Package: robflreg (via r-universe)

August 21, 2024

Type Package

Title Robust Functional Linear Regression

Version 1.2

Date 2024-01-24

Depends R (>= 3.5.0), fda, MASS, robustbase

Imports expm, fda.usc, goffda, mvtnorm, pcaPP, fields

LazyLoad yes

ByteCompile TRUE

Maintainer Ufuk Beyaztas <ufukbeyaztas@gmail.com>

Description Functions for implementing robust methods for functional linear regression. In the functional linear regression, we consider scalar-on-function linear regression and function-on-function linear regression. More details, see Beyaztas, U., and Shang, H. L. (2021) <arXiv:2111.01238> and Beyaztas, U., and Shang, H. L. (2022) <arXiv:2203.05065>.

License GPL-3

NeedsCompilation no

Author Ufuk Beyaztas [aut, cre, cph] (<https://orcid.org/0000-0002-5208-4950>), Han Lin Shang [aut] (<https://orcid.org/0000-0003-1769-6430>)

Date/Publication 2024-01-23 21:30:03 UTC

Repository https://ufukbeyaztas.r-universe.dev

RemoteUrl https://github.com/cran/robflreg

RemoteRef HEAD

RemoteSha f1172bdcfeac71bc3642e00b4033d6e80fa0af20

Contents

`robflreg-package` *Robust function-on-function regression*

Description

This package presents robust methods for analyzing functional linear regression.

Author(s)

Ufuk Beyaztas and Han Lin Shang

Maintainer: Ufuk Beyaztas <ufukbeyaztas@gmail.com>

References

B. Akturk, U. Beyaztas, H. L. Shang, A. Mandal (2024) Robust functional logistic regression, *Advances in Data Analysis and Classification*, in press.

U. Beyaztas, H. L. Shang and A. Mandal (2024) Robust function-on-function interaction regression, *Statistical Modelling: An International Journal*, in press.

U. Beyaztas, M. Tez and H. L. Shang (2024) Robust scalar-on-function partial quantile regression, *Journal of Applied Statistics*, in press.

U. Beyaztas and H. L. Shang (2023) Robust functional linear regression models, *The R Journal*, **15**(1), 212-233.

M. Mutis, U. Beyaztas, G. G. Simsek and H. L. Shang (2023) A robust scalar-on-function logistic regression for classification, *Communications in Statistics - Theory and Methods*, **52**(23), 8538-8554.

S. Saricam, U. Beyaztas, B. Asikgil and H. L. Shang (2022) On partial least-squares estimation in scalar-on-function regression models, *Journal of Chemometrics*, **36**(12), e3452.

U. Beyaztas and H. L. Shang (2022) A comparison of parameter estimation in function-on-function regression, Communications in Statistics - Simulation and Computation, 51(8), 4607-4637.

U. Beyaztas and H. L. Shang (2022) A robust functional partial least squares for scalar-on-multiple-function regression, Journal of Chemometrics, 36(4), e3394.

U. Beyaztas, H. L. Shang and A. Alin (2022) Function-on-function partial quantile regression, Journal of Agricultural, Biological, and Environmental Statistics, 27(1), 149-174.

U. Beyaztas and H. L. Shang (2021) A partial least squares approach for function-on-function interaction regression, Computational Statistics, 36(2), 911-939.

U. Beyaztas and H. L. Shang (2021) A robust partial least squares approach for function-on-function regression, Brazilian Journal of Probability and Statistics, 36(2), 199-219.

U. Beyaztas and H. L. Shang (2021) Function-on-function linear quantile regression, Mathematical Modelling and Analysis, 27(2), 322-341.

U. Beyaztas and H. L. Shang (2020) On function-on-function regression: partial least squares approach, Environmental and Ecological Statistics, 27(1), 95-114.

U. Beyaztas and H. L. Shang (2019) Forecasting functional time series using weighted likelihood methodology, 89(16), 3046-3060.

`generate.ff.data` *Generate functional data for the function-on-function regression model*

Description

This function provides a unified simulation structure for the function-on-function regression model

$$Y(t) = \sum_{m=1}^{M} \int X_m(s)\beta_m(s,t)ds + \epsilon(t),$$

where $Y(t)$ denotes the functional response, $X_m(s)$ denotes the m-th functional predictor, $\beta_m(s,t)$ denotes the m-th bivariate regression coefficient function, and $\epsilon(t)$ is the error function.

Usage

```
generate.ff.data(n.pred, n.curve, n.gp, out.p = 0)
```

Arguments

`n.pred`	An integer, denoting the number of functional predictors to be generated.
`n.curve`	An integer, specifying the number of observations for each functional variable to be generated.
`n.gp`	An integer, denoting the number of grid points, i.e., a fine grid on the interval [0, 1].
`out.p`	An integer between 0 and 1, denoting the outlier percentage in the generated data.

