Package 'robflreg'

October 21, 2025

Type Package

Title Robust Functional Linear Regression

Version 1.3

Date 2025-10-22

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

Imports expm, fda.usc, goffda, mvtnorm, pcaPP, fields, methods, Matrix, cvTools, quantreg

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.

License GPL-3

NeedsCompilation no

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

Repository CRAN

Date/Publication 2025-10-21 21:00:01 UTC

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

U. Beyaztas, A. Mandal and H. L. Shang (2026+) Enhancing spatial functional linear regression with robust dimension reduction methods, *Journal of Multivariate Analysis*, **in press**.

U. Beyaztas, H. L. Shang and S. Saricam (2025) Penalized function-on-function linear quantile regression, *Computational Statistics*, **40**, 301-329.

B. Akturk, U. Beyaztas, H. L. Shang, A. Mandal (2025) Robust functional logistic regression, *Advances in Data Analysis and Classification*, **19**, 121-145.

U. Beyaztas, H. L. Shang and A. Mandal (2025) Robust function-on-function interaction regression, *Statistical Modelling: An International Journal*, **25**(3), 195-215.

M. Mutis, U. Beyaztas, F. Karaman and H. L. Shang (2025) On function-on-function linear quantile regression, *Journal of Applied Statistics*, **52**(4), 814-840.

M. Mutis, U. Beyaztas, G. G. Simsek, H. L. Shang and Z. M. Yaseen (2024) Development of functional quantile autoregressive model for river flow curve forecasting, *Earth and Space Science*, **11**, e2024EA003564.

S. Gurer, H. L. Shang, A. Mandal and U. Beyaztas (2024) Locally sparse and robust partial least squares in scalar-on-function regression, *Statistics and Computing*, **34**, article number 150.

U. Beyaztas, M. Tez and H. L. Shang (2024) Robust scalar-on-function partial quantile regression, *Journal of Applied Statistics*, **51**(7), 1359-1377.

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.

U. Beyaztas and H. L. Shang (2022) Robust bootstrap prediction intervals for univariate and multivariate autoregressive time series models, *Journal of Applied Statistics*, **49**(5), 1179-1202.

U. Beyaztas, H. L. Shang and A. G. Abdel-Salam (2022) Functional linear model for interval-valued data, *Communications in Statistics - Simulation and Computation*, **51**(7), 3513-3532.

U. Beyaztas and H. L. Shang (2022) Machine-learning-based functional time series forecasting: Application to age-specific mortality rates, *Forecasting*, **4**, 394-408.

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, H. L. Shang and Z. M. Yaseen (2021) A functional autoregressive model based on exogenous hydrometeorological variables for river flow prediction, *Journal of Hydrology*, **598**, 126380.

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, *Journal of Statistical Computation and Simulation*, **89**(16), 3046-3060.

`fpqr` *Function-on-function partial quantile regression*

Description

This function is used to perform function-on-function linear quantile regression model

$$Q_\tau[Y(u)|X(v)] = \int X(v)\beta_\tau(u,v)dv$$

based on the functional partial quantile regression.

Usage

```
fpqr(y, x, h, tau, nby, nbx, gpy, gpx, qc.type = c("dodge","choi","li"),
hs, nbys, nbxs, nfold, CV)
```

Arguments

y	An matrix containing the observations of function response $Y(u)$.
x	A matrix containing the observations of functional predictor $X(v)$.
h	A numeric value denoting the number of functional partial quantile regression components to be computed.
tau	Quantile level.
nby	A numeric value denoting the number of B-spline basis expansion functions to be used to approximate the functional response variable.
nbx	A numeric value denoting the number of B-spline basis expansion functions to be used to approximate the functional predictor variable.
gpy	A vector containing the grid points of the functional response $Y(u)$.
gpx	A vector containing the grid points of the functional predictor $X(v)$.
qc.type	Method type used to estimate the quantile covariance. Possibilities are "dodge", "choi", and "li".
hs	A vector containing the candidate elements for the h.
nbys	A vector containing the candidate elements for the nby.
nbxs	A vector containing the candidate elements for the nbx.
nfold	An integer denoting the number of folds used in the k-fold cross validation. Default value is 5.
CV	Logical. If TRUE, then nfold cross-validation is used to find optimum values of h, nby, and nbx. If FALSE, then the specified h, nby, and nbx values are used in the model.

