Package ‘robfpqr’

June 10, 2022

Type Package

Title Robust Functional Partial Quantile Regression

Version 1.0

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

Imports expm, fda.usc, goffda, pcaPP

LazyLoad yes

ByteCompile TRUE

Encoding UTF-8

Maintainer Ufuk Beyaztas <ufukbeyaztas@gmail.com>

Description Functions for implementing robust methods for functional linear quantile regression.

License GPL-3

R topics documented:

`dgp` *Generate a dataset for the scalar-on-function regression model*

Description

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

$$Y = \int X(t)\beta(t)dt + \epsilon,$$

where Y denotes the scalar response, $X(t)$ denotes the functional predictor, $\beta(t)$ denotes the regression coefficient function, and ϵ is the error process.

Usage

```
dgp(n.train, n.test, n.gp, data.type = c("normal", "contaminated"),
                out.type = c("y","yx"), out.perc)
```

Arguments

`n.train`	An integer, specifying the number of observations for each variable to be generated in the training sample.
`n.test`	An integer, specifying the number of observations for each variable to be generated in the test sample.
`n.gp`	An integer, denoting the number of grid points, i.e., a fine grid on the interval [0, 1].
`data.type`	Data type to be generated. Possibilities are "normal" and "contaminated".
`out.type`	Outlier type to be generated. Possibilities are "y" and "yx".
`out.perc`	A numeric value between 0 and 1 specifying the outlier percentage.

Details

In the data generation process, first, the functional predictor is generated based on the following process:

$$X(t) = \sum_{j=1}^{5} \kappa_j v_j(t),$$

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

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

The regression coefficient function is generated from $2\sin(2\pi t)$. The error process is generated from the standard normal distribution.

If `data.type = "normal"`, then the data are generated as above. On the other hand, if `data.type = "contaminated"`, then $n.train \times out.perc$ part of the data in the training sample is contaminated by outliers. If `out.type = "y"`, then only the scalar response variable is contaminated by outiers. While doing so, $n.train \times out.perc$ of ϵ is generated from a normal distribution with mean 15 and variance 1. If `out.type = "yx"`, then both the scalar response and functional predictor are contaminated by outliers. In doing so, $n.train \times out.perc$ of $X(t)$ are contaminated by an Ornstein-Uhlenbeck process. In addition, $n.train \times out.perc$ of ϵ is generated from a normal distribution with mean 15 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.train`	An $n.train \times$ 1-dimensional matrix containing the observations of simulated scalar response variable in the training sample.
`y.test`	An $n.test \times$ 1-dimensional matrix containing the observations of simulated scalar response variable in the test sample.
`x.train`	A matrix with dimension $n.train \times n.gp$ containing the observations of simulated functional predictor variable in the training sample.
`x.test`	A matrix with dimension $n.test \times n.gp$ containing the observations of simulated functional predictor variable in the test sample.
`f.coef`	A vector with length $n.gp$ containing the generated regression coefficient function.

Author(s)

Ufuk Beyaztas, Mujgan Tez and Han Li Shang

Examples

```
library(fda.usc)
sim.data <- dgp(n.train = 420, n.test = 180, n.gp = 201, data.type = "contaminated",
                out.type = "yx", out.perc = 0.1)
y.train <- sim.data$y.train
y.test <- sim.data$y.test
x.train <- sim.data$x.train
x.test <- sim.data$x.test
coeffs <- sim.data$f.coef
fx <- fdata(x.train, argvals = seq(0, 1, length.out = 201))
par(mfrow = c(1,2))
plot(y.train, type = "p", pch = 16, xlab = "Index", ylab = "", main = "Response")
plot(fx, lty = 1, ylab = "", xlab = "Grid point",
     main = expression(X(t)), mgp = c(2, 0.5, 0))
dev.off()
```

`fpcr` *Scalar-on-function linear quantile regression based on principal component analysis*

Description

This function can be used to perform both scalar-on-function linear regression model

$$Y = \int X(t)\beta(t)dt + \epsilon,$$

and scalar-on-function linear quantile regression model

$$Q_\tau(Y|X) = \int X(t)\beta_\tau(t)dt$$

based on the functional principal component decomposition of the functional predictor.

Usage

```
fpcr(y, x, tau, nbf, gp, ncp, model.type = c("linear", "quantile"))
```

Arguments

`y`	An $n \times 1$-dimensional matrix containing the observations of scalar response Y, where n denotes the sample size.
`x`	A matrix with dimension $n \times p$ containing the observations of functional predictor $X(t)$, where n is the sample size and p denotes the number of grid points for $X(t)$.
`tau`	Quantile level.
`nbf`	A numeric value denoting the number of B-spline basis expansion functions to be used to approximate the functional principal components for the functional predictor $X(t)$.

`gp`	A vector containing the grid points of the functional predictor $X(t)$.
`ncp`	A numeric value denoting the number of functional principal components to be computed for the functional predictor $X(t)$.
`model.type`	Fitting model used to estimate the scalar-on-function regression model. Possibilities are "linear" and "quantile".