Details

In the data generation process, first, the functional predictors are simulated based on the following process:

$$X_m(s) = \sum_{j=1}^{5} \kappa_j v_j(s),$$

where κ_j is a vector generated from a Normal distribution with mean one and variance $\sqrt{a}j^{-1/2}$, a is a uniformly generated random number between 1 and 4, and

$$v_j(s) = \sin(j\pi s) - \cos(j\pi s).$$

The bivariate regression coefficient functions are generated from a coefficient space that includes ten different functions such as:

$$b\sin(2\pi s)\sin(\pi t)$$

and

$$be^{-3(s-0.5)^2}e^{-4(t-1)^2},$$

where b is generated from a uniform distribution between 1 and 3. The error function $\epsilon(t)$, on the other hand, is generated from the Ornstein-Uhlenbeck process:

$$\epsilon(t) = l + [\epsilon_0(t) - l]e^{-\theta t} + \sigma \int_0^t e^{-\theta(t-u)} dW_u,$$

where $l, \theta > 0, \sigma > 0$ are constants, $\epsilon_0(t)$ is the initial value of $\epsilon(t)$ taken from W_u, and W_u is the Wiener process. If outliers are allowed in the generated data, i.e., $out.p > 0$, then, the randomly selected $n.curve \times out.p$ of the data are generated in a different way from the aforementioned process. In more detail, if $out.p > 0$, the bivariate regression coefficient functions (possibly different from the previously generated coefficient functions) generated from the coefficient space with b^* (instead of b), where b^* is generated from a uniform distribution between 1 and 2, are used to generate the outlying observations. In addition, in this case, the following process is used to generate functional predictors:

$$X_m^*(s) = \sum_{j=1}^{5} \kappa_j^* v_j^*(s),$$

where κ_j^* is a vector generated from a Normal distribution with mean one and variance $\sqrt{a}j^{-3/2}$ and

$$v_j^*(s) = 2\sin(j\pi s) - \cos(j\pi s).$$

All the functions are generated equally spaced point in the interval $[0, 1]$.

Value

A list object with the following components:

`Y`	An $n.curve \times n.gp$-dimensional matrix containing the observations of simulated functional response variable.
`X`	A list with length n.pred. The elements are the $n.curve \times n.gp$-dimensional matrices containing the observations of simulated functional predictor variables.
`f.coef`	A list with length n.pred. Each element is a matrix and contains the generated bivariate regression coefficient function.
`out.indx`	A vector with length $n.curve \times out.p$ denoting the indices of outlying observations.

Author(s)

Ufuk Beyaztas and Han Lin Shang

References

E. Garcia-Portugues and J. Alvarez-Liebana J and G. Alvarez-Perez G and W. Gonzalez-Manteiga W (2021) "A goodness-of-fit test for the functional linear model with functional response", *Scandinavian Journal of Statistics*, **48**(2), 502-528.

Examples

```
library(fda)
library(fda.usc)
set.seed(2022)
sim.data <- generate.ff.data(n.pred = 5, n.curve = 200, n.gp = 101, out.p = 0.1)
Y <- sim.data$Y
X <- sim.data$X
coeffs <- sim.data$f.coef
out.indx <- sim.data$out.indx
fY <- fdata(Y, argvals = seq(0, 1, length.out = 101))
plot(fY[-out.indx,], lty = 1, ylab = "", xlab = "Grid point",
     main = "Response", mgp = c(2, 0.5, 0), ylim = range(fY))
lines(fY[out.indx,], lty = 1, col = "black") # Outlying functions
```

---

generate.sf.data *Generate functional data for the scalar-on-function regression model*

Description

This function is used to simulate data for the scalar-on-function regression model

$$Y = \sum_{m=1}^{M} \int X_m(s)\beta_m(s)ds + \epsilon,$$

where Y denotes the scalar response, $X_m(s)$ denotes the m-th functional predictor, $\beta_m(s)$ denotes the m-th regression coefficient function, and ϵ is the error process.

Usage

```
generate.sf.data(n, n.pred, n.gp, out.p = 0)
```

Arguments

`n`	An integer, specifying the number of observations for each variable to be generated.
`n.pred`	An integer, denoting the number of functional predictors to be generated.
`n.gp`	An integer, denoting the number of grid points, i.e., a fine grid on the interval [0, 1].
`out.p`	An integer between 0 and 1, denoting the outlier percentage in the generated data.

Details

In the data generation process, first, the functional predictors are simulated based on the following process:

$$X_m(s) = \sum_{j=1}^{5} \kappa_j v_j(s),$$

where κ_j is a vector generated from a Normal distribution with mean one and variance $\sqrt{a}j^{-3/2}$, a is a uniformly generated random number between 1 and 4, and

$$v_j(s) = \sin(j\pi s) - \cos(j\pi s).$$

The regression coefficient functions are generated from a coefficient space that includes ten different functions such as:

$$b\sin(2\pi s)$$

and

$$b\cos(2\pi s),$$

where b is generated from a uniform distribution between 1 and 3. The error process is generated from the standard normal distribution. If outliers are allowed in the generated data, i.e., $out.p > 0$, then, the randomply selected $n \times out.p$ of the data are generated in a different way from the aforementioned process. In more detail, if $out.p > 0$, the regression coefficient functions (possibly different from the previously generated coefficient functions) generated from the coefficient space with b^* (instead of b), where b^* is generated from a uniform distribution between 3 and 5, are used to generate the outlying observations. In addition, in this case, the following process is used to generate functional predictors:

$$X_m^*(s) = \sum_{j=1}^{5} \kappa_j^* v_j^*(s),$$

where κ_j^* is a vector generated from a Normal distribution with mean one and variance $\sqrt{a}j^{-1/2}$ and

$$v_j^*(s) = 2\sin(j\pi s) - \cos(j\pi s).$$

Moreover, the error process is generated from a normal distribution with mean 1 and variance 1. All the functional predictors are generated equally spaced point in the interval $[0, 1]$.

Value

A list object with the following components:

`Y`	An $n \times 1$-dimensional matrix containing the observations of simulated scalar response variable.
`X`	A list with length n.pred. The elements are the $n \times n.gp$-dimensional matrices containing the observations of simulated functional predictor variables.
`f.coef`	A list with length n.pred. Each element is a vector and contains the generated regression coefficient function.
`out.indx`	A vector with length $n \times out.p$ denoting the indices of outlying observations.