Details

If qc.type = "dodge", then, the quantile covariance proposed by Dodge and Whittaker (2009) is used in the functional partial quantile regression decomposition.

If qc.type = "choi", then, the quantile covariance proposed by Choi and Shin (2018) is used in the functional partial quantile regression decomposition.

If qc.type = "li", then, the quantile covariance proposed by Li et al. (2015) is used in the functional partial quantile regression decomposition.

Value

fitted.values	An matrix containing the fitted values of the functional response.
b0.hat	A vector containing the estimate of intersecp parameter.
b.hat	A matrix containing the estimate of bivariate regression coefficient coducted between the functional response and functional predictor.

mdts	A numeric value containing the estimated intercept parameter.
pqr.coefs	A vector containing the estimated regression parameter for the regression problem of scalar response on the partial quantile regression components.
V	A matrix whose rows are the eigenvectors
model.details	A list object containing model details, such as number of basis functions, number of partial quantile components, and grid points used for the functional variables.

Author(s)

Muge Mutis, Ufuk Beyaztas, Filiz Karaman, and Han Lin Shang

References

Y. Dodge and J. Whittaker (2009), "Partial quantile regression", *Metrika*, **70**(1), 35-57. J. E.. Choi and D. W. Shin (2018), "Quantile correlation coefficient: A new tail dependence measure", *Statistical Papers*, **63**, 1075-1104. G. Li and Y. Li and C. L. Tsai (2015), "Quantile correlations and quantile autoregressive modeling" *Journal of American Statistical Association: Theory and Methods*, **110**(509), 246-261.

Examples

```
## Not run:
  gpx <- (1:50)/50
  gpy <- (1:60)/60
  data <- fpqr_dgp(n = 100, gpy = gpy, gpx = gpx, err.dist = "normal")
  y <- data$y
  x <- data$x

  fpqr.model.dodge <- fpqr(y=y, x=x, tau = 0.5, gpx = gpx, gpy = gpy,
                           qc.type = "dodge")

  fpqr.model.choi <- fpqr(y=y, x=x, tau = 0.5, gpx = gpx, gpy = gpy,
                          qc.type = "choi")

  fpqr.model.li <- fpqr(y=y, x=x, tau = 0.5, gpx = gpx, gpy = gpy,
                        qc.type = "li")

## End(Not run)
```

---

fpqr_dgp *Generate a dataset for the function-on-function regression model*

Description

This function can be used to generate a dataset for the function-on-function regression model

$$Y(u) = \int X(v)\beta(u,v)dv + \epsilon(u),$$

where $Y(u)$ denotes the functional response, $X(v)$ denotes the functional predictor, $\beta(u,v)$ denotes the bivariate regression coefficient function, and $\epsilon(u)$ is the error function.

Usage

```
fpqr_dgp(n, nphi, gpy, gpx, err.dist, out.p)
```

Arguments

n	An integer, specifying the number of observations for each variable to be generated.
nphi	An integer, specifying the number of sub-random variables used in the generation of functional predictor. Default value is 10.
gpy	A vector, denoting the grid points of the functional response variable $Y(u)$.
gpx	A vector, denoting the grid points of the functional predictor variable $X(v)$.
err.dist	Distribution of the error term. Possibilities are "normal","t","chisq", "cn".
out.p	An integer between 0 and 1, denoting the outlier percentage in the generated data when the error distribution is contaminated normel, that is, "cn".

