Package ‘QRegVCM’

March 16, 2018

Type Package

Title Quantile Regression in Varying-Coefficient Models

Version 1.2

Date 2018-02-26

Author 'Andriyana, Y.' [aut, cre], 'Ibrahim M. A.' [aut], 'Gijbels, I.' [ctb], 'Verhasselt A.' [ctb]

Maintainer `Andriyana.Y" <y.andriyana@unpad.ac.id>

Description Quantile regression in varying-coefficient models (VCM) using one particular nonparametric technique called P-splines. The functions can be applied on three types of VCM; (1) Homoscedastic VCM, (2) Simple heteroscedastic VCM, and (3) General heteroscedastic VCM.

Depends R (>= 3.4.0), quantreg, SparseM, truncSP

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2018-03-16 10:02:26 UTC

R topics documented:

AHeVT .......................................................... 2
AHeVXT ......................................................... 5
CD4 ............................................................ 8
QRIndiv .......................................................... 9
QRSimul .......................................................... 11
QRStepwise ..................................................... 14
QRWSimul ....................................................... 17
simul_shapetest ............................................... 20
test_variability .............................................. 22
wages ........................................................... 27

Index 29
Description

The adapted He (1997) approach considering a simple hetersocedastic varying-coefficient model, $V(t)$.

$$Y(t) = \sum_{k=0}^{p} \beta_k(t)X^{(k)}(t) + V(t)\varepsilon(t).$$

Usage

`AHeVT(VecX, times, subj, X, y, d, tau, kn, degree, lambda, gam)`

Arguments

- `VecX` The representative values for each covariate used to estimate the desired conditional quantile curves.
- `times` The vector of time variable.
- `subj` The vector of subjects/individuals.
- `X` The covariate containing 1 as its first component (including intercept in the model).
- `y` The response vector.
- `d` The order of differencing operator for each covariate.
- `tau` The quantiles of interest.
- `kn` The number of knots for each covariate.
- `degree` The degree of B-spline basis for each covariate.
- `lambda` The grid of smoothing parameter to control the trade of between fidelity and penalty term (use a fine grid of lambda).
- `gam` The power used in estimating the smooting parameter for each covariate (e.g. gam=1 or gam=0.5).

Value

- `hat_bt50` The median coefficients estimators.
- `hat_VT` The variability estimator.
- `C` The estimators of the tau-th quantile of the estimated residuals.
- `qhat` The conditional quantile curves estimator.

Note

Some warning messages are related to the function `rq.fit.sf`.n.
Author(s)

Yudhie Andriyana

References


See Also

*rq.fit.sfns as.matrix.csr truncSP*

Examples

data(PM10)

PM10 = PM10[order(PM10$day,PM10$hour,decreasing=FALSE),]

y = PM10$PM10[1:200]
times = PM10$hour[1:200]
subj = PM10$day[1:200]
dim = length(y)
x0 = rep(1,200)
x1 = PM10$cars[1:200]
x2 = PM10$wind.speed[1:200]

X = cbind(x0, x1, x2)

VecX = c(1, max(x1), max(x2))

##################################################
## Input parameters ##
##################################################
kn = c(10, 10, 10)
degree = c(3, 3, 3)
taus = seq(0.1, 0.9, 0.1)
lambdas = c(1, 1.5, 2)
d = c(1, 1, 1)
gam = 1/2
##################################################

AHe = AHeVT(VecX=VecX, times=times, subj=subj, X=X, y=y, d=d, tau=taus, kn=kn, degree=degree, lambda=lambdas, gam=gam)
hat_bt50 = AHe$hat_bt50
hat_VT = AHe$hat_Vt
C = AHe$C
```r
qhat = AHeS$qhat
qhat1 = qhat[,1]
qhat2 = qhat[,2]
qhat3 = qhat[,3]
qhat4 = qhat[,4]
qhat5 = qhat[,5]
qhat6 = qhat[,6]
qhat7 = qhat[,7]
qhat8 = qhat[,8]
qhat9 = qhat[,9]

hat_bt0 = hat_bt50[seq(1,dim)]
hat_bt1 = hat_bt50[seq((dim+1),(2*dim))]
hat_bt2 = hat_bt50[seq((2*dim+1),(3*dim))]

i = order(times, hat_VT, qhat1, qhat2, qhat3, qhat4, qhat5, qhat6, qhat7, 
    qhat8, qhat9, hat_bt0, hat_bt1, hat_bt2);
times = times[i]; hat_VT=hat_VT[i]; qhat1 = qhat1[i]; qhat2=qhat2[i];
    qhat3=qhat3[i]; qhat4=qhat4[i]; qhat5=qhat5[i]; qhat6=qhat6[i];
    qhat7=qhat7[i]; qhat8=qhat8[i]; qhat9=qhat9[i];
    hat_bt0=hat_bt0[i]; hat_bt1=hat_bt1[i]; hat_bt2=hat_bt2[i];

### Plot coefficients
plot(hat_bt0~times, lwd=2, type="l", xlab="hour", ylab="baseline PM10");
plot(hat_bt1~times, lwd=2, type="l", xlab="hour", 
    ylab="coefficient of cars");
plot(hat_bt2~times, lwd=2, type="l", xlab="hour", 
    ylab="coefficient of wind");

### Plot variability V(t)
plot(hat_VT~times, ylim=c(min(hat_VT), max(hat_VT)), xlab="hour", 
    ylab="", type="1", lwd=2);

### Plot conditional quantiles estimators
ylim = c(-4, 6)
plot(qhat1~times, col="magenta", cex=0.2, lty=5, lwd=2, type="l", 
    ylim=ylim, xlab="hour", ylab="PM10");
lines(qhat2~times, col="aquamarine4", cex=0.2, lty=4, lwd=2);
lines(qhat3~times, col="blue", cex=0.2, lty=3, lwd=3);
lines(qhat4~times, col="brown", cex=0.2, lty=2, lwd=2);
lines(qhat5~times, col="black", cex=0.2, lty=1, lwd=2);
lines(qhat6~times, col="orange", cex=0.2, lty=2, lwd=2)
lines(qhat7~times, col="darkcyan", cex=0.2, lty=3, lwd=3);
lines(qhat8~times, col="green", cex=0.2, lty=4, lwd=2);
lines(qhat9~times, col="red", cex=0.2, lty=5, lwd=3)
```
The adapted He (1997) approach considering a general heteroscedastic varying-coefficient model, \( V(X(t), t) \).