Author(s)

Ufuk Beyaztas and Han Lin Shang

Examples

```
library(fda.usc)
library(fda)
set.seed(2022)
sim.data <- generate.sf.data(n = 400, n.pred = 5, n.gp = 101, out.p = 0.1)
Y <- sim.data$Y
X <- sim.data$X
coeffs <- sim.data$f.coef
out.indx <- sim.data$out.indx
plot(Y[-out.indx,], type = "p", pch = 16, xlab = "Index", ylab = "",
main = "Response", ylim = range(Y))
points(out.indx, Y[out.indx,], type = "p", pch = 16, col = "blue") # Outliers
fX1 <- fdata(X[[1]], argvals = seq(0, 1, length.out = 101))
plot(fX1[-out.indx,], lty = 1, ylab = "", xlab = "Grid point",
     main = expression(X[1](s)), mgp = c(2, 0.5, 0), ylim = range(fX1))
lines(fX1[out.indx,], lty = 1, col = "black") # Leverage points
```

`get.ff.coeffs` *Get the estimated bivariate regression coefficient functions for function-on-function regression model*

Description

This function is used to obtain the estimated bivariate regression coefficient functions $\beta_m(s,t)$ for function-on-function regression model (see the description in `rob.ff.reg` based on output object obtained from `rob.ff.reg`).

Usage

```
get.ff.coeffs(object)
```

Arguments

`object`	The output object of `rob.ff.reg`.

Details

In the estimation of bivariate regression coefficient functions, the estimated functional principal components of response $\hat{\Phi}(t)$ and predictor $\hat{\Psi}_m(s)$ variables and the estimated regression parameter function obtained from the regression model between the principal component scores of response and predictor variables $\hat{B}$ are used, i.e., $\hat{\beta}_m(s,t) = \hat{\Psi}_m^{\top}(s)\hat{B}\hat{\Phi}(t)$.

Value

A list object with the following components:

vars	A numeric vector specifying the indices of functional predictors used in the function-on-function regression model rob.ff.reg.
gpY	A vector containing the grid points of the functional response $Y(t)$.
gpX	A list with length M. The m-th element of gpX is a vector containing the grid points of the m-th functional predictor $X_m(s)$.
coefficients	A list with length M. The m-th element of coefficients is a matrix of the estimated values of the coefficient function for the m-th functional predictor $X_m(s)$.

Author(s)

Ufuk Beyaztas and Han Lin Shang

Examples

```
sim.data <- generate.ff.data(n.pred = 5, n.curve = 200, n.gp = 101)
Y <- sim.data$Y
X <- sim.data$X
gpY = seq(0, 1, length.out = 101) # grid points of Y
gpX <- rep(list(seq(0, 1, length.out = 101)), 5) # grid points of Xs
model.fit <- rob.ff.reg(Y, X, model = "full", emodel = "classical",
                        gpY = gpY, gpX = gpX)
coefs <- get.ff.coeffs(model.fit)
```

---

get.sf.coeffs — *Get the estimated regression coefficient functions for scalar-on-function regression model*

Description

This function is used to obtain the estimated regression coefficient functions $\beta_m(s)$ and the estimated regression coefficients γ_r (if $X.scl \neq NULL$) for scalar-on-function regression model (see the description in rob.sf.reg based on output object obtained from rob.sf.reg).

Usage

```
get.sf.coeffs(object)
```

Arguments

object	The output object of rob.sf.reg.

Details

In the estimation of regression coefficient functions, the estimated functional principal components of predictor $\hat{\Psi}_m(s), 1 \leq m \leq M$ variables and the estimated regression parameter function obtained from the regression model of scalar response on the principal component scores of the functional predictor variables $\hat{B}$ are used, i.e., $\hat{\beta}_m(s) = \hat{\Psi}_m^{\top}(s)\hat{B}$.

Value

A list object with the following components:

`gp` A list with length M. The m-th element of gp is a vector containing the grid points of the m-th functional predictor $X_m(s)$.

`coefficients` A list with length M. The m-th element of coefficients is a vector of the estimated values of the coefficient function for the m-th functional predictor $X_m(s)$.

`scl.coefficients`
A vector consisting of the estimated coefficients of the scalar predictor $X.scl$.

Author(s)

Ufuk Beyaztas and Han Lin Shang

Examples

```
sim.data <- generate.sf.data(n = 400, n.pred = 5, n.gp = 101)
Y <- sim.data$Y
X <- sim.data$X
gp <- rep(list(seq(0, 1, length.out = 101)), 5) # grid points of Xs
model.fit <- rob.sf.reg(Y, X, emodel = "classical", gp = gp)
coefs <- get.sf.coeffs(model.fit)
```

---

getPCA *Functional principal component analysis*

Description

This function is used to perform functional principal component analysis.

Usage

```
getPCA(data, nbasis, ncomp, gp, emodel = c("classical", "robust"))
```

Arguments

data	An $n \times p$-dimensional data matrix for functional data $X(s)$, where n denotes the sample size and p denotes the number of grid points for $X(s)$.
nbasis	An integer specifying the number of B-spline basis expansion functions used to approximate the functional principal components.
ncomp	An integer specifying the number of functional principal components to be computed.
gp	A vector containing the grid points for the functional data for $X(s)$.
emodel	Method to be used for functional principal component decomposition. Possibilities are "classical" and "robust".