Details

When generating the data for the function-on-function regression model, first, the functional predictor is generated using the following process:

$$X_i(v) = \sum_{k=1}^{10} \frac{1}{k^2}\{\zeta_{i1,k}\sqrt{2}\sin(k\pi v)\zeta_{i2,k}\sqrt{2}\cos(k\pi v)\}$$

where $\zeta_{i1,k}$ and $\zeta_{i2,k}$ for $k = 1,\ldots,10$ are independent random variables generated from the standard normal distribution. The true intercept and bivariate regression parameter functions are generated as $\alpha(u) = 2e^{\{} - (u-1)^2\}$ and $\beta(v,u) = 4\cos(2\pi u)\sin(\pi v)$, respectively, Then, the elements of the functional response are generated as follows:

$$Y_i(u) = \alpha(u) + \int X_i(v)\beta(v,u)dv + \varepsilon_i(u),$$

where $\varepsilon_i(u)$ are generated from one of the following distributions:

When err.dist is "normal", then the error terms are generated from the standard normal distribution.

When err.dist is "t", then the error terms are generated from the t distribution with five degrees of freedom.

When err.dist is "chisq", then the error terms are generated from the chi-square distribution with one degree of freedom.

When err.dist is "cn", then the error terms are generated from the contaminated normal distribution.

Value

x	A matrix containing the observations of simulated functional predictor variable.
y	A matrix containing the observations of simulated functional response variable.
x.true	A matrix containing the observations of simulated functional predictor variable without measurement error.
y.true	A matrix containing the observations of simulated functional response variable without measurement error.
coef.a	A vector containing the intercept function.
coef.b	A matrix containing the bivariate regression coefficient vector.

Author(s)

Muge Mutis, Ufuk Beyaztas, Filiz Karaman, and Han Lin Shang

References

C. Xiong, X. Liugen, and C. Jiguo (2021), "Robust penalized M-estimation for function-on-function linear regression"", *Stat*, **10**(1), e390.

Examples

```
gpx <- (1:50)/50
gpy <- (1:60)/60
data <- fpqr_dgp(n = 250, gpy = gpy, gpx = gpx, err.dist = "normal")
y <- data$y
x <- data$x
```

---

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_fpqr *Prediction for a function-on-function linear quantile regression model based on functional partial quantile regression*

Description

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

Usage

```
predict_fpqr(object, xnew)
```

Arguments

object	An output object obtained from fpqr.
xnew	A matricx consisting of the new observations of functional predictor. The argument xnew must have the same length and the same structure as the input x of fpqr.

Value

A matrix of predicted values of the functional response variable for the given set of new functional predictor xnew.

Author(s)

Muge Mutis, Ufuk Beyaztas, Filiz Karaman, and Han Lin Shang

Examples

```
gpx <- (1:25)/25
gpy <- (1:30)/30
data <- fpqr_dgp(n = 10, gpy = gpy, gpx = gpx, err.dist = "normal")
y <- data$y
x <- data$x
data.test <- fpqr_dgp(n = 10, gpy = gpy, gpx = gpx, err.dist = "normal")
x.test <- data.test$x
y.test <- data.test$y.true
fpqr.model.li <- fpqr(y=y, x=x, tau = 0.5, gpx = gpx, gpy = gpy, qc.type = "li")
predictions <- predict_fpqr(object = fpqr.model.li, xnew = x.test)
```

---

predict_sffr2SLS *Out-of-sample prediction for Penalised Spatial FoFR models*

Description

Given a fitted model returned by sffr_pen2SLS, this function produces predicted functional responses at new spatial units whose functional covariates and spatial weight matrix are supplied by the user. A fixed-point solver enforces the spatial autoregressive feedback implicit in the SFoFR model.