Value

A list object with the following components:

`y`	An $n \times 1$-dimensional matrix containing the observations of scalar response Y.
`x`	A matrix with dimension $n \times p$ containing the observations of functional predictor $X(t)$.
`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.
`coeffs`	A vector containing the estimate of parameters of the regression model coducted between the scalar response and principal component scores of the functional predictor.
`model.details`	A list object containing model details, such as number of basis functions, number of principal components, and grid points used for the functional predictor variable.

Author(s)

Ufuk Beyaztas, Mujgan Tez and Han Lin Shang

Examples

```
sim.data <- dgp(n.train = 420, n.test = 180, n.gp = 201, data.type = "contaminated",
            out.type = "yx", out.perc = 0.1)
y <- sim.data$y.train
x <- sim.data$x.train
gp <- seq(0, 1, length.out = 201) # grid points of X
model.linear <- fpcr(y=y, x=x, nbf=20, gp=gp, ncp=4,
                     model.type = "linear")
model.quantile <- fpcr(y=y, x=x, tau=0.5, nbf=20, gp=gp, ncp=4,
                     model.type = "quantile")
```

`fpqr` *Functional partial quantile regression*

Description

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

$$Q_\tau(Y|X) = \int X(t)\beta_\tau(t)dt$$

based on the partial quantile regression.

Usage

```
fpqr(y, x, tau, h, nbasis, gp, method.type = c("classical","robust"),
     probp1 = 0.95, hampelp2 = 0.975, hampelp3 = 0.999,
     maxit = 1000, conv = 0.01)
```

Arguments

`y`	An $n \times 1$-dimensional matrix containing the observations of scalar response Y, where n denotes the sample size.
`x`	A matrix with dimension $n \times p$ containing the observations of functional predictor $X(t)$, where n is the sample size and p denotes the number of grid points for $X(t)$.
`tau`	Quantile level.
`h`	A numeric value denoting the number of functional partial quantile regression components to be computed.
`nbasis`	A numeric value denoting the number of B-spline basis expansion functions to be used to approximate the functional partial quantile regression components.
`gp`	A vector containing the grid points of the functional predictor $X(t)$.
`method.type`	Method type used to estimate the scalar-on-function linear quantile regression model. Possibilities are "classical" and "robust".
`probp1`	A numeric value used to determine the first outlier cutoff point for the weights.
`hampelp2`	A numeric value used to determine the first outlier cutoff point for the weights.
`hampelp3`	A numeric value used to determine the third outlier cutoff point for the weights.
`maxit`	An integer value defining the maximum iteration used to achieve convergence.
`conv`	A numeric value used for the precision of the coefficient estimate.

Details

If `method.type = "classical"`, then, the partial quantile regression algorithm of Dodge and Whittaker (2009) is used to obtain functional partial quantile regression components.

If `method.type = "robust"`, then, the partial quantile regression algorithm of Dodge and Whittaker (2009) along with a modified version of the iteratively reweighting algorithm of Serneels et al. (2005) is used to obtain functional partial quantile regression components.

Value

A list object with the following components:

`y`	An $n \times 1$-dimensional matrix containing the observations of scalar response Y.
`x`	A matrix with dimension $n \times p$ containing the observations of functional predictor $X(t)$.
`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.
`coeffs`	A vector containing the estimate of parameters of the regression model coducted between the scalar response and principal component scores of the functional predictor.
`intercept`	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 predictor variable.

Author(s)

Ufuk Beyaztas, Mujgan Tez and Han Lin Shang

References

Y. Dodge and J. Whittaker (2009), "Partial quantile regression, *Metrika*, **70**(1), 35–57.

S. Serneels and C. Croux and P. Filzmoser and P. J. V. Espen (2005), "Partial robust M-regression", *Chemometrics and Intelligent Laboratory Systems*, **79**(1-2), 55–64.

Examples

```
sim.data <- dgp(n.train = 420, n.test = 180, n.gp = 201, data.type = "contaminated",
            out.type = "yx", out.perc = 0.1)
y <- sim.data$y.train
x <- sim.data$x.train
gp <- seq(0, 1, length.out = 201) # grid points of X
model.classic <- fpqr(y=y, x=x, tau=0.5, h=4, nbasis=20, gp=gp, method.type = "classical")
model.robust <- fpqr(y=y, x=x,tau=0.5, h=4, nbasis=20, gp=gp, method.type = "robust")
```

---

get.fpcr.coeff — *Get the estimated regression coefficient functions for scalar-on-function linear and/or quantile regression model*

Description

This function is used to obtain the estimated regression coefficient function $\beta(t)$ or $\beta(t,\ \tau)$ for scalar-on-function linear and/or quantile regression model based on output object obtained from fpcr).

Usage

```
get.fpcr.coeff(object)
```

Arguments

object — The output object of fpcr.

Value

A vector containing the estimated coefficient function $\beta(t)$ or $\beta(t,\ \tau)$ depending on model.type.