Details

The functional principal decomposition of a functional data $X(s)$ is given by

$$X(s) = \bar{X}(s) + \sum_{k=1}^{K} \xi_k \psi_k(s),$$

where $\bar{X}(s)$ is the mean function, $\psi_k(s)$ is the k-th weight function, and ξ_k is the corresponding principal component score which is given by

$$\xi_k = \int (X(s) - \bar{X}(s))\psi_k(s)ds.$$

When computing the estimated functional principal components, first, the functional data is expressed by a set of B-spline basis expansion. Then, the functional principal components are equal to the principal components extracted from the matrix $D\varphi^{1/2}$, where D is the matrix of basis expansion coefficients and φ is the inner product matrix of the basis functions, i.e., $\varphi = \int \varphi(s)\varphi^{\top}(s)ds$. Finally, the k-th weight function is given by $\psi_k(s) = \varphi^{-1/2}a_k$, where a_k is the k-th eigenvector of the sample covariance matrix of $D\varphi^{1/2}$.

If `emodel = "classical"`, then, the standard functional principal component decomposition is used as given by Ramsay and Dalzell (1991).

If `emodel = "robust"`, then, the robust principal component algorithm of Hubert, Rousseeuw and Verboven (2002) is used.

Value

A list object with the following components:

PCAcoef	A functional data object for the eigenfunctions.
PCAscore	A matrix of principal component scores.
meanScore	A functional data object for the mean function.
bs_basis	A functional data object for B-spline basis expansion.
evalbase	A matrix of the B-spline basis expansion functions.

ncomp	An integer denoting the computed number of functional principal components. If the input "ncomp" is NULL, then, the output ncomp equals to the number of functional principal components whose usage results in at least 95% explained variation.
gp	A vector containing the grid points for the functional data for $X(s)$.
emodel	A character vector denoting the method used for functional principal component decomposition.

Author(s)

Ufuk Beyaztas and Han Lin Shang

References

J. O. Ramsay and C. J. Dalzell (1991) "Some tools for functional data analysis (with discussion)", *Journal of the Royal Statistical Society: Series B*, **53**(3), 539-572.

M. Hubert and P. J. Rousseeuw and S. Verboven (2002) "A fast robust method for principal components with applications to chemometrics", *Chemometrics and Intelligent Laboratory Systems*, **60**(1-2), 101-111.

P. Filzmoser and H. Fritz and K Kalcher (2021) pcaPP: Robust PCA by Projection Pursuit, R package version 1.9-74, URL: https://cran.r-project.org/web/packages/pcaPP/index.html.

J. L. Bali and G. Boente and D. E. Tyler and J.-L. Wang (2011) "Robust functional principal components: A projection-pursuit approach", *The Annals of Statistics*, **39**(6), 2852-2882.

Examples

```
sim.data <- generate.ff.data(n.pred = 5, n.curve = 200, n.gp = 101)
Y <- sim.data$Y
gpY <- seq(0, 1, length.out = 101) # grid points
rob.fpca <- getPCA(data = Y, nbasis = 20, ncomp = 4, gp = gpY, emodel = "robust")
```

---

getPCA.test *Get the functional principal component scores for a given test sample*

Description

This function is used to compute the functional principal component scores of a test sample based on outputs obtained from getPCA.

Usage

```
getPCA.test(object, data)
```

Arguments

object	An output object of getPCA.
data	An $n \times p$-dimensional data matrix for functional data $X(s)$ (test sample), where n denotes the sample size and p denotes the number of grid points for $X(s)$.

Details

See getPCA for details.

Value

A matrix of principal component scores for the functional data.

Author(s)

Ufuk Beyaztas and Han Lin Shang

Examples

```
sim.data <- generate.ff.data(n.pred = 5, n.curve = 200, n.gp = 101)
Y <- sim.data$Y
Y.train <- Y[1:100,]
Y.test <- Y[101:200,]
gpY = seq(0, 1, length.out = 101) # grid points
rob.fpca <- getPCA(data = Y.train, nbasis = 20, ncomp = 4,
gp = gpY, emodel = "robust")
rob.fpca.test <- getPCA.test(object = rob.fpca, data = Y.test)
```

MaryRiverFlow *Hourly River Flow Measurements in the Mery River*

Description

Hourly river flow measurements obtained from January 2009 to December 2014 (6 years in total) in the Mery River, Australia.

Usage

```
data(MaryRiverFlow)
```

Author(s)

Ufuk Beyaztas and Han Lin Shang

Examples

```
data(MaryRiverFlow)
# Plot
library(fda.usc)
fflow <- fdata(MaryRiverFlow, argvals = 1:24)
plot(fflow, lty = 1, ylab = "", xlab = "Hour",
main = "", mgp = c(2, 0.5, 0), ylim = range(fflow))
```

`plot_ff_coeffs` *Image plot of bivariate regression coefficient functions of a function-on-function regression model*

Description

This function is used to obtain image plots of bivariate regression coefficient functions of a function-on-function regression model based on output object obtained from `get.ff.coeffs`.

Usage

```
plot_ff_coeffs(object, b)
```

Arguments

`object`	The output object of `get.ff.coeffs`.
`b`	An integer value indicating which regression parameter function to be plotted.

Value

No return value, called for side effects.

Author(s)

Ufuk Beyaztas and Han Lin Shang

References

D. Nychka and R. Furrer and J. Paige and S. Sain (2021) fields: Tools for spatial data. R package version 14.1, URL: https://github.com/dnychka/fieldsRPackage.