Usage

```
predict_sffr2SLS(object, xnew, Wnew)
```

Arguments

`object`	An object of class `"sffr2SLS"`: the list produced by `sffr_pen2SLS`. At minimum it must contain `gpx`, `gpy`, `K0`, `Ky`, `Kx`, and the B-spline coefficient matrices `b0_mat`, `b_mat`, `r_mat`.
`xnew`	Numeric matrix of dimension $n_{\text{new}} \times \lvert\text{gpx}\rvert$, holding the functional covariate for the *new* spatial units, evaluated on the same predictor grid used during model fitting.
`Wnew`	Row-normalised $n_{\text{new}} \times n_{\text{new}}$ spatial weight matrix that captures proximity among the *new* units. Its definition should mirror that of the training matrix (e.g. inverse distance, k-nearest neighbours, etc.).

Details

Let $\widehat{\beta}_0(t)$, $\widehat{\beta}(t,s)$, and $\widehat{\rho}(t,u)$ be the estimated surfaces stored in `object`. For each new unit i the algorithm first forms the non-spatial regression prediction

$$\widehat{G}_i(t) \;=\; \widehat{\beta}_0(t) + \int_0^1 X_i(s)\,\widehat{\beta}(t,s)\,ds,$$

computed efficiently by pre-evaluated B-spline bases. Spatial feedback is then introduced by iterating

$$Y_i^{(\ell+1)}(t) \;=\; \widehat{G}_i(t) + \sum_{j=1}^{n_{\text{new}}} w_{ij} \int_0^1 Y_j^{(\ell)}(u)\,\widehat{\rho}(t,u)\,du,$$

until the sup-norm difference between successive curves falls below `1e-3` or `1,000` iterations are reached. Convergence is guaranteed when $\|\widehat{\rho}\|_\infty < 1/\|Wnew\|_\infty$, a condition typically satisfied by the fitted model if the training weight matrix met it during estimation.

Value

A numeric matrix of dimension $n_{\text{new}} \times \lvert\text{gpy}\rvert$ containing the predicted functional responses evaluated on `gpy`. Row i corresponds to the i-th row of `xnew`.

Note

If the new weight matrix induces very strong dependence, the fixed-point iterations may converge slowly. Consider scaling `Wnew` to have $\|Wnew\|_\infty \leq 1$ or relaxing the tolerance.

Author(s)

Ufuk Beyaztas, Han Lin Shang, and Gizel Bakicierler Sezer

References

Beyaztas, U., Shang, H. L., and Sezer, G. B. (2025). Penalised Spatial Function-on-Function Regression. *Journal of Agricultural, Biological, and Environmental Statistics*, **in press**.

See Also

sff_dgp for simulated data generation; sffr_pen2SLS for model fitting.

Examples

```
# 1. Fit a model on small simulated data
train <- sff_dgp(n = 500, rf = 0.5)
lam   <- list(lb = c(10^{-3}, 10^{-2}, 10^{-1}), lrho = c(10^{-3}, 10^{-2}, 10^{-1}))
fit <- sffr_pen2SLS(train$Y, train$X, train$W,
                    gpy = seq(0, 1, length = 101),
                    gpx = seq(0, 1, length = 101),
                    K0 = 10, Ky = 10, Kx = 10,
                    lam_cands = lam)

# 2. Simulate NEW covariates and a compatible weight matrix
test <- sff_dgp(n = 1000, rf = 0.5)  ## we keep only X and W
pred <- predict_sffr2SLS(fit, xnew = test$X, Wnew = test$W)
```

---

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)
```

---

sffr_pen2SLS *Penalised Spatial Two-Stage Least Squares for SFoFR*

Description

Fits the penalised spatial function-on-function regression (SFoFR) model via the two-stage least-squares (Pen2SLS) estimator introduced by Beyaztas, Shang and Sezer (2025). It selects optimal smoothing parameters, estimates regression and spatial autocorrelation surfaces, and (optionally) builds percentile bootstrap confidence bands.

Usage