\[
Y(t) = \sum_{k=0}^{p} \beta_k(t)X^{(k)}(t) + V(X(t), t)\varepsilon(t)
\]

where

\[
V(X(t), t) = \sum_{k=0}^{p} \gamma_k(t)X^{(k)}(t)
\]

Usage

\texttt{AHeVXT}(\texttt{VecX}, \texttt{times}, \texttt{subj}, \texttt{X}, \texttt{y}, \texttt{d}, \texttt{tau}, \texttt{kn}, \texttt{degree}, \texttt{lambda}, \texttt{gam})

Arguments

\begin{itemize}
  \item \texttt{VecX} The representative values for each covariate used to estimate the desired conditional quantile curves.
  \item \texttt{times} The vector of the time variable.
  \item \texttt{subj} The vector of subjects/individuals.
  \item \texttt{X} The covariate matrix containing 1 as its first column (including intercept in the model).
  \item \texttt{y} The response vector.
  \item \texttt{d} The order of the differencing operator for each covariate.
  \item \texttt{tau} The quantiles of interest.
  \item \texttt{kn} The number of knots for each covariate.
  \item \texttt{degree} The degree of the B-spline basis function for each covariate.
  \item \texttt{lambda} The grid for the smoothing parameter to control the trade of between fidelity and penalty term (use a fine grid of lambda).
  \item \texttt{gam} The power used in estimating the smoothing parameter for each covariate (e.g. \texttt{gam}=1 or \texttt{gam}=0.5).
\end{itemize}
Value

- `hat_bt50`: The median coefficients estimators.
- `hat_gt50`: The median coefficients estimators for the variability function $V(X(t),t)$.
- `hat_VXT`: The variability estimator.
- `C`: The estimators of the tau-th quantile of the estimated residuals.
- `qhat`: The conditional quantile curves estimator.

Note

Some warning messages are related to the function `rq.fit.sfn`.

Author(s)

Yudhie Andriyana

References


See Also

`rq.fit.sfn`, `as.matrix.csr`, `truncSP`

Examples

data(PM10)

PM10 = PM10[order(PM10$day, PM10$hour, decreasing=FALSE),]

y = PM10$PM10[1:200]
times = PM10$hour[1:200]
subj = PM10$day[1:200]
dim = length(y)
x0 = rep(1, 200)
x1 = PM10$cars[1:200]
x2 = PM10$wind.speed[1:200]

X = cbind(x0, x1, x2)

VecX = c(1, max(x1), max(x2))

#######################################
#### Input parameters ####
#######################################
\begin{verbatim}
kn = c(10, 10, 10)
degree = c(3, 3, 3)
taus = seq(0.1, 0.9, 0.1)
lambdas = c(1, 1.5, 2)
d = c(1, 1, 1)
gam = 1/2

AHe = AHeVXT(VecX=VecX, times=times, subj=subj, X=X, y=y, d=d, 
tau=taus, kn=kn, degree=degree, lambda=lambdas, gam=gam)
hat.bt50 = AHe$hat.bt50
hat.gt50 = AHe$hat.gt50
hat_VXT = AHe$hat_VXT
C = AHe$C
qhat = AHe$qhat

qhat1 = qhat[,1]
qhat2 = qhat[,2]
qhat3 = qhat[,3]
qhat4 = qhat[,4]
qhat5 = qhat[,5]
qhat6 = qhat[,6]
qhat7 = qhat[,7]
qhat8 = qhat[,8]
qhat9 = qhat[,9]