Examples

```
sim.data <- generate.ff.data(n.pred = 5, n.curve = 200, n.gp = 101)
Y <- sim.data$Y
X <- sim.data$X
gpY = seq(0, 1, length.out = 101) # grid points of Y
gpX <- rep(list(seq(0, 1, length.out = 101)), 5) # grid points of Xs
model.fit <- rob.ff.reg(Y, X, model = "full", emodel = "classical",
                        gpY = gpY, gpX = gpX)
coefs <- get.ff.coeffs(model.fit)
plot_ff_coeffs(object = coefs, b = 1)
```

plot_sf_coeffs *Plot of regression coefficient functions of a scalar-on-function regression model*

Description

This function is used to obtain the plots of regression coefficient functions of a scalar-on-function regression model based on output object obtained from get.sf.coeffs.

Usage

```
plot_sf_coeffs(object, b)
```

Arguments

object	The output object of get.sf.coeffs.
b	An integer value indicating which regression parameter function to be plotted.

Value

No return value, called for side effects.

Author(s)

Ufuk Beyaztas and Han Lin Shang

Examples

```
sim.data <- generate.sf.data(n = 400, n.pred = 5, n.gp = 101)
Y <- sim.data$Y
X <- sim.data$X
gp <- rep(list(seq(0, 1, length.out = 101)), 5) # grid points of Xs
model.fit <- rob.sf.reg(Y, X, emodel = "classical", gp = gp)
coefs <- get.sf.coeffs(model.fit)
plot_sf_coeffs(object = coefs, b = 1)
```

predict_ff_regression *Prediction for a function-on-function regression model*

Description

This function is used to make prediction for a new set of functional predictors based upon a fitted function-on-function regression model in the output of rob.ff.reg.

Usage

```
predict_ff_regression(object, Xnew)
```

Arguments

object	An output object obtained from rob.ff.reg.
Xnew	A list of matrices consisting of the new observations of functional predictors. The argument Xnew must have the same length and the same structure as the input X of rob.ff.reg.

Value

An $n_{test} \times p$-dimensional matrix of predicted functions of the response variable for the given set of new functional predictors Xnew. Here, n_{test}, the number of rows of the matrix of predicted values, equals to the number of rows of Xnew, and p equals to the number of columns of Y, the input in the rob.ff.reg.

Author(s)

Ufuk Beyaztas and Han Lin Shang

Examples

```
set.seed(2022)
sim.data <- generate.ff.data(n.pred = 5, n.curve = 200, n.gp = 101, out.p = 0.1)
out.indx <- sim.data$out.indx
Y <- sim.data$Y
X <- sim.data$X
indx.test <- sample(c(1:200)[-out.indx], 60)
indx.train <- c(1:200)[-indx.test]
Y.train <- Y[indx.train,]
Y.test <- Y[indx.test,]
X.train <- X.test <- list()
for(i in 1:5){
  X.train[[i]] <- X[[i]][indx.train,]
  X.test[[i]] <- X[[i]][indx.test,]
}
gpY = seq(0, 1, length.out = 101) # grid points of Y
gpX <- rep(list(seq(0, 1, length.out = 101)), 5) # grid points of Xs

model.MM <- rob.ff.reg(Y = Y.train, X = X.train, model = "full", emodel = "robust",
                       fmodel = "MM", gpY = gpY, gpX = gpX)
pred.MM <- predict_ff_regression(object = model.MM, Xnew = X.test)
round(mean((Y.test - pred.MM)^2), 4)        # 0.5925 (MM method)
```

---

predict_sf_regression *Prediction for a scalar-on-function regression model*

Description

This function is used to make prediction for a new set of functional and scalar (if any) predictors based upon a fitted scalar-on-function regression model in the output of rob.sf.reg.

Usage

```
predict_sf_regression(object, Xnew, Xnew.scl = NULL)
```

Arguments

`object`	An output object obtained from `rob.sf.reg`.
`Xnew`	A list of matrices consisting of the new observations of functional predictors. The argument `Xnew` must have the same length and the same structure as the input `X` of `rob.sf.reg`.
`Xnew.scl`	A matrix consisting of the new observations of scalar predictors. The argument `Xnew.scl` must have the same length and the same structure as the input `X.scl` of `rob.sf.reg`.

Value

An $n_{test} \times 1$-dimensional matrix of predicted values of the scalar response variable for the given set of new functional and scalar (if any) predictors `Xnew` and `Xnew.scl`, respectively. Here, n_{test}, the number of rows of the matrix of predicted values, equals to the number of rows of `Xnew` and and `Xnew.scl` (if any).

Author(s)

Ufuk Beyaztas and Han Lin Shang

Examples

```
set.seed(2022)
sim.data <- generate.sf.data(n = 400, n.pred = 5, n.gp = 101, out.p = 0.1)
out.indx <- sim.data$out.indx
indx.test <- sample(c(1:400)[-out.indx], 120)
indx.train <- c(1:400)[-indx.test]
Y <- sim.data$Y
X <- sim.data$X
Y.train <- Y[indx.train,]
Y.test <- Y[indx.test,]
X.train <- X.test <- list()
for(i in 1:5){
  X.train[[i]] <- X[[i]][indx.train,]
  X.test[[i]] <- X[[i]][indx.test,]
}
gp <- rep(list(seq(0, 1, length.out = 101)), 5) # grid points of Xs

model.tau <- rob.sf.reg(Y.train, X.train, emodel = "robust", fmodel = "tau", gp = gp)
pred.tau <- predict_sf_regression(object = model.tau, Xnew = X.test)
round(mean((Y.test - pred.tau)^2), 4)        # 1.868 (tau method)
```

`rob.ff.reg` *Robust function-on-function regression*

Description

This function is used to perform both classical and robust function-on-function regression model

$$Y(t) = \sum_{m=1}^{M} \int X_m(s)\beta_m(s,t)ds + \epsilon(t),$$

where $Y(t)$ denotes the functional response, $X_m(s)$ denotes the m-th functional predictor, $\beta_m(s,t)$ denotes the m-th bivariate regression coefficient function, and $\epsilon(t)$ is the error function.

