Package 'quantregGrowth'

July 22, 2025

Type Package

Title Non-Crossing Additive Regression Quantiles and Non-Parametric Growth Charts

Version 1.7-1

Date 2024-05-20

Maintainer Vito M. R. Muggeo <vito.muggeo@unipa.it>

Description Fits non-crossing regression quantiles as a function of linear covariates and multiple smooth terms, including varying coefficients, via B-splines with L1-norm difference penalties. Random intercepts and variable selection are allowed via the lasso penalties.
The smoothing parameters are estimated as part of the model fitting, see Muggeo and others (2021) <doi:10.1177/1471082X20929802>. Monotonicity and concavity constraints on the fitted curves are allowed, see Muggeo and others (2013) <doi:10.1007/s10651-012-0232-1>,
and also <doi:10.13140/RG.2.2.12924.85122> or <doi:10.13140/RG.2.2.29306.21445> some code examples.

Depends R (>= 3.5.0), quantreg, splines, SparseM, methods

License GPL

Suggests knitr, rmarkdown, mgcv, markdown

VignetteBuilder knitr

NeedsCompilation no

Author Vito M. R. Muggeo [aut, cre] (ORCID: <https://orcid.org/0000-0002-3386-4054>)

Repository CRAN

Date/Publication 2024-05-20 09:10:02 UTC

Contents

`quantregGrowth-package`

Non-Crossing Additive Regression Quantiles and Non-Parametric Growth Charts.

Description

Fits non-crossing regression quantiles as a function of linear covariates and smooth terms via P-splines with difference penalties. Random intercepts and selection of linear variables are allowed via the lasso penalties. Estimation of (possibly adaptive) smoothing/tuning parameters (for the spline terms, the random intercepts and the variables selector) are carried out efficientely as part of model fitting.

Details

Package:	quantregGrowth
Type:	Package
Version:	1.7-1
Date:	2024-05-20
License:	GPL

Package quantregGrowth allows estimation of growth charts via quantile regression. Given a set of percentiles (i.e. probability values), `gcrq` estimates non-crossing quantile curves as a flexible function of quantitative covariates (typically age in growth charts), and possibly additional linear terms. To ensure flexibility, B-splines with a difference L_1 penalty are employed to estimate non parametrically the curves wherein monotonicity and concavity constraints may be also set. Multiple smooth terms, including varying coefficients, are allowed and the amount of smoothness for each term is efficiently included in the model fitting algorithm, see Muggeo et al. (2021). `plot.gcrq` displays the fitted lines along with observations and poitwise confidence intervals.

Author(s)

Vito M.R. Muggeo

Maintainer: Vito M.R. Muggeo <vito.muggeo@unipa.it>

References

Muggeo VMR, Torretta F, Eilers PHC, Sciandra M, Attanasio M (2021). Multiple smoothing parameters selection in additive regression quantiles, *Statistical Modelling*, **21**, 428-448.

Muggeo VMR (2021). Additive Quantile regression with automatic smoothness selection: the R package quantregGrowth.
https://www.researchgate.net/publication/350844895

Muggeo VMR, Sciandra M, Tomasello A, Calvo S (2013). Estimating growth charts via nonparametric quantile regression: a practical framework with application in ecology, *Environ Ecol Stat*, **20**, 519-531.

Muggeo VMR (2018). Using the R package quantregGrowth: some examples.
https://www.researchgate.net/publication/323573492

Some references on growth charts (the first two papers employ the so-called LMS method)

Cole TJ, Green P (1992) Smoothing reference centile curves: the LMS method and penalized likelihood. *Statistics in Medicine* **11**, 1305-1319.

Rigby RA, Stasinopoulos DM (2004) Smooth centile curves for skew and kurtotic data modelled using the Box-Cox power exponential distribution. *Statistics in Medicine* **23**, 3053-3076.

Wei Y, Pere A, Koenker R, He X (2006) Quantile regression methods for reference growth charts. *Statistics in Medicine* **25**, 1369-1382.

