])>142))]=4
zoneHR[which((rowMeans(cycling$HR[,1:1700])>160))]=5
# first functional variable - power (WATTS)
watts = t(cycling$WATTS[,1:1700])
# set up a fourier basis system due to its cycling pattern
xbasis = create.fourier.basis(c(1,1700),50) # 50 basis functions for example
watts.fd = smooth.basis(c(1:1700),watts,xbasis)$fd
zoneHR = as.factor(zoneHR)
# additional functional variable - cadence (CAD)
cad = t(cycling$CAD[,1:1700])
# set up a functional variable for cad
xbasis2 = create.bspline.basis(c(1,1700), nbasis = 25, norder = 4)
cad.fd = smooth.basis(c(1:1700),cad,xbasis2)$fd
formula = zoneHR ~ watts.fd + cad.fd
olfreg.model = olfreg(formula = formula)
plot_olfreg(olfreg.model)
```

```
predict.freg

Predict FREG model

Description

Prediction of FREG model

Usage

## S3 method for class 'freg'
predict(object, ..., newdata = NULL)
```
**Arguments**

- **object**: FREG model for which predictions are computed
- **...**: additional arguments relevant for the generic method
- **newdata**: an optional argument. Newdata should be organized as a list. The elements of the list are covariates from FREG model, respectively. No data transformation is needed. Thus, functional covariates are entered in the list newdata in their raw form. The `predict.freg` function will take care of the transformation of such covariates into the functional form of their equivalents from FREG model.

**Value**

predictions of dependent variable \( y \)

**Examples**

```r
library(fda)
y = log10(apply(CanadianWeather$dailyAv[1:334,,2],2,sum))
x = CanadianWeather$dailyAv[1:334,,1]  # temperature
xbasis = create.fourier.basis(c(1,334),5)
xfd = smooth.basis(c(1:335),x,xbasis)$fd
bbasis = create.fourier.basis(c(1,334),5)
latitude = CanadianWeather$coordinates[,1]
longitude = CanadianWeather$coordinates[,2]
xfdlist = list(xfd, latitude, longitude)
bbasis = create.constant.basis(c(1,334))
betalist = list(bbasis, cbasis, cbasis)
formula = y ~ xfd + latitude + longitude
freg.model = freg(formula = formula, betalist = betalist)
# Prediction with new data included
newdata = list(CanadianWeather$dailyAv[1:365,,1], latitude, longitude)
# newdata = list(xfd_1, latitude, longitude)  # funct. and scalar variable(s)
yhat = predict(freg.model, newdata = newdata)
```

---

**predict.lfreg**

*Predict LFREG model*

**Description**

Prediction of LFREG model

**Usage**

```r
## S3 method for class 'lfreg'
predict(object, ..., newdata = NULL, type = c("probabilities", "labels"))
```
predict.olfreg

Description

Prediction of OLFREG model

Usage

## S3 method for class 'olfreg'
predict(object, ..., newdata = NULL, type = c("probabilities", "labels"))
Arguments

- object: OLFREG model for which predictions are computed
- newdata: an optional argument. Newdata should be organized as a list. The elements of the list are covariates from OLFREG model, respectively. No data transformation is needed. Thus, functional covariates are entered in the list newdata in their raw form. The predict.olfreg function will take care of the transformation of such covariates into the functional form of their equivalents from OLFREG model.
- type: c("probabilities", "labels")

Value

predictions of dependent variable y

Examples

```r
# cycling dataset
library(fda)
# creation of ordinal variable from HR variable
zoneHR=rep(0,216)
zoneHR[which(rowMeans(cycling$HR[,1:1700])<107)]=1
zoneHR[which((rowMeans(cycling$HR[,1:1700])<125)&(rowMeans(cycling$HR[,1:1700])>107))]=2
zoneHR[which((rowMeans(cycling$HR[,1:1700])<142)&(rowMeans(cycling$HR[,1:1700])>125))]=3
zoneHR[which((rowMeans(cycling$HR[,1:1700])<160)&(rowMeans(cycling$HR[,1:1700])>142))]=4
zoneHR[which((rowMeans(cycling$HR[,1:1700])>160))]=5
# first functional variable - power (WATTS)
watts = t(cycling$WATTS[,1:1700])
# set up a fourier basis system due to its cycling pattern
xbasis = create.fourier.basis(c(1,1700),50) # 50 basis functions for example
watts.fd = smooth.basis(c(1:1700),watts,xbasis)$fd
zoneHR = as.factor(zoneHR)
# additional functional variable - cadence (CAD)
cad = t(cycling$CAD[,1:1700])
# set up a functional variable for cad
cad.fd = smooth.basis(c(1:1700),cad,xbasis2)$fd
formula = zoneHR ~ watts.fd + cad.fd
olfreg.model = olfreg(formula = formula)

# Predict with new data included
watts_new = t(cycling$WATTS[101:1800])
cad_new = t(cycling$CAD[101:1800])
newdata = list(watts_new, cad_new) # could also be fd var instead of raw data
yhat = predict(olfreg.model, newdata = newdata, type = "labels")
```
Romberg integration is a process of numerical integration. Composite Trapezoidal Rule is used for the approximation of an integral. Then, Richardson extrapolation is used in order to improve previously computed approximations. The range over which the integral is defined is the range in which functional data are defined. When the relative error is infinitesimally small, convergence criterion is fulfilled.

Usage

romberg_alg(xbasis, bbasis)

Arguments

xbasis: basis functional data object used to represent functional covariate X. If covariate is a scalar variable, xbasis is a constant basis functional data object

bbasis: basis functional data object used to represent beta regression coefficient function for each independent variable

Value

S matrix: a matrix of inner product of two basis objects. The dimensions of the matrix are determined by the number of basis functions. The number of rows is equal to the number of basis functions for X, and the number of columns is equal to the number of basis functions for beta regression coefficient functions.

summary.freg produce summary of FREG model fitting function freg.

Usage

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

Arguments

object: FREG model

...: additional arguments relevant for the generic method
Value

call of the function
beta regression coefficient functions

Description

summary.lfreg produce summary of LFREG model fitting function lfreg.

Usage

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

Arguments

object LFREG model
... additional arguments relevant for the generic method

Value

call of the function
beta regression coefficient functions

Description

summary.olfreg produce summary of OLFREG model fitting function olfreg.

Usage

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

Arguments

object OLFREG model
... additional arguments relevant for the generic method

Value

call of the function
alpha regression coefficients and beta regression coefficient functions
Index

* datasets
  CanadianWeather, 2
  cycling, 3
CanadianWeather, 2
  cycling, 3
freg, 3
gradients, 5
Hessian, 5
lfreg, 6
loglik, 7
olfreg, 8
optimization, 9
plot_freg, 10
plot_lfreg, 11
plot_olfreg, 11
predict.freg, 12
predict.lfreg, 13
predict.olfreg, 14
romberg_alg, 16
summary.freg, 16
summary.lfreg, 17
summary.olfreg, 17