Usage

```
rob.ff.reg(Y, X, model = c("full", "selected"), emodel = c("classical", "robust"),
 fmodel = c("MCD", "MLTS", "MM", "S", "tau"), nbasisY = NULL, nbasisX = NULL,
 gpY = NULL, gpX = NULL, ncompY = NULL, ncompX = NULL)
```

Arguments

`Y`	An $n \times p$-dimensional matrix containing the observations of functional response $Y(t)$, where n is the sample size and p denotes the number of grid points for $Y(t)$.
`X`	A list consisting of M functional predictors $X_m(s), 1 \leq m \leq M$. Each element of `X` is an $n \times p_m$-dimensional matrix containing the observations of m-th functional predictor $X_m(s)$, where n is the sample size and p_m denotes the number of grid points for $X_m(s)$.
`model`	Model to be fitted. Possibilities are "full" and "selected".
`emodel`	Method to be used for functional principal component decomposition. Possibilities are "classical"" and "robust".
`fmodel`	Fitting model used to estimate the function-on-function regression model. Possibilities are "MCD", "MLTS", "MM", "S", and "tau".
`nbasisY`	An integer value specifying the number of B-spline basis expansion functions to be used to approximate the functional principal components for the response variable $Y(t)$. If `NULL`, then, $min(20, p/4)$ number of B-spline basis expansion functions are used.
`nbasisX`	A vector with length M. Its m-th value denotes the number of B-spline basis expansion functions to be used to approximate the functional principal components for the m-th functional predictor $X_m(s)$. If `NULL`, then, $min(20, p_m/4)$ number of B-spline basis expansion functions are used for each functional predictor, where p_m denotes the number of grid points for $X_m(s)$.
`gpY`	A vector containing the grid points of the functional response $Y(t)$. If `NULL`, then p equally spaced time points in te interval [0, 1] are used.

gpX — A list with length M. The m-th element of gpX is a vector containing the grid points of the m-th functional predictor $X_m(s)$. If NULL, then, p_m equally spaced time points in te interval [0, 1] are used for the m-th functional predictor.

ncompY — An integer specifying the number of functional principal components to be computed for the functional response $Y(t)$. If NULL, then, the number whose usage results in at least 95% explained variation is used as the number of principal components.

ncompX — A vector with length M. Its m-th value denotes the number of functional principal components to be computed for the m-th functional predictor $X_m(s)$. If NULL, then, for each functional predictor, the number whose usage results in at least 95% explained variation is used as the number of principal components.

Details

When performing a function-on-function regression model based on the functional principal component analysis, first, both the functional response $Y(t)$ and functional predictors $X_m(s), 1 \leq m \leq M$ are decomposed by the functional principal component analysis method:

$$Y(t) = \bar{Y}(t) + \sum_{k=1}^{K} \nu_k \phi_k(t),$$

$$X_m(s) = \bar{X}_m(s) + \sum_{l=1}^{K_m} \xi_{ml} \psi_{ml}(s),$$

where $\bar{Y}(t)$ and $\bar{X}_m(s)$ are the mean functions, $\phi_k(t)$ and $\psi_{ml}(s)$ are the weight functions, and $\nu_k = \int (Y(t) - \bar{Y}(t))\phi_k(t)$ and $\xi_{ml} = \int (X_m(s) - \bar{X}_m(s))\psi_{ml}(s)$ are the principal component scores for the functional response and m-th functional predictor, respectively. Assume that the m-th bivariate regression coefficient function admits the expansion

$$\beta_m(s,t) = \sum_{k=1}^{K} \sum_{l=1}^{K_m} b_{mkl} \phi_k(t) \psi_{ml}(s),$$

where $b_{mkl} = \int \int \beta_m(s,t)\phi_k(t)\psi_{ml}(s)dtds$. Then, the following multiple regression model is obtained for the functional response:

$$\hat{Y}(t) = \bar{Y}(s) + \sum_{k=1}^{K} (\sum_{m=1}^{M} \sum_{l=1}^{K_m} b_{mkl} \xi_{ml}) \phi_k(t).$$

If model = "full", then, all the functional predictor variables are used in the model.

If model = "selected", then, only the significant functional predictor variables determined by the forward variable selection procedure of Beyaztas and Shang (2021) are used in the model.

If emodel = "classical", then, the least-squares method is used to estimate the function-on-function regression model.

If emodel = "robust", then, the robust functional principal component analysis of Bali et al. (2011) along with the method specified in fmodel is used to estimate the function-on-function regression model.

If `fmodel = "MCD"`, then, the minimum covariance determinant estimator of Rousseeuw et al. (2004) is used to estimate the function-on-function regression model.

If `fmodel = "MLTS"`, then, the multivariate least trimmed squares estimator Agullo et al. (2008) is used to estimate the function-on-function regression model.

If `fmodel = "MM"`, then, the MM estimator of Kudraszow and Maronna (2011) is used to estimate the function-on-function regression model.