Some references on regression quantiles

Koenker R (2005) Quantile regression. Cambridge University Press, Cambridge.

Cade BS, Noon BR (2003) A gentle introduction to quantile regression for ecologists. *Front Ecol Environ* **1**, 412-420.

See Also

gcrq, rq in package quantreg

Examples

```
#see ?gcrq for some examples
```

`charts` *Easy computing growth charts*

Description

Computes and returns quantiles as a function of the specified covariate values

Usage

```
charts(fit, k, file = NULL, digits=2, conf.level=0,
    dataframe=FALSE, transf=NULL, se.type=c("sandw","boot"), ...)
```

Arguments

fit	The object fit returned by gcrq
k	Numeric indicating the covariate values. If integer (and scalar, specified via 5L, say) , k equispaced values in the covariate range are taken.
file	If specified, the (path) file name wherein the returned matrix including the quantiles will be written via write.csv()
digits	Number of digits whereby the estimated quantiles are rounded.
conf.level	If larger than zero, the pointwise confidence intervals for the estimated quantiles are also returned. If conf.level=0 the simple point estimates.
dataframe	Logical. If TRUE and conf.level>0 a dataframe is returned having point estimate and confidence intervals collapsed. Otherwise a matrix having number of rows equal to the number of covariate values.
transf	An optional character string (with "y" as argument) meaning a function to apply to the predicted values. E.g. "(exp(y)-0.1)". If NULL (default) it is taken as the inverse of function transf (*if*) supplied in gcrq. The standard errors (provided se.fit=TRUE has been set) are adjusted accordingly via the Delta method. See argument "transf" in gcrq(). If transf has been specified in gcrq(), use transf="y" to force predictions on the transformed scale, i.e. without back transforming.
se.type	Which covariance matrix should be used, provided that conf.level>0. See type in predict.gcrq.
...	Further arguments passed on to write.csv()

Details

This function is simply a wrapper for predict.gcrq

Value

A matrix having number of columns equal to the number of quantile curves and number of rows depending k

Note

charts just works with models having a single smooth term. See predict.gcrq when the model involves multiple covariates.

Author(s)

Vito Muggeo

See Also

predict.gcrq

Examples

```
## Not run:
charts(_fit_, k=1L) #prediction at the minimum of covariate
charts(_fit_, k=1) #prediction at covariate value 1.

charts(_fit_, k=10L)

## End(Not run)
```

---

gcrq *Growth charts regression quantiles with automatic smoothness estimation*

Description

Modelling unspecified nonlinear relationships between covariates and quantiles of the response conditional distribution. Typical example is estimation nonparametric growth charts (via quantile regression). Quantile curves are estimated via B-splines with a L_1 penalty on the spline coefficient differences, while non-crossing and possible monotonicity and concavity restrictions are set to obtain estimates more biologically plausible. Linear terms can be specified in the model formula. Multiple smooth terms, including varying coefficients, with automatic selection of corresponding smoothing parameters are allowed.

Usage

```
gcrq(formula, tau=c(.1,.25,.5,.75,.9), data, subset, weights, na.action,
    transf=NULL, y=TRUE, n.boot=0, eps=0.001, display=FALSE,
   method=c("REML","ML"), df.opt=2, df.nc=FALSE, lambda0=.1, h=0.8, lambda.max=1000,
    tol=0.01, it.max=15, single.lambda=TRUE, foldid=NULL, nfolds=10,
    lambda.ridge=0, adjX.constr=TRUE, contrasts=NULL, sparse=FALSE)
```

Arguments

formula	a standard R formula to specify the response in the left hand side, and the covariates in the right hand side, such as y~ps(x)+z, see Details for further examples.
tau	a numeric vector to specify the quantile curves of interest. Default to probability values $(.1, .25, .5, .75, .9)$.
data	the dataframe where the variables required by the formula, subset and weights arguments are stored.
subset	optional. A vector specifying a subset of observations to be used in the fitting process.
weights	optional. A numeric vector specifying weights to be assigned to the observations in the fitting process. Currently unimplemented.
na.action	a function which indicates how the possible 'NA's are handled.