Author(s)

Ufuk Beyaztas, Mujgan Tez and Han Lin Shang

Examples

```
sim.data <- dgp(n.train = 420, n.test = 180, n.gp = 201, data.type = "contaminated",
            out.type = "yx", out.perc = 0.1)
y <- sim.data$y.train
x <- sim.data$x.train
gp <- seq(0, 1, length.out = 201) # grid points of X
model.linear <- fpcr(y=y, x=x, nbf=20, gp=gp, ncp=4,
                     model.type = "linear")
model.quantile <- fpcr(y=y, x=x, tau=0.5, nbf=20, gp=gp, ncp=4,
                     model.type = "quantile")
linear.coeff.fun <- get.fpcr.coeff(object = model.linear)
quantile.coeff.fun <- get.fpcr.coeff(object = model.quantile)
```

---

get.fpqr.coeff — *Get the estimated regression coefficient functions for functional partial quantile regression model*

Description

This function is used to obtain the estimated regression coefficient function $\beta(t,\ \tau)$ for scalar-on-function linear quantile regression model based on output object obtained from fpqr).

Usage

```
get.fpqr.coeff(object)
```

Arguments

object	The output object of fpqr.

Value

A vector containing the estimated coefficient function $\beta(t,\ \tau)$.

Author(s)

Ufuk Beyaztas, Mujgan Tez and Han Lin Shang

Examples

```
sim.data <- dgp(n.train = 420, n.test = 180, n.gp = 201, data.type = "contaminated",
            out.type = "yx", out.perc = 0.1)
y <- sim.data$y.train
x <- sim.data$x.train
gp <- seq(0, 1, length.out = 201) # grid points of X
model.classic <- fpqr(y=y, x=x, tau=0.5, h=4, nbasis=20, gp=gp, method.type = "classical")
model.robust <- fpqr(y=y, x=x,tau=0.5, h=4, nbasis=20, gp=gp, method.type = "robust")
coeff.func.clsc <- get.fpqr.coeff(object = model.classic)
coeff.func.rbst <- get.fpqr.coeff(object = model.robust)
```

predict_fpcr	*Prediction for a scalar-on-function linear and/or quantile regression model based on functional principal component analysis*

Description

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

Usage

```
predict_fpcr(object, xnew)
```

Arguments

object	An output object obtained from fpcr.
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 fpcr.

Value

An $n_{test} \times 1$-dimensional matrix of predicted values of the scalar response variable for the given set of new functional predictor xnew. Here, n_{test}, the number of rows of the matrix of predicted values, equals to the number of rows of xnew.

Author(s)

Ufuk Beyaztas, Mujgan Tez and Han Lin Shang

Examples

```
set.seed(123)
sim.data <- dgp(n.train = 420, n.test = 180, n.gp = 201, data.type = "contaminated",
                out.type = "yx", out.perc = 0.1)
y.train <- sim.data$y.train
y.test <- sim.data$y.test
x.train <- sim.data$x.train
x.test <- sim.data$x.test
gp <- seq(0, 1, length.out = 201) # grid points of X
model.linear <- fpcr(y=y.train, x=x.train, nbf=20, gp=gp,
                     ncp=4, model.type = "linear")
model.quantile <- fpcr(y=y.train, x=x.train, tau=0.5, nbf=20,
                       gp=gp, ncp=4, model.type = "quantile")
predict.linear <- predict_fpcr(object=model.linear, xnew=x.test)
predict.quantile <- predict_fpcr(object=model.quantile, xnew=x.test)
# Mean squared prediction error
mspe.linear <- mean((predict.linear-y.test)^2)
mspe.quantile <- mean((predict.quantile-y.test)^2)
round(mspe.linear, 4) # 3.0747
round(mspe.quantile, 4) # 3.3102
```

predict_fpqr — *Prediction for a scalar-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 scalar-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

An $n_{test} \times 1$-dimensional matrix of predicted values of the scalar response variable for the given set of new functional predictor xnew. Here, n_{test}, the number of rows of the matrix of predicted values, equals to the number of rows of xnew.

Author(s)

Ufuk Beyaztas, Mujgan Tez and Han Lin Shang

Examples

```
set.seed(123)
sim.data <- dgp(n.train = 420, n.test = 180, n.gp = 201, data.type = "contaminated",
                out.type = "yx", out.perc = 0.1)
y.train <- sim.data$y.train
y.test <- sim.data$y.test
x.train <- sim.data$x.train
x.test <- sim.data$x.test
gp <- seq(0, 1, length.out = 201) # grid points of X
model.classic <- fpqr(y=y.train, x=x.train, tau=0.5, h=4, nbasis=20,
                      gp=gp, method.type = "classical")
model.robust <- fpqr(y=y.train, x=x.train,tau=0.5, h=4, nbasis=20, gp=gp,
                     method.type = "robust")
predict.classic <- predict_fpqr(object=model.classic, xnew=x.test)
predict.robust <- predict_fpqr(object=model.robust, xnew=x.test)
# Mean squared prediction error
mspe.classic <- mean((predict.classic-y.test)^2)
mspe.robust <- mean((predict.robust-y.test)^2)
round(mspe.classic, 4) # 3.2151
round(mspe.robust, 4) # 1.3902
```

Index