If `fmodel = "S"`, then, the S estimator of Bilodeau and Duchesne (2000) is used to estimate the function-on-function regression model.

If `fmodel = "tau"`, then, the tau estimator of Ben et al. (2006) is used to estimate the function-on-function regression model.

Value

A list object with the following components:

`data`	A list of matrices including the original functional response and functional predictors.
`fitted.values`	An $n \times p$-dimensional matrix containing the fitted values of the functional response.
`residuals`	An $n \times p$-dimensional matrix containing the residual functions.
`fpca.results`	A list object containing the functional principal component analysis results of the functional predictor and functional predictors variables.
`model.details`	A list object containing model details, such as number of basis functions, number of principal components, and grid points used for each functional variable.

Author(s)

Ufuk Beyaztas and Han Lin Shang

References

J. Agullo and C. Croux and S. V. Aelst (2008), "The multivariate least-trimmed squares estimator", *Journal of Multivariate Analysis*, **99**(3), 311-338.

M. G. Ben and E. Martinez and V. J. Yohai (2006), "Robust estimation for the multivariate linear model based on a τ scale", *Journal of Multivariate Analysis*, **97**(7), 1600-1622.

U. Beyaztas and H. L. Shang (2021), "A partial least squares approach for function-on-function interaction regression", *Computational Statistics*, **36**(2), 911-939.

J. L. Bali and G. Boente and D. E. Tyler and J. -L.Wang (2011), "Robust functional principal components: A projection-pursuit approach", *The Annals of Statistics*, **39**(6), 2852-2882.

M. Bilodeau and P. Duchesne (2000), "Robust estimation of the SUR model", *The Canadian Journal of Statistics*, **28**(2), 277-288.

N. L. Kudraszow and R. A. Moronna (2011), "Estimates of MM type for the multivariate linear model", *Journal of Multivariate Analysis*, **102**(9), 1280-1292.

P. J. Rousseeuw and K. V. Driessen and S. V. Aelst and J. Agullo (2004), "Robust multivariate regression", *Technometrics*, **46**(3), 293-305.

Examples

```
sim.data <- generate.ff.data(n.pred = 5, n.curve = 200, n.gp = 101)
Y <- sim.data$Y
X <- sim.data$X
gpY <- seq(0, 1, length.out = 101) # grid points of Y
gpX <- rep(list(seq(0, 1, length.out = 101)), 5) # grid points of Xs
model.MM <- rob.ff.reg(Y = Y, X = X, model = "full", emodel = "robust",
                       fmodel = "MM", gpY = gpY, gpX = gpX)
```

`rob.out.detect` *Outlier detection in the functional response*

Description

This function is used to detect outliers in the functional response based on a fitted function-on-function regression model in the output of `rob.ff.reg`.

Usage

```
rob.out.detect(object, alpha = 0.01, B = 200, fplot = FALSE)
```

Arguments

`object`	An output object obtained from `rob.ff.reg`.
`alpha`	Percentile of the distribution of the functional depth. The default value is 0.01.
`B`	The number of bootstrap samples. The default value is 200.
`fplot`	If `TRUE`, then the outlying points flagged by the method is plotted along with the values of functional response $Y(t)$.

Details

The functional depth-based outlier detection method of Febrero-Bande et al. (2008) together with the h-modal depth proposed by Cuaves et al. (2007) is applied to the estimated residual functions obtained from `rob.ff.reg` to determine the outliers in the response variable. This method makes it possible to determine both magnitude and shape outliers in the response variable Hullait et al., (2021).

Value

A vector containing the indices of outlying observations in the functional response.

Author(s)

Ufuk Beyaztas and Han Lin Shang

References

M. Febrero-Bande and P. Galeano and W. Gonzalez-Mantelga (2008), "Outlier detection in functional data by depth measures, with application to identify abnormal NOx levels", *Environmetrics*, **19**(4), 331-345.

A. Cuaves and M. Febrero and R Fraiman (2007), "Robust estimation and classification for functional data via projection-based depth notions", *Computational Statistics*, **22**(3), 481-496.

H. Hullait and D. S. Leslie and N. G. Pavlidis and S. King (2021), "Robust function-on-function regression", *Technometrics*, **63**(3), 396-409.

Examples

```
sim.data <- generate.ff.data(n.pred = 5, n.curve = 200, n.gp = 101, out.p = 0.1)
out.indx <- sim.data$out.indx
Y <- sim.data$Y
X <- sim.data$X
gpY = seq(0, 1, length.out = 101) # grid points of Y
gpX <- rep(list(seq(0, 1, length.out = 101)), 5) # grid points of Xs

model.MM <- rob.ff.reg(Y = Y, X = X, model = "full", emodel = "robust", fmodel = "MM",
                       gpY = gpY, gpX = gpX)
rob.out.detect(object = model.MM, fplot = TRUE)
sort(out.indx)
```

`rob.sf.reg` *Robust scalar-on-function regression*

Description

This function is used to perform both classical and robust scalar-on-function regression model

$$Y = \sum_{m=1}^{M} \int X_m(s)\beta_m(s)ds + X.scl\gamma + \epsilon,$$

where Y denotes the scalar response, $X_m(s)$ denotes the m-th functional predictor, $\beta_m(s)$ denotes the m-th regression coefficient function, $X.scl$ denotes the matrix of scalar predictors, γ denotes the vector of coefficients for the scalar predictors' matrix, and ϵ is the error function, which is assumed to follow standard normal distribution.