hat.bt0 = hat.bt50[seq(1,dim)]
hat.bt1 = hat.bt50[seq((dim+1),(2*dim))]  
hat.bt2 = hat.bt50[seq((2*dim+1),(3*dim))]

hat.gt0 = hat.gt50[seq(1,dim)]
hat.gt1 = hat.gt50[seq((dim+1),(2*dim))]  
hat.gt2 = hat.gt50[seq((2*dim+1),(3*dim))]

i = order(times, hat_VXT, qhat1, qhat2, qhat3, qhat4, qhat5, qhat6, qhat7, 
qhat8, qhat9, 
hat.bt0, hat.bt1, hat.bt2, hat.gt0, hat.gt1, hat.gt2); 
times = times[i]; hat_VXT=hat_VXT[i]; qhat1 = qhat1[i]; qhat2=qhat2[i]; 
qhat3=qhat3[i]; qhat4=qhat4[i]; qhat5=qhat5[i]; qhat6=qhat6[i]; 
qhat7=qhat7[i]; qhat8=qhat8[i]; qhat9=qhat9[i]; 
hat.bt0=hat.bt0[i]; hat.bt1=hat.bt1[i]; hat.bt2=hat.bt2[i]; 
hat.gt0=hat.gt0[i]; hat.gt1=hat.gt1[i]; hat.gt2=hat.gt2[i];

### Plot coefficients 

plot(hat.bt0~times, lwd=2, type="l", xlab="hour", ylab="baseline PM10"); 
plot(hat.bt1~times, lwd=2, type="l", xlab="hour", ylab="coefficient of cars"); 
plot(hat.gt2~times, lwd=2, type="l", xlab="hour", ylab="coefficient of wind");

###
\end{verbatim}
The CD4 dataset

Description

This dataset is a subset of the Multicenter AIDS cohort study. The data contain repeated measurements of physical examinations and CD4 percentages of 283 homosexual men who became HIV-positive between year 1984 and 1991.

Usage

CD4
Format

data frame with 6 variables and 1817 observations.

Details

It contains the following variables:

subj  the subject (man) indicator
Time  the time in years
Smooking the smoking status
Age  age at HIV infection
PreCD4 the pre-infection CD4 percentage
CD4  the CD4 percentage

Description

The estimation of conditional quantile curves using individual quantile objective function.

Usage

QRIndiv(VecX, tau, times, subj, X, y, d, kn, degree, lambda, gam)

Arguments

VecX The representative values for each covariate used to estimate the desired conditional quantile curves.
tau The quantiles of interest.
times The vector of the time variable.
subj The vector of subjects/individuals.
X The covariate matrix containing 1 as its first column (including intercept in the model).
y The response vector.
d The order of the differencing operator for each covariate.
kn The number of knots for each covariate.
degree The degree of the B-spline basis function for each covariate.
lambda The grid for the smoothing parameter to control the trade of between fidelity and penalty term (use a fine grid of lambda).
gam The power used in estimating the smothing parameter for each covariate (e.g. gam=1 or gam=0.5).
Value

alpha The estimator of the coefficient vector of the basis B-splines.
hat_bt The varying coefficients estimators.
qhat The conditional quantile curves estimator.

Note

Some warning messages are related to the function `rq.fit.sfn`.

Author(s)

Yudhie Andriyana

References


See Also

`rq.fit.sfn as.matrix.csr truncSP`

Examples

data(PM10)
PM10 = PM10[order(PM10$day,PM10$hour,decreasing=FALSE),]

y = PM10$PM10[1:200]
times = PM10$hour[1:200]
subj = PM10$day[1:200]
dim = length(y)
x0 = rep(1,200)
x1 = PM10$cars[1:200]
x2 = PM10$wind.speed[1:200]

X = cbind(x0, x1, x2)

VecX = c(1, max(x1), max(x2))

###############################################
#### Input parameters ####
###############################################
kn = c(10, 10, 10)
degree = c(3, 3, 3)
taus = seq(0.1,0.9,0.1)
lambdas = c(1,1.5,2)
d = c(1, 1, 1)
gam = 1/2
###############################################
qhat = QRIndiv(VcX=vecX, tau=taus, times=times, subj=subj, X=X, y=y, d=d, kn=kn, degree=degree, lambda=lamdas, gam=gam) & qhat

qhat1 = qhat[,1]
qhat2 = qhat[,2]
qhat3 = qhat[,3]
qhat4 = qhat[,4]
qhat5 = qhat[,5]
qhat6 = qhat[,6]
qhat7 = qhat[,7]
qhat8 = qhat[,8]
qhat9 = qhat[,9]

i = order(times, y, qhat1, qhat2, qhat3, qhat4, qhat5, qhat6, qhat7, qhat8, qhat9);

times = times[i]; y = y[i]; qhat1 = qhat1[i]; qhat2=qhat2[i];
qhat3=qhat3[i]; qhat4=qhat4[i]; qhat5=qhat5[i]; qhat6=qhat6[i];
qhat7=qhat7[i]; qhat8=qhat8[i]; qhat9=qhat9[i];