```
sffr_pen2SLS(y, x, W, gpy, gpx, K0, Ky, Kx, lam_cands,
    boot = FALSE, nboot = NULL, percentile = NULL)
```

Arguments

y	n x length(gpy) matrix of functional responses evaluated on grid gpy.
x	n x length(gpx) matrix of functional predictors evaluated on grid gpx.
W	n x n row-normalised spatial weight matrix, typically inverse-distance.
gpy	Numeric vector of response evaluation points $t \in [0, 1]$.
gpx	Numeric vector of predictor evaluation points $s \in [0, 1]$.
K0	Integer; number of basis functions for the intercept $\beta_0(t)$.
Ky	Integer; number of basis functions in the *response* direction of the bivariate surfaces $\rho(t, u)$ and $\beta(t, s)$.
Kx	Integer; number of basis functions in the *predictor* direction of the regression surface $\beta(t, s)$.
lam_cands	Two-column matrix or data frame whose rows contain candidate smoothing pairs $(\lambda_\rho,\ \lambda_\beta)$ to be ranked by BIC.
boot	Logical; if TRUE percentile bootstrap confidence intervals are produced.
nboot	Number of bootstrap resamples. Required when boot = TRUE.
percentile	Desired CI nominal width in percent (e.g., 95). Required when boot = TRUE.

Details

The estimator minimises the penalised objective

$$\|Z^*\{\mathrm{vec}(Y) - \Pi\theta\}\|^2 \ + \ \frac{1}{2}\lambda_\rho P(\rho) \ + \ \frac{1}{2}\lambda_\beta P(\beta),$$

where $\theta = (\mathrm{vec}\,\rho, \mathrm{vec}\,\beta)$ are tensor-product B-spline coefficients, Z^* is the projection onto instrumental variables, and $P(\cdot)$ are Kronecker-sum quadratic roughness penalties in both surface directions. Candidate smoothing pairs are scored by the Bayesian Information Criterion

$$\mathrm{BIC} = -2\log\mathcal{L} + \omega\log n,$$

with log-likelihood based on squared residuals and ω equal to the effective degrees of freedom.

If boot = TRUE, residuals are centred, resampled, and the entire estimation procedure is repeated nboot times. Lower and upper percentile bounds are then extracted for $\beta(t, s)$, $\rho(t, u)$, and $\widehat{Y}_i(t)$.

Value

A named list:

b0hat Estimated intercept curve $\widehat{\beta}_0(t)$.

bhat Matrix of $\widehat{\beta}(t, s)$ values.

rhohat Matrix of $\widehat{\rho}(t, u)$ values.

b0_mat, b_mat, r_mat Raw coefficient matrices of B-spline basis weights for β_0, β, and ρ.

fitted.values $\widehat{Y}_i(t)$ matrix.

residuals $\widehat{\varepsilon}_i(t)$ matrix.

CI_bhat Two-element list with lower/upper percentile surfaces (NULL unless `boot = TRUE`).

CI_rhohat Analogous list for ρ.

CIy Percentile bands for the fitted responses.

gpy, gpx, K0, Ky, Kx Returned for convenience.

Author(s)

Ufuk Beyaztas, Han Lin Shang, and Gizel Bakicierler Sezer

References

Beyaztas, U., Shang, H. L., and Sezer, G. B. (2025). *Penalised Spatial Function–on–Function Regression*. *Journal of Agricultural, Biological, and Environmental Statistics*, **in press**.

Examples

```
# 1. simulate data
sim <- sff_dgp(n = 100, rf = 0.7)
# 2. candidate smoothing grid (four pairs)
lam <- list(lb = c(10^{-3}, 10^{-2}, 10^{-1}),
            lrho = c(10^{-3}, 10^{-2}, 10^{-1}))
# 3. fit model without bootstrap
fit <- sffr_pen2SLS(y = sim$Y, x = sim$X, W = sim$W,
   gpy = seq(0, 1, length.out = 101),
   gpx = seq(0, 1, length.out = 101),
   K0 = 10, Ky = 10, Kx = 10,
   lam_cands = lam, boot = FALSE)
```

`sff_dgp` *Simulate data from a Spatial Function–on–Function Regression model*

Description

Generates synthetic functional predictors and responses from the spatial function-on-function regression (SFoFR) data-generating process described in Beyaztas, Shang and Sezer (2025). The model embeds spatial autoregression on the functional response, Fourier-type basis structure for the covariate, and user-controlled Gaussian noise.

Usage