Usage

```
rob.sf.reg(Y, X, X.scl = NULL, emodel = c("classical", "robust"),
fmodel = c("LTS", "MM", "S", "tau"), nbasis = NULL, gp = NULL, ncomp = NULL)
```

Arguments

Y	An $n \times 1$-dimensional matrix containing the observations of scalar response Y, where n denotes the sample size.
X	A list consisting of M functional predictors $X_m(s), 1 \leq m \leq M$. Each element of X is an $n \times p_m$-dimensional matrix containing the observations of m-th functional predictor $X_m(s)$, where n is the sample size and p_m denotes the number of grid points for $X_m(s)$.
X.scl	An $n \times R$-dimensional matrix consisting of scalar predictors $X_r, 1 \leq r \leq R$.
emodel	Method to be used for functional principal component decomposition. Possibilities are "classical"" and "robust".
fmodel	Fitting model used to estimate the function-on-function regression model. Possibilities are "LTS", "MM", "S", and "tau".
nbasis	A vector with length M. Its m-th value denotes the number of B-spline basis expansion functions to be used to approximate the functional principal components for the m-th functional predictor $X_m(s)$. If NULL, then, $min(20, p_m/4)$ number of B-spline basis expansion functions are used for each functional predictor, where p_m denotes the number of grid points for $X_m(s)$.
gp	A list with length M. The m-th element of gp is a vector containing the grid points of the m-th functional predictor $X_m(s)$. If NULL, then, p_m equally spaced time points in the interval [0, 1] are used for the m-th functional predictor.
ncomp	A vector with length M. Its m-th value denotes the number of functional principal components to be computed for the m-th functional predictor $X_m(s)$. If NULL, then, for each functional predictor, the number whose usage results in at least 95% explained variation is used as the number of principal components.

Details

When performing a scalar-on-function regression model based on the functional principal component analysis, first, the functional predictors $X_m(s), 1 \leq m \leq M$ are decomposed by the functional principal component analysis method:

$$X_m(s) = \bar{X}_m(s) + \sum_{l=1}^{K_m} \xi_{ml}\psi_{ml}(s),$$

where $\bar{X}_m(s)$ is the mean function, $\psi_{ml}(s)$ is the weight function, and $\xi_{ml} = \int (X_m(s) - \bar{X}_m(s))\psi_{ml}(s)$ is the principal component score for the m-th functional predictor. Assume that the m-th regression coefficient function admits the expansion

$$\beta_m(s) = \sum_{l=1}^{K_m} b_{ml}\psi_{ml}(s),$$

where $b_{ml} = \int \beta_m(s)\psi_m(s)ds$. Then, the following multiple regression model is obtained for the scalar response:

$$\hat{Y} = \bar{Y} + \sum_{m=1}^{M}\sum_{l=1}^{K_m} b_{ml}\xi_{ml} + X.scl\gamma.$$

If `emodel = "classical"`, then, the least-squares method is used to estimate the scalar-on-function regression model.

If `emodel = "robust"`, then, the robust functional principal component analysis of Bali et al. (2011) along with the method specified in `fmodel` is used to estimate the scalar-on-function regression model.

If `fmodel = "LTS"`, then, the least trimmed squares robust regression of Rousseeuw (1984) is used to estimate the scalar-on-function regression model.

If `fmodel = "MM"`, then, the MM-type regression estimator described in Yohai (1987) and Koller and Stahel (2011) is used to estimate the scalar-on-function regression model.

If `fmodel = "S"`, then, the S estimator is used to estimate the scalar-on-function regression model.

If `fmodel = "tau"`, then, the tau estimator proposed by Salibian-Barrera et al. (2008) is used to estimate the scalar-on-function regression model.

Value

A list object with the following components:

`data`	A list of matrices including the original scalar response and both the scalar and functional predictors.
`fitted.values`	An $n \times 1$-dimensional matrix containing the fitted values of the scalar response.
`residuals`	An $n \times 1$-dimensional matrix containing the residuals.
`fpca.results`	A list object containing the functional principal component analysis results of the functional predictors variables.
`model.details`	A list object containing model details, such as number of basis functions, number of principal components, and grid points used for each functional predictor variable.

Author(s)

Ufuk Beyaztas and Han Lin Shang

References

J. L. Bali and G. Boente and D. E. Tyler and J. -L.Wang (2011), "Robust functional principal components: A projection-pursuit approach", *The Annals of Statistics*, **39**(6), 2852-2882.

P. J. Rousseeuw (1984), "Least median of squares regression", *Journal of the American Statistical Association*, **79**(388), 871-881.

P. J. Rousseeuw and K. van Driessen (1999) "A fast algorithm for the minimum covariance determinant estimator", *Technometrics*, **41**(3), 212-223.

V. J. Yohai (1987), "High breakdown-point and high efficiency estimates for regression", *The Annals of Statistics*, **15**(2), 642-65.

M. Koller and W. A. Stahel (2011), "Sharpening Wald-type inference in robust regression for small samples", *Computational Statistics & Data Analysis*, **55**(8), 2504-2515.

M. Salibian-Barrera and G. Willems and R. Zamar (2008), "The fast-tau estimator for regression", *Journal of Computational and Graphical Statistics*, **17**(3), 659-682

Examples

```
sim.data <- generate.sf.data(n = 400, n.pred = 5, n.gp = 101)
Y <- sim.data$Y
X <- sim.data$X
gp <- rep(list(seq(0, 1, length.out = 101)), 5) # grid points of Xs
model.tau <- rob.sf.reg(Y, X, emodel = "robust", fmodel = "tau", gp = gp)
```

Index