ylim = range(qhat1, qhat9)
plot(qhat1~times, col="magenta", cex=0.2, lty=5, lwd=2, type="l",
     ylim=ylim, xlab="hour", ylab="PM10");
lines(qhat2~times, col="aquamarine4", cex=0.2, lty=4, lwd=2);
lines(qhat3~times, col="blue", cex=0.2, lty=4, lwd=2);
lines(qhat4~times, col="brown", cex=0.2, lty=2, lwd=2);
lines(qhat5~times, col="black", cex=0.2, lty=1, lwd=2);
lines(qhat6~times, col="orange", cex=0.2, lty=2, lwd=2)
lines(qhat7~times, col="darkcyan", cex=0.2, lty=3, lwd=3);
lines(qhat8~times, col="green", cex=0.2, lty=4, lwd=2);
lines(qhat9~times, col="red", cex=0.2, lty=5, lwd=3)

legend("bottom", c(expression(tau=0.9), expression(tau=0.8),
       expression(tau=0.7), expression(tau=0.6), expression(tau=0.5),
       expression(tau=0.4), expression(tau=0.3), expression(tau=0.2),
       expression(tau=0.1)), ncol=1, col=c("red","green","darkcyan",
       "orange","black","brown","blue","aquamarine","magenta"),
lwd=c(2,2,2,2,2,2,2,2), lty=c(5,4,3,2,1,2,3,4,5))

QRSimul

Description

The estimation of conditional quantile curves using unweighted simultaneous objective function involving non-crossing constraints.
Usage

QRSimul(VecX, tau, times, subj, X, y, d, kn, degree, lambda, gam)

Arguments

VecX The representative values for each covariate used to estimate the desired conditional quantile curves.
tau The quantiles of interest.
times The vector of the time variable.
subj The vector of subjects/individuals.
X The covariate matrix containing 1 as its first column (including intercept in the model).
y The response vector.
d The order of the differencing operator for each covariate.
kn The number of knots for each covariate.
degree The degree of the B-spline basis function for each covariate.
lambda The grid for the smoothing parameter to control the trade of between fidelity and penalty term (use a fine grid of lambda).
gam The power used in estimating the smoothing parameter for each covariate (e.g. gam=1 or gam=0.5).

Value

w The weight for each subject corresponding to the length of its repeated measurement.
alpha The estimators of the coefficient vector of the basis B-splines.
hat_bt0 The baseline estimators.
hat_btk The varying coefficient estimators.
qhat_h The estimators of the $\tau_h$-th conditional quantile curves.

Note

Some warning messages are related to the function rq.fit.sfn.

Author(s)

Yudhie Andriyana

References


See Also

rq.fit sfN as.matrix csr truncSP

Examples

data(PM10)
PM10 = PM10[order(PM10$day, PM10$hour, decreasing = FALSE),]
y = PM10$PM10[1:200]
times = PM10$hour[1:200]
subj = PM10$day[1:200]
dim = length(y)
x0 = rep(1, 200)
x1 = PM10$cars[1:200]
x2 = PM10$wind.speed[1:200]
X = cbind(x0, x1, x2)
VecX = c(1, max(x1), max(x2))

# Input parameters
kn = c(10, 10, 10)
degree = c(3, 3, 3)
taus = seq(0.1, 0.9, 0.1)
lambdas = c(1, 1.5, 2)
d = c(1, 1, 1)
gam = 1

Simul = QRSimul(VecX = VecX, tau = taus, times = times, subj = subj, X = X, y = y,
d = d, kn = kn, degree = degree, lambda = lambdas, gam = gam)

hat_bt0 = Simul$hat_bt0
hat_btk = Simul$hat_btk
qhat = Simul$qhat

qhat1 = qhat[, 1]
qhat2 = qhat[, 2]
qhat3 = qhat[, 3]
qhat4 = qhat[, 4]
qhat5 = qhat[, 5]
qhat6 = qhat[, 6]
qhat7 = qhat[, 7]
qhat8 = qhat[, 8]
qhat9 = qhat[, 9]
i = order(times, y, qhat1, qhat2, qhat3, qhat4, qhat5, qhat6, qhat7, qhat8, qhat9);

times = times[i]; y = y[i]; qhat1 = qhat1[i]; qhat2=qhat2[i];
qhat3=qhat3[i]; qhat4=qhat4[i]; qhat5=qhat5[i]; qhat6=qhat6[i];
qhat7=qhat7[i]; qhat8=qhat8[i]; qhat9=qhat9[i];

ylim = range(qhat1, qhat9)
ylim = c(-4, 6)
plot(qhat1~times, col="magenta", cex=0.2, lty=5, lwd=2, type="l", ylim=ylim,
xlab="hour", ylab="pmQP";
lines(qhat2~times, col="aquamarine4", cex=0.2, lty=4, lwd=2);
lines(qhat3~times, col="blue", cex=0.2, lty=3, lwd=3);
lines(qhat4~times, col="brown", cex=0.2, lty=2, lwd=2);
lines(qhat5~times, col="black", cex=0.2, lty=1, lwd=2);
lines(qhat6~times, col="orange", cex=0.2, lty=2, lwd=2)
lines(qhat7~times, col="darkcyan", cex=0.2, lty=3, lwd=3);
lines(qhat8~times, col="green", cex=0.2, lty=4, lwd=2);
lines(qhat9~times, col="red", cex=0.2, lty=5, lwd=3)

