Package 'QRegVCM'

July 19, 2016

Type Package

Title Quantile Regression in Varying-Coefficient Models

Version 1.0

Date 2016-07-19

Author Andriyana, Y. with contribution from Gijbels, I.

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 (>= 3.1.3), quantreg, SparseM, truncSP

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2016-07-19 18:57:11

R topics documented:

AHeVT *AHe V(t)-approach*

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.sfn` (See http://www.inside-r.org/packages/cran/quantreg/docs/sfnMessage).

Author(s)

Yudhie Andriyana

References

Andriyana, Y., Gijbels, I., and Verhasselt, A. P-splines quantile regression estimation in varying coefficient models. *Test* 23, 1 (2014a),153–194.

Andriyana, Y., Gijbels, I. and Verhasselt, A. (2014b). Quantile regression in varying coefficient models: non-crossingness and heteroscedasticity. *Manuscript*.

He, X. (1997). Quantile curves without crossing. *The American Statistician*, 51, 186–192.

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


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
```

```
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))]

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="l", lwd=2);
mtext(expression(hat(V)(t)), side=2, cex=1, line=3)

### 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)
```

```
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), lty=c(5,4,3,2,1,2,3,4,5))
```

AHeVXT *AHe V(X(t),t)-approach*

Description

The adapted He (1997) approach considering a general hetersocedastic 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

```
AHeVXT(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 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.
tau	The quantiles of interest.
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 smooting parameter for each covariate (e.g. gam=1 or gam=0.5).

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 (See http://www.inside-r.org/packages/cran/quantreg/docs/sfnMessage).

Author(s)

Yudhie Andriyana

References

Andriyana, Y., Gijbels, I., and Verhasselt, A. P-splines quantile regression estimation in varying coefficient models. *Test* 23, 1 (2014a),153–194.

Andriyana, Y., Gijbels, I. and Verhasselt, A. (2014b). Quantile regression in varying coefficient models: non-crossingness and heteroscedasticity. *Manuscript*.

He, X. (1997). Quantile curves without crossing. *The American Statistician*, 51, 186–192.

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

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_bt2~times, lwd=2, type="l", xlab="hour",
    ylab="coefficient of wind");
```

```
###
plot(hat_gt0~times, lwd=2, type="l", xlab="hour", ylab="baseline PM10");
plot(hat_gt1~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");

### Plot variability V(X(t),t)

plot(hat_VXT~times, ylim=c(min(hat_VXT), max(hat_VXT)), xlab="hour", ylab="");
mtext(expression(hat(V)(X(t),t)), side=2, cex=1, line=3)

### Plot conditional quantiles estimators

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="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)

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=3, col=c("red","green","darkcyan","orange",
      "black","brown","blue","aquamarine4","magenta"), lwd=c(2,2,3,2,2,2,3,2,2),
      lty=c(5,4,3,2,1,2,3,4,5))
```

QRIndiv *Individual quantile objective function*

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 smooting 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 (See http://www.inside-r.org/packages/cran/quantreg/docs/sfnMessage).

Author(s)

Yudhie Andriyana

References

Andriyana, Y., Gijbels, I., and Verhasselt, A. P-splines quantile regression estimation in varying coefficient models. *Test* 23, 1 (2014a),153–194.

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(VecX=VecX, tau=taus, times=times, subj=subj, X=X,
y=y, d=d, kn=kn, degree=degree, lambda=lambdas, 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=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), lty=c(5,4,3,2,1,2,3,4,5))
```

`QRSimul` *Unweighted simultaneous objective function*

Description

The etimation 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 smooting 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 τ_h-th conditional quantile curves.

Note

Some warning messages are related to the function rq.fit.sfn (See http://www.inside-r.org/packages/cran/quantreg/docs/sfnMessage).

Author(s)

Yudhie Andriyana

References

Andriyana, Y., Gijbels, I., and Verhasselt, A. P-splines quantile regression estimation in varying coefficient models. *Test* 23, 1 (2014a),153–194.

Andriyana, Y., Gijbels, I. and Verhasselt, A. (2014b). Quantile regression in varying coefficient models: non-crossingness and heteroscedasticity. *Manuscript*.

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="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)

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), lty=c(5,4,3,2,1,2,3,4,5))
```

`QRStepwise` *Stepwise procedure*

Description

The etimation 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 smooting 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` (See http://www.inside-r.org/packages/cran/quantreg/docs/sfnMessage).

Author(s)

Yudhie Andriyana

References

Andriyana, Y., Gijbels, I., and Verhasselt, A. P-splines quantile regression estimation in varying coefficient models. *Test* 23, 1 (2014a),153–194.

Andriyana, Y., Gijbels, I. and Verhasselt, A. (2014b). Quantile regression in varying coefficient models: non-crossingness and heteroscedasticity. *Manuscript*.

Wu, Y. and Liu, Y. Stepwise multiple quantile regression estimation using non-crossing constraints. *Statistics and Its Interface* 2, (2009), 299–310.

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,
       lambda=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];

ylim = range(qhat1, qhat9)
ylim = c(-4, 6)
plot(qhat1~time_ub, col="magenta", cex=0.2, lty=5, lwd=2, type="l", ylim=ylim,
    xlab="hour", ylab="PM10");
lines(qhat2~time_ub, col="aquamarine4", cex=0.2, lty=4, lwd=2);
lines(qhat3~time_ub, col="blue", cex=0.2, lty=3, lwd=3);
lines(qhat4~time_ub, col="brown", cex=0.2, lty=2, lwd=2);
lines(qhat5~time_ub, col="black", cex=0.2, lty=1, lwd=2);
lines(qhat6~time_ub, col="orange", cex=0.2, lty=2, lwd=2)
lines(qhat7~time_ub, col="darkcyan", cex=0.2, lty=3, lwd=3);
lines(qhat8~time_ub, col="green", cex=0.2, lty=4, lwd=2);
lines(qhat9~time_ub, 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), lty=c(5,4,3,2,1,2,3,4,5))
```

QRWSimul *Weighted simultaneous objective function*

Description

The etimation 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 smooting 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 τ_h-th conditional quantile curves.
Wtau	The weight of each order of quantile τ_h.

Note

Some warning messages are related to the function `rq.fit.sfn` (See http://www.inside-r.org/packages/cran/quantreg/docs/sfnMessage).

Author(s)

Yudhie Andriyana

References

Andriyana, Y., Gijbels, I., and Verhasselt, A. P-splines quantile regression estimation in varying coefficient models. *Test* 23, 1 (2014a),153–194.

Andriyana, Y., Gijbels, I. and Verhasselt, A. (2014b). Quantile regression in varying coefficient models: non-crossingness and heteroscedasticity. *Manuscript*.

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


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

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), lty=c(5,4,3,2,1,2,3,4,5))
```

Index