```
sff_dgp(
  n,
  nphi     = 10,
  gpy      = NULL,
  gpx      = NULL,
```

```
  rf       = 0.9,
  sd.error = 0.01,
  tol      = 0.001,
  max_iter = 1000
)
```

Arguments

`n`	Number of spatial units (curves) to generate.
`nphi`	Number of sine *and* cosine basis functions used to build each functional predictor. Total latent scores generated are therefore `2 * nphi`.
`gpy`	Numeric vector of evaluation points for the response domain $t \in [0, 1]$. Defaults to an equally-spaced grid of 101 points.
`gpx`	Numeric vector of evaluation points for the predictor domain $s \in [0, 1]$. Defaults to an equally-spaced grid of 101 points.
`rf`	Scalar in $(0, 1)$ controlling the strength of spatial autocorrelation through the surface $\rho(t, u)$. Values closer to 1 yield stronger dependence.
`sd.error`	Standard deviation of the i.i.d. Gaussian noise $\varepsilon_i(t)$ added to the latent regression part.
`tol`	Absolute tolerance used in the fixed-point iteration that solves the spatial autoregressive operator equation (stopping rule on the sup-norm of successive iterates).
`max_iter`	Maximum number of fixed-point iterations. Prevents infinite looping when strong spatial feedback and small `tol` interact.

Details

The generator mimics the penalised SFoFR set-up:

$$Y_i(t) \;=\; \sum_{j=1}^{n} w_{ij} \int_0^1 Y_j(u)\, \rho(t,u)\, du \;+\; \int_0^1 X_i(s)\, \beta(t,s)\, ds \;+\; \varepsilon_i(t),$$

where

- w_{ij} are row-normalised inverse-distance weights,
- $X_i(s)$ is built from Fourier scores $\xi_{ijk} \sim \mathcal{N}(0,1)$ and damped basis functions $\phi_k^{\cos}(s) = (k^{-3/2})\sqrt{2}\cos(k\pi s)$ and $\phi_k^{\sin}(s) = (k^{-3/2})\sqrt{2}\sin(k\pi s)$,
- the regression surface is $\beta(t,s) = 2 + s + t + 0.5\sin(2\pi s t)$,
- the spatial autocorrelation surface is $\rho(t,u) = rf\,(1+ut)/(1+|u-t|)$,
- $\varepsilon_i(t) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$, with $\sigma =$ `sd.error`.

Given the contraction condition $\|\rho\|_\infty < 1/\|W\|_\infty$, the Neumann series defining $(\mathbb{I} - \mathcal{T})^{-1}$ converges and the solution is obtained by simple fixed-point iterations until the change is below `tol`. Full details are in Beyaztas, Shang and Sezer (2025).

Value

A named `list` with components

Y `n` x `length(gpy)` matrix of observed functional responses on the grid `gpy`.

Y_true Same dimension as `Y`; noise-free latent responses before adding $\varepsilon_i(t)$.

X `n` x `length(gpx)` matrix of functional predictors.

W `n` x `n` row-normalised spatial weight matrix based on inverse distances.

rho `length(gpy)` x `length(gpy)` matrix containing $\rho(t, u)$ evaluated on the response grid.

beta `length(gpx)` x `length(gpy)` matrix containing $\beta(t, s)$ evaluated on the Cartesian product of the predictor and response grids.

Author(s)

Ufuk Beyaztas, Han Lin Shang, and Gizel Bakicierler Sezer

References

Beyaztas, U., Shang, H. L., and Sezer, G. B. (2025). Penalised Spatial Function-on-Function Regression. *Journal of Agricultural, Biological, and Environmental Statistics*, **in press**.

Examples

```
# generate a toy data set
dat <- sff_dgp(n = 250, rf = 0.5)
```

Index