legend("bottom", c(expression(tau==0.9), expression(tau==0.8),
expression(tau==0.7), expression(tau==0.6), expression(tau==0.5),
expression(tau==0.4), expression(tau==0.3), expression(tau==0.2),
expression(tau==0.1)), ncol=1, col=c("red","green","darkcyan",
"orange","black","brown","blue","aquamarine4","magenta"),
lwd=c(2,2,3,2,2,3,2,2,1), lty=c(5,4,3,2,1,2,3,4,5))

---

**QRStepwise**  
**Stepwise procedure**

**Description**  
The estimation of conditional quantile curves step-by-step involving the non-crossing constraints.

**Usage**  
QRStepwise(VecX, tau, times, subj, X, y, d, kn, degree, lambda, gam)

**Arguments**  
VecX The representative values for each covariate used to estimate the desired conditional quantile curves.  
tau The quantiles of interest.
times
  The vector of the time variable.
subj
  The vector of subjects/individuals.
x
  The covariate matrix containing 1 as its first column (including intercept in the model).
y
  The response vector.
d
  The order of the differencing operator for each covariate.
kn
  The number of knots for each covariate.
degree
  The degree of the B-spline basis function for each covariate.
lambda
  The grid for the smoothing parameter to control the trade of between fidelity and penalty term (use a fine grid of lambda).
gam
  The power used in estimating the smoothing parameter for each covariate (e.g. gam=1 or gam=0.5).

Value

alpha
  The estimators of the coefficient vector of the basis B-splines.
hat_bt
  The varying-coefficient estimators.
w
  The weight for each subject corresponding to the length of its repeated measurement.
qhat
  The conditional quantile curves estimator.

Note

Some warning messages are related to the function `rq.fit.sfn`.

Author(s)

Yudhie Andriyana

References


See Also

`rq.fit.sfn`, `as.matrix.csr`, `truncSP`
Examples

data(PM10)

PM10 = PM10[order(PM10$day, PM10$hour, decreasing=FALSE),]

y = PM10$PM10[1:200]
time_ub = PM10$hour[1:200]
subj = PM10$day[1:200]
dim = length(y)
x0 = rep(1,200)
x1 = PM10$cars[1:200]
x2 = PM10$wind.speed[1:200]

X = cbind(x0, x1, x2)
VecX = c(1, max(x1), max(x2))

####################################
### Input parameters ###
####################################
kn = c(10, 10, 10)
degree = c(3, 3, 3)
taus = seq(0.1, 0.9, 0.1)
lambdas = c(1, 1.5, 2)
d = c(1, 1, 1)
gam = 1
####################################

Step = QRStepwise(VecX=VecX, tau=taus, time_ub, subj, X, y, d, kn, degree, lambdas=lambdas, gam=gam)

qhat = Step$qhat

qhat1 = qhat[,1]
qhat2 = qhat[,2]
qhat3 = qhat[,3]
qhat4 = qhat[,4]
qhat5 = qhat[,5]
qhat6 = qhat[,6]
qhat7 = qhat[,7]
qhat8 = qhat[,8]
qhat9 = qhat[,9]

i = order(time_ub, y, qhat1, qhat2, qhat3, qhat4, qhat5, qhat6, qhat7, qhat8, qhat9);

time_ub = time_ub[i]; y = y[i]; qhat1 = qhat1[i]; qhat2=qhat2[i];
qhat3=qhat3[i]; qhat4=qhat4[i]; qhat5=qhat5[i]; qhat6=qhat6[i]; qhat7=qhat7[i]; qhat8=qhat8[i]; qhat9=qhat9[i];
The estimation of conditional quantile curves using weighted simultaneous objective function involving non-crossing constraints.

Usage

QRWSimul(VecX, tau, times, subj, X, y, d, kn, degree, lambda, gam)

Arguments

VecX  The representative values for each covariate used to estimate the desired conditional quantile curves.

tau  The quantiles of interest.

times  The vector of the time variable.

subj  The vector of subjects/individuals.

X  The covariate matrix containing 1 as its first column (including intercept in the model).

y  The response vector.

d  The order of the differencing operator for each covariate.
kn The number of knots for each covariate.
degree The degree of the B-spline basis function for each covariate.
lambda The grid for the smoothing parameter to control the trade of between fidelity and penalty term (use a fine grid of lambda).
gam The power used in estimating the smoothing parameter for each covariate (e.g. gam=1 or gam=0.5).

Value

w The weight for each subject corresponding to the length of its repeated measurement
alpha The estimators of the coefficient vector of the basis B-splines.
hat_bt0 The baseline estimators.
hat_btk The varying coefficient estimators.
qhat_h The estimators of the $\tau_h$-th conditional quantile curves.
$w_{tau}$ The weight of each order of quantile $\tau_h$.