transf	an optional character string (with "y" as argument) meaning a function to apply to the response variable before fitting. It can be useful to guarantee fitted values within a specified range; e.g. if y>=0, we could set "log(y+0.1)". If provided, the resulting object fit refers to the model for the transformed response and it will include the corresponding inverse function (numerically computed) to be used to back transform predictions (see argument transf in predict.gcrq and plot.gcrq).
y	logical. If TRUE (default) the returned object includes also the responses vector.
n.boot	Number of nonparametric (cases resampling) bootstrap samples to be used. If n.boot>0, the covariance matrix can be obtained as empirical covariance matrix of the bootstrap distributions, see vcov.gcrq. Notice that the smoothing parameter (if relevant) is assumed fixed. Namely it does change throughout the bootstrap replicates.
eps	A small positive constant to ensure noncrossing curves (i.e. the minimum distance between two consecutive curves). Use it at your risk! If eps is large, the resulting fitted quantile curves could appear unreasonable.
display	Logical. Should the iterative process be printed? Ignored if no smooth is specified in the formula or if all the smoothing parameters specified in ps terms are fixed.
method	character, "ML" or "REML" affecting the smoothing parameter estimation. Default is "REML" which appears to provide better performance in simulation studies. Ignored if no smoothing parameter has to be estimated.
df.opt	How the model and term-specific degrees of freedom are computed. df.opt=1 means via the null penalized coefficients, and df.opt=2 via the trace of the approximate hat matrix. Ignored if no smoothing parameter is be estimated.
df.nc	logical. If TRUE and the model refers to multiple quantile curves, the degrees of freedom account for the noncrossing constraints. Ignored for single quantile fits. Default to FALSE, as it is still experimental.
lambda0	the starting value for the lambdas to be estimated. Ignored if all the smoothing parameters specified in ps terms are fixed.
h	The step halving factor affecting estimation of the smoothing parameters. Lower values lead to slower updates in the lambda values. Ignored if all the smoothing parameters specified in ps terms are fixed.
lambda.max	The upper bound for lambda estimation. Ignored if all the smoothing parameters specified in ps terms are fixed.
tol	The tolerance value to declare convergence. Ignored if all the smoothing parameters specified in ps terms are fixed.
it.max	The maximum number of iterations in lambdas estimation. Ignored if all the smoothing parameters specified in ps terms are fixed.
single.lambda	Logical. Should the smoothing parameter (for each smooth term) to be the same across the quantile curves being estimated? Ignored when just a single quantile curve is being estimated.
foldid	optional. A numeric vector identifying the group labels to perform cross validation to select the smoothing parameter. Ignored if the lambda argument in ps() is not a vector.

nfolds	optional. If foldid is not provided, it is scalar specifying the number of 'folds' (groups) which should be used to perform cross validation to select the smoothing parameter. Default to 10, but it is ignored if the lambda argument in ps() is not a vector.
lambda.ridge	Numerical value (typically very small) to stabilize model estimation.
adjX.constr	logical. If TRUE, each linear covariate is shifted (by substracting its min) in order to set up effective constraints to prevent crossing. Useful only if tau is a vector and the model includes linear terms.
contrasts	an optional list. See argument contrasts.arg in model.matrix.default.
sparse	logical. If TRUE, the model is fitted via sparse algebra as implemented in the SparseM package. Typically sparse=TRUE is used when the model involves a single smooth with a very rich basis and a large sample size, see Details.

Details

The function fits regression quantiles at specified percentiles given in tau as a function of covariates specified in the formula argument. The formula may include linear terms and one or several ps terms to model nonlinear relationships with quantitative covariates, usually age in growth charts. When the lambda argument in ps() is a negative scalar, the smoothing parameter is estimated iteratively as discussed in Muggeo et al. (2021). If a positive scalar, it represents the actual smoothing parameter value.