Note

Some warning messages are related to the function rq.fit.sfN.

Author(s)

Yudhie Andriyana

References


See Also

rq.fit.sfN as.matrix.csr truncSP

Examples

data(PM10)
PM10 = PM10[order(PM10$day,PM10$hour,decreasing=FALSE),]
y = PM10$y
$PM10[1:200]$
times = PM10$hour[1:200]$
subj = PM10$day[1:200]$
$dim = length(y)$
x0 = rep(1,200)
x1 = PM10$cars[1:200]
x2 = PM10$wind.speed[1:200]

X = cbind(x0, x1, x2)
 VecX = c(1, max(x1), max(x2))

############################################################
 # Input parameters ###
############################################################
 kn = c(10, 10, 10)
degree = c(3, 3, 3)
taus = seq(0.1, 0.9, 0.1)
lambdas = c(1, 1.5, 2)
d = c(1, 1, 1)
gam = 1

############################################################

QRWSimul = QRWSimul(VecX=VecX, tau=taus, times=times, subj=subj, X=x, y=y,
 d=d, kn=kn, degree=degree, lambda=lambdas, gam=gam)

hat_bt0 = QRWSimul$hat_bt0
hat_btk = QRWSimul$hat_btk
qhat = QRWSimul$qhat
hat_Vt = QRWSimul$hat_Vt
Wtau = QRWSimul$Wtau

qhat1 = qhat[,1]
qhat2 = qhat[,2]
qhat3 = qhat[,3]
qhat4 = qhat[,4]
qhat5 = qhat[,5]
qhat6 = qhat[,6]
qhat7 = qhat[,7]
qhat8 = qhat[,8]
qhat9 = qhat[,9]

i = order(times, y, qhat1, qhat2, qhat3, qhat4, qhat5, qhat6, qhat7,
 qhat8, qhat9, hat_Vt);

times = times[i]; y = y[i]; qhat1 = qhat1[i]; qhat2=qhat2[i];
qhat3=qhat3[i]; qhat4=qhat4[i]; qhat5=qhat5[i]; qhat6=qhat6[i];
qhat7=qhat7[i]; qhat8=qhat8[i]; qhat9=qhat9[i];
hat_Vt = hat_Vt[i]

# Variability function V(t)

plot(hat_Vt~times, ylim=c(min(hat_Vt), max(hat_Vt)), xlab="hour",
 ylab="", type="l", lwd=2);

mtext(expression(hat(V)(t))), side=2, cex=1, line=3)
# Plot conditional quantiles estimators

```r
ylim = range(qhat1, qhat9)
ylim = c(-4, 6)
plot(qhat1~times, col="magenta", cex=0.2, lty=5, lwd=2, type="l", ylim=ylim,
     xlab="time since infection", ylab="CD4 percentage after infection");
lines(qhat2~times, col="aquamarine4", cex=0.2, lty=4, lwd=2);
lines(qhat3~times, col="blue", cex=0.2, lty=3, lwd=3);
lines(qhat4~times, col="brown", cex=0.2, lty=2, lwd=2);
lines(qhat5~times, col="black", cex=0.2, lty=1, lwd=2);
lines(qhat6~times, col="orange", cex=0.2, lty=2, lwd=2)
lines(qhat7~times, col="darkcyan", cex=0.2, lty=3, lwd=3);
lines(qhat8~times, col="green", cex=0.2, lty=4, lwd=2);
lines(qhat9~times, col="red", cex=0.2, lty=5, lwd=3)
legend("bottom", c(expression(tau==0.9), expression(tau==0.8),
expression(tau==0.7), expression(tau==0.6), expression(tau==0.5),
expression(tau==0.4), expression(tau==0.3), expression(tau==0.2),
expression(tau==0.1)), ncol=1, col=c("red","green","darkcyan",
"orange","black","brown","blue","aquamarine4","magenta"),
lwd=c(2,2,3,2,2,2,3,2,2,2,2,2), lty=c(5,4,3,2,1,2,3,4,5))
```

---

**simul_shapetest**  
*Testing the shape of a functional coefficient in the median and/or the variability function*

**Description**

Testing a functional coefficient of a covariate in the median and/or the variability function, considering the general heteroscedastic varying-coefficient model in Gijbels et al (2017a).

\[
Y(t) = \sum_{k=0}^{p} \beta_k(t)X^{(k)}(t) + \gamma(t) \exp\left(\sum_{k=1}^{p} \theta_k(t)X^{(k)}(t)\right)e(t).
\]

**Usage**

```r
simul_shapetest(times, subj, X, y, d, kn, degree, lambda, gam, v,
                 nr.bootstrap.samples, seed, test, omega)
```

**Arguments**

- **times**  
The vector of time variable.
- **subj**  
The vector of subject indicator.
- **X**  
The covariates, containing 1 as its first component (including intercept in the model).
- **y**  
The response vector.
The order of differencing operator for each covariate.

The number of knot intervals for each covariate.

The degree of B-spline basis for each covariate.

The grid of smoothing parameter to control the trade between fidelity and penalty term (use a fine grid of lambda).

The power used in estimating the smoothing parameter for each covariate (e.g. gam=1 or gam=0.5).

The covariate indicator for which the shape test is interested.

The number of bootstrap samples used.

The seed for the random generator in the bootstrap resampling.

The requested type of testing, it consists two arguments: the first argument for median and the second for the variability function. "c" stands for constancy, "m" stands for monotonicity, and "conv" stands for convexity. insert NA to the other argument when only for median/variability function is needed.

A user defined constraint parameter for monotonicity or convexity (in Equation (7) of Gijbels etal (2017a)), chosen as large as possible.

The testing procedures.

The p-values.

The test statistics for the given data.

The bootstrap test statistics.

Some warning messages are related to the function `rq.fit.sfn`.

Mohammed Abdulkerim Ibrahim


See Also

rq.fit.sfn as.matrix.csr truncSP

Examples

data(wages)
y = wages$resp ## the hourly wage
times = wages$exper ## the duration of work experience in years
subj = wages$id ## subject indicator (individual)
dim=length(y) ## number of rows in the data = 6402
x0 = rep(1,dim) ## for intercept
### the covariates
## creating 2 dummy variables for the race covariate
wages$r1[wages$race="black"] = 1
wages$r1[wages$race!="black"] = 0
wages$r2[wages$race="hisp"] = 1
wages$r2[wages$race!="hisp"] = 0
x1 = wages$r1 # stands for black
x2 = wages$r2 # stands for hispanic
x3 = wages$hgc ## the highest grade completed by the individual
X = cbind(x0, x1, x2, x3) ## the covariate matrix
px=ncol(X)

##############################
### Input parameters ###
##############################
lambda = 1 # we used 10*seq(-2, 1, 0.1) in Gijbels et al (2017a)
kn = rep(1,px) # used rep(5,px) in Gijbels et al (2017a)
degree = rep(2,px) # the degree of splines
d = rep(1,px)
gam=0.25
nr.bootstrap.samples=2 # used 200 in Gijbels et al (2017a)
seed=110
##############################
test1=simul_shapetest(times=times, subj=subj, X=X, y=y, d=d, kn=kn,
        degree=degree, lambda=lambda, gam=gam, v=1,
        nr.bootstrap.samples=nr.bootstrap.samples,seed=seed,
        test=c("c",NA),omega=10^3)

### Testing results
test1$result # the testing procedures
test1$P ## p-values
test1$R ## test statistics

---

**test_variability**

Variability Estimation and Testing
Description

Estimating and Testing the variability function using the following hetersocedastic varying-coefficient model.

\[ Y(t) = \sum_{k=0}^{p} \beta_k(t)X^{(k)}(t) + V(X(t), t)\epsilon(t) \]

where \( V(X(t), t) \) one of the six variability function in Gijbels et al (2017).

Usage

test_variability(times, subj, X, y, d, kn, degree, lambda, gam, tau, nr.bootstrap.samples, seed, test, model)

Arguments

times The vector of time variable.
subj The vector of subjects/individuals.
X The covariate containing 1 as its first component (including intercept in the model).
y The response vector.
d The order of differencing operator for each covariate.
kn The number of knot intervals for each covariate.
degree The degree of B-spline basis for each covariate.
lambda The grid of smoothing parameter to control the trade between fidelity and penalty term (use a fine grid of lambda).
gam The power used in estimating the smooting parameter for each covariate (e.g. gam=1 or gam=0.5).
tau The quantiles of interest.

nr.bootstrap.samples The number of bootstrap samples used.
seed The seed for the random generator in the bootstrap resampling.
test To request for testing the specific shape of the variability function ("Y" for test and "N" for only estimation of the parameters, the default is "Y").
model The variability model used to estimate the quantile of errors (the default is 4, model 4).

Value

est_median the median estimator.
hat_bt50 The median coefficients estimators.
qhat5_s2_m0 The variability (model 0) estimator.
qhat5_s2_m1 The variability (model 1) estimator.
qhat5_s2_m2 The variability (model 2) estimator.
qhat5_s2_m3  The variability (model 3) estimator.
qhat5_s2_m4  The variability (model 4) estimator.
qhat5_s2_m5  The variability (model 5) estimator.
hat_btv_0    The variability coefficients (model 0) estimators.
hat_btv_1    The variability coefficients (model 1) estimators.
hat_btv_2    The variability coefficients (model 2) estimators.
hat_btv_3    The variability coefficients (model 3) estimators.
hat_btv_4    The variability coefficients (model 4) estimators.
hat_btv_5    The variability coefficients (model 5) estimators.
C             The estimators of the tau-th quantile of the estimated residuals.
comp          The pairwise comparisons for testing the variability function.
P             The p-values.
GR            The test statistics for the given data.
Gb             The bootstrap test statistics.

Note

Some warning messages are related to the function `rq.fit.sfn`.