When the model includes a single ps term, setting sparse=TRUE (introduced since version 1.7-0) could reduce the computational time especially when the sample size is large and the basis involves several terms. However when sparse=TRUE is set, no linear term is allowed and for the smooth term a full and uncentred basis is used (i.e., equivalent to setting dropc=FALSE and center=FALSE along with constr.fit=FALSE in ps()). Therefore the correct call would be

```
gcrq(y ~ 0+ps(x), sparse=TRUE) #if y~ps(x), a warning is printed as the intercept is not explicitly removed
```

which is equivalent to

```
gcrq(y ~ 0+ps(x, dropc=FALSE, center=FALSE)) #sparse=FALSE is the default
```

Smoothing parameter selection via 'K-fold' cross validation (CV) is also allowed (but not recommended) if the model includes a single ps term: lambda should be a vector of candidate values, and the final fit is returned at the 'optimal' lambda value. To select the smoothing parameter via CV, foldid or nfolds may be supplied. If provided, foldid overwrites nfolds, otherwise foldid is obtained via random extraction, namely sample(rep(seq(nfolds), length = n)). However selection of smoothing parameter via CV is allowed only with a unique ps term in the formula.

Value

This function returns an object of class gcrq, that is a list with the following components (only the most important are listed)

coefficients	The matrix of estimated regression parameters; the number of columns equals the number of the fitted quantile curves.
x	the design matrix of the final fit (including the dummy rows used by penalty).

edf.j	a matrix reporting the edf values for each term at each quantile curve. See the section 'Warning' below.
rho	a vector including the values of the objective functions at the solution for each quantile curve.
fitted.values	a matrix of fitted quantiles (a column for each tau value)
residuals	a matrix of residuals (a column for each tau value)
D.matrix	the penalty matrix (multiplied by the smoothing parameter value).
D.matrix.nolambda	the penalty matrix.
pLin	number of linear covariates in the model.
info.smooth	some information on the smoothing term (if included in the formula via ps).
BB	further information on the smoothing term (if present in the formula via ps), including stuff useful for plotting via plot.gcrq().
Bderiv	if the smooth term is included, the first derivative of the B spline basis.
boot.coef	The array including the estimated coefficients at different bootstrap samples (provided that n.boot>0 has been set).
y	the response vector (if gcrq() has been called with y=TRUE).
contrasts	the contrasts used, when the model contains a factor.
xlevels	the levels of the factors (when included) used in fitting.
taus	a vector of values between 0 and 1 indicating the estimated quantile curves.
call	the matched call.

Warning

Variable selection is still at an experimental stage.

When including linear terms, the covariates X_j are shifted such that $min(X_j) = 0$.

The options 'REML' or 'ML' of the argument method, refer to how the degrees of freedom are computed to update the lambda estimates.

Currently, standard errors are obtained via the sandwich formula or the nonparametric bootstrap (case resampling). Both methods ignore uncertainty in the smoothing parameter selection.

Since version 1.2-1, computation of the approximate edf can account for the noncrossing constraints by specifying df.nc=TRUE. That could affect model estimation when the smoothing parameter(s) have to be estimated, because the term specific edf are used to update the lambda value(s). When lambda is not being estimated (it is fixed or there is no ps term in the formula), parameter estimate is independent of the df.nc value. The summary.gcrq method reports if the edf account for the noncrossing constraints.

Using ps(.., center=TRUE) in the formula leads to lower uncertainty in the fitted curve while guaranteeing noncrossing constraints.

Currently, decomposition of Bsplines (i.e. ps(..,decom=TRUE)) is incompatible with shape (monotonicity and concavity) restrictions and even with noncrossing constraints.

Note

This function is based upon the package quantreg by R. Koenker. Currently methods specific to the class "gcrq" are print.gcrq, summary.gcrq, vcov.gcrq, plot.gcrq, predict.gcrq, AIC.gcrq, and logLik.gcrq.

If the sample is not large, and/or the basis rank is large (i.e. a large number of columns) and/or there are relatively few distinct values in the covariate distribution, the fitting algorithm may fail returning error messages like the following

> Error info = 20 in stepy2: singular design

To remedy it, it suffices to change some arguments in ps(): to decrease ndx or deg (even by a small amount) or to increase (even by a small amount) the lambda value. Sometimes even by changing slightly the tau probability value (for instance from 0.80 to 0.79) can bypass the aforementioned errors.

Author(s)

Vito M. R. Muggeo, <vito.muggeo@unipa.it>

References

V.M.R. Muggeo, F. Torretta, P.H.C. Eilers, M. Sciandra, M. Attanasio (2021). Multiple smoothing parameters selection in additive regression quantiles, Statistical Modelling, 21: 428-448.

V. M. R. Muggeo (2021). Additive Quantile regression with automatic smoothness selection: the R package quantregGrowth. https://www.researchgate.net/publication/350844895

V. M. R. Muggeo, M. Sciandra, A. Tomasello, S. Calvo (2013). Estimating growth charts via nonparametric quantile regression: a practical framework with application in ecology, Environ Ecol Stat, 20, 519-531.

V. M. R. Muggeo (2018). Using the R package quantregGrowth: some examples. https://www.researchgate.net/publication/323573492

See Also

ps, plot.gcrq, predict.gcrq

Examples

```
## Not run:
##=== Example 1: an additive model from ?mgcv::gam

d<-mgcv::gamSim(n=200, eg=1)
o<-gcrq(y ~ ps(x0) + ps(x1)+ ps(x2) + ps(x3), data=d, tau=.5, n.boot=50)
plot(o, res=TRUE, col=2, conf.level=.9, shade=TRUE, split=TRUE)



##=== Example 2: some simple examples involving just a single smooth

data(growthData) #load data
tauss<-seq(.1,.9,by=.1) #fix the percentiles of interest
```

```
m1<-gcrq(y~ps(x), tau=tauss, data=growthData) #lambda estimated..

m2<-gcrq(y~ps(x, lambda=0), tau=tauss, data=growthData) #unpenalized.. very wiggly curves
#strongly penalized models
m3<-gcrq(y~ps(x, lambda=1000, d=2), tau=tauss, data=growthData) #linear
m4<-gcrq(y~ps(x, lambda=1000, d=3), tau=tauss, data=growthData) #quadratic

#penalized model with monotonicity restrictions
m5<-gcrq(y~ps(x, monotone=1, lambda=10), tau=tauss, data=growthData)

#monotonicity constraints,lambda estimated, and varying penalty
m6<-gcrq(y~ps(x, monotone=1, lambda=10, var.pen="(1:k)"), tau=tauss, data=growthData)
m6a<-gcrq(y~ps(x, monotone=1, lambda=10, var.pen="(1:k)^2"), tau=tauss, data=growthData)

par(mfrow=c(2,3))
plot(m1, res=TRUE, col=-1)
plot(m2, pch=20, res=TRUE)
plot(m3, add=TRUE, lty=2, col=4)
plot(m4, pch=20, res=TRUE)
plot(m5, pch=4, res=TRUE, legend=TRUE, col=2)
plot(m6, lwd=2, col=3)
plot(m6a, lwd=2, col=4)

#select lambda via 'K-fold' CV (only with a single smooth term)
m7<-gcrq(y~ps(x, lambda=seq(0,10,l=20)), tau=tauss, data=growthData)
par(mfrow=c(1,2))
plot(m7, cv=TRUE) #display CV score versus lambda values
plot(m7, res=TRUE, grid=list(x=5, y=8), col=4) #fit at the best lambda (by CV)


##=== Example 3: VC models

n=50
x<-1:n/n
mu0<-10+sin(2*pi*x)
mu1<- 7+4*x
y<-c(mu0,mu1)+rnorm(2*n)*.2 #small noise.. just to illustrate..
x<-c(x,x)
z<-rep(0:1, each=n)

# approach 1: a smooth in each *factor* level
g<-factor(z)
o <-gcrq(y~ g+ps(x,by=g), tau=.5)
predict(o, newdata=data.frame(x=c(.3,.7), g=factor(c(0,1))))
par(mfrow=c(1,2))
plot(x[1:50],mu0,type="l")
plot(o, term=1, add=TRUE)
points(c(.3,.7), predict(o, newdata=data.frame(x=c(.3,.7), g=factor(c(0,0)))), pch=4, lwd=3,col=2)

plot(x[1:50],mu1,type="l")
plot(o, term=2, add=T, shift=coef(o)["g1",], col=3) #note the argument 'shift'
points(c(.3,.7), predict(o, newdata=data.frame(x=c(.3,.7), g=factor(c(1,1)))), pch=4, lwd=3,col=3)
```

```
# approach 2: a general smooth plus the (smooth) 'interaction' with a continuous covariate..
b1 <-gcrq(y~ ps(x) + z+ ps(x,by=z), tau=.5)
par(mfrow=c(1,2))
plot(x[1:50],mu0,type="l")
plot(b1, add=TRUE, term=1) #plot the 1st smooth term

#plot the 2nd smooth of 'b1' (which is actually the difference mu1-mu0)
plot(x[1:50], mu1-mu0, type="l")
plot(b1, term=2, add=TRUE, interc=FALSE, shift=coef(b1)["z",])



##=== Example 4: random intercepts example

n=50
x<-1:n/n

set.seed(69)
z<- sample(1:15, size=n, replace=TRUE)
#table(z)

#true model: linear effect + 3 non-null coeffs when z= 3, 7, and 13
y<-2*x+  I(z==3)- I(z==7)  + 2*I(z==13) + rnorm(n)*.15
id<-factor(z)

o <-gcrq(y~x+ps(id), tau=.5)
plot(o, term=1) #plot the subject-specific intercepts


#== variable selection
n=50
p=30
p1<-10

X<-matrix(rnorm(n*p,5),n,p)
b<-rep(0,p)
id<-sample(1:p, p1)
b[id]<-round(runif(p1,.5,2),2)
b[id]<-b[id]* sign(ifelse(runif(p1)<.5,1,-1))
lp <- drop(tcrossprod(X,t(b)))
y <- 2+lp+rnorm(n)*1.5

gcrq(y~ps(X), tau=.7)

## End(Not run)
```

---

growthData *Simulated data to illustrate capabilities of the package*

Description

The growthData data frame has 200 rows and 3 columns.

Usage

```
data(growthData)
```

Format

A data frame with 200 observations on the following 3 variables.

- x the supposed 'age' variable.
- y the supposed growth variable (e.g. weight).
- z an additional variable to be considered in the model.

Details

Simulated data to illustrate capabilities of the package.

Examples

```
data(growthData)
with(growthData, plot(x,y))
```

---

logLik.gcrq *Log Likelihood, AIC and BIC for gcrq objects*

Description

The function returns the log-likelihood value(s) evaluated at the estimated coefficients

Usage

```
## S3 method for class 'gcrq'
logLik(object, summ=TRUE, ...)
## S3 method for class 'gcrq'
AIC(object, ..., k=2, bondell=FALSE)
```

Arguments

object	A gcrq fit returned by gcrq()
summ	If TRUE, the log likelihood values (and relevant edf) are summed over the different taus to provide a unique value accounting for the different quantile curves. If FALSE, tau-specific values are returned.
k	Optional numeric specifying the penalty of the edf in the AIC formula. k < 0 means k=log(n).

bondell	Logical. If TRUE, the *SIC* according to formula (7) in Bondell et al. (2010) is computed.
...	optional arguments (nothing in logLik.gcrq). For AIC.gcrq, summ=TRUE or FALSE can be set.

Details

The 'logLikelihood' is computed by assuming an asymmetric Laplace distribution for the response as in logLik.rq, namely $n(\log(\tau(1-\tau)) - 1 - \log(\rho_\tau/n))$, where ρ_τ is the minimized objective function. When there are multiple quantile curves $j = 1, 2, ..., J$ (and summ=TRUE) the formula is

$n(\sum_j \log(\tau_j(1-\tau_j)) - J - \log(\sum_j \rho_{\tau_j}/(nJ)))$

AIC.gcrq simply returns -2*logLik + k*edf where k is 2 or log(n).

Value

The log likelihood(s) of the model fit object

Author(s)

Vito Muggeo

References

Bondell HD, Reich BJ, Wang H (2010) Non-crossing quantile regression curve estimation, Biometrika, 97: 825-838.

See Also

logLik.rq

Examples

```
## logLik(o) #a unique value (o is the fit object  from gcrq)
## logLik(o, summ=FALSE) #vector of the log likelihood values
## AIC(o, k=-1) #BIC
```

---

ncross.rq.fitXB *Estimation of noncrossing regression quantiles with monotonicity restrictions.*

Description

These are internal functions of package quantregGrowth and should be not called by the user.

Usage

```
ncross.rq.fitXB(y, x, B=NULL, X=NULL, taus, monotone=FALSE, concave=FALSE,
    nomiBy=NULL, byVariabili=NULL, ndx=10, deg=3, dif=3, lambda=0, eps=.0001,
    var.pen=NULL, penMatrix=NULL, lambda.ridge=0, dropcList=FALSE,
    decomList=FALSE, vcList=FALSE, dropvcList=FALSE, centerList=FALSE,
    ridgeList=FALSE, ps.matrix.list=FALSE, colmeansB=NULL, Bconstr=NULL,
    adjX.constr=TRUE, adList=FALSE, it.j=10, myeps=NULL, ...)

ncross.rq.fitXBsparse(y, x, B=NULL, X=NULL, taus, monotone=FALSE, concave=FALSE,
    nomiBy=NULL, byVariabili=NULL, ndx=10, deg=3, dif=3, lambda=0, eps=.0001,
   var.pen=NULL, penMatrix=NULL, lambda.ridge=0, dropcList=FALSE, decomList=FALSE,
    vcList=FALSE, dropvcList=FALSE, centerList=FALSE, ridgeList=FALSE,
    ps.matrix.list=FALSE, colmeansB=NULL, Bconstr=NULL, adjX.constr=TRUE,
    adList=FALSE, it.j=10, myeps=NULL, ...)


ncross.rq.fitX(y, X = NULL, taus, adjX.constr=TRUE, lambda.ridge = 0,
    eps = 1e-04, ...)


gcrq.rq.cv(y, B, X, taus, monotone, concave, ndx, lambda, deg, dif, var.pen=NULL,
    penMatrix=NULL, lambda.ridge=0, dropcList=FALSE, decomList=FALSE,
    vcList=vcList, dropvcList=FALSE, nfolds=10, foldid=NULL, eps=.0001,
    sparse=FALSE, ...)
```

Arguments

y	the responses vector. see gcrq
x	the covariate supposed to have a nonlinear relationship.
B	the B-spline basis.
X	the design matrix for the linear parameters.
taus	the percentiles of interest.
monotone	numerical value (-1/0/+1) to define a non-increasing, unconstrained, and non-decreasing flexible fit, respectively.
concave	numerical value (-1/0/+1) to possibly define concave or convex fits.
nomiBy	useful for VC models (when B is not provided).
byVariabili	useful for VC models (when B is not provided).
ndx	number of internal intervals within the covariate range, see ndx in ps.
deg	spline degree