Author(s)

Mohammed Abdulkerim Ibrahim

References


See Also

rq.fit.sfn as.matrix.csr
**Examples**

```r
data(PM10)
PM10 = PM10[order(PM10$day, PM10$hour, decreasing=FALSE),]

y = PM10$PM10 ## the logarithm of the concentration of PM10
times = PM10$hour ## the time in hours
subj = PM10$day ## subject indicator (day)
dim=length(y) ## number of rows in the data is 500
x0 = rep(1, dim) ## for intercept

# the covariates
x1 = PM10$cars ## logarithm of number of cars per hour
x2 = PM10$wind.speed ## the wind speed (in meters/second)
x3 = PM10$temp ## the temperature (in degree Celsius)
X = cbind(x0, x1, x2, x3) ## the covariate matrix
px = ncol(X)

# Input parameters
lambda = Q # used in Gijbels et al (2017)
kn = rep(3, px) # used in Gijbels et al (2017)
degree = rep(3, px)
d = rep(1, px)
gam = 0.25
ten.bootstrap.samples = 4 # used in Gijbels et al (2017)
seed = 110
taus = 0.1

# Testing results

```
times.o = times[i]; qhat5_s2_m4_o=qhat5_s2_m4[i];
qhat5_s2_m5_o=qhat5_s2_m5[i]; qhat5_s2_m0_o=qhat5_s2_m0[i]; gamma0_o=gamma0[i];
gamma1_o=gamma1[i]; gamma2_o=gamma2[i]; gamma3_o=gamma3[i]

### variability functions plots
plot(qhat5_s2_m4_o~times_o, col="magenta", cex=0.2,
xlab="hour", ylab="estimated variability function")
lines(qhat5_s2_m5_o~times_o, col="red", cex=0.2, lty=1, lwd=2); 
lines(qhat5_s2_m0_o~times_o, col="black", cex=0.2, lty=5, lwd=2); 
legend("topleft", c("Model 4", "Model 5", "Model 0"), ncol=1, 
   col=c("magenta", "red", "black"), lwd=c(1,2,2), lty=c(3,1,5))

### Plot of coefficients for variability function
plot(gamma0~times, lwd=2, type="l", xlab="hour", ylab="coefficients")
plot_expression(hat(gamma)(7));
plot(gamma1~times, lwd=2, type="l", xlab="hour", ylab="coefficient of logarithm of number of cars per hour");
plot(gamma2~times, lwd=2, type="l", xlab="hour", ylab="coefficient of wind speed");
plot(gamma3~times, lwd=2, type="l", xlab="hour", ylab="coefficient of temperature")

## Not run:
# The real data example in Section 6 of Gijbels etal (2017)

data(CD4)

subj = CD4$subj ## subject indicator (a man)
dim = length(subj) ## number of rows in the data = 1817
y = CD4$CD4 ## the CD4 percentage
X0 = rep(1,dim) ## the intercept
X1 = CD4$Smoking ## the smoking status
X2 = CD4$Age ## age at HIV infection
X3 = CD4$PreCD4 ## the pre-infection CD4 percentage
times = CD4$Time ## the time in years
X = cbind(X0, X1, X2, X3) ## the covariate matrix
px=ncol(X)

lambdas = c(0.01,1,10) # used 10^seq(-2, 1, 0.1) in Gijbels etal (2017)
kn = rep(10,px) # the number of internal knots for each covariate
degree = rep(3,px) # the degree of splines
d = rep(1,px) ## The differencing order in the penalty term for each covariate
gam=0.25 ## the smoothing parameter for each covariate
nr.bootstrap.samples=100 ## used 200 in Gijbels etal (2017)
seed=110 ## the seed for the random generator in the bootstrap resampling

tax = seq(0,1,0.2)

test2=test_variability(times=times, subj=subj, X=X, y=y, d=d, kn=kn,
degree=degree, lambdas=lambdas, gam=gam, tau=tau,
nr.bootstrap.samples=nr.bootstrap.samples, seed=seed, test="Y", model=4)
wages

The wages dataset

Description

This dataset is taken from the National Longitudinal Survey of Youth, collected in U.S.A. The data contain repeated measurements for 888 individuals of age 14 – 17 years old.

Usage

wages
Format

a data.frame with 7 variables and 6402 observations.

Details

It contains the following variables:

- **id** the subject (individual) indicator
- **exper** the duration of work experience in years
- **hge** the highest grade completed by the individual
- **race** the race of the individual: black, hispanic and white
- **resp** the hourly wage
- **r1** the dummy variable for black individuals
- **r2** the dummy variable for hispanic individuals

References

Index

+Topic **datasets**
  CD4, 8
  wages, 27

AHeVT, 2
AHeVXT, 5
as.matrix.csr, 3, 6, 10, 13, 15, 18, 22, 24

CD4, 8

QRIndiv, 9
QRSimul, 11
QRStepwise, 14
QRWSimul, 17

rq.fit.sfn, 3, 6, 10, 13, 15, 18, 21, 22, 24

simul_shapetest, 20

test_variability, 22
truncSP, 3, 6, 10, 13, 15, 18, 22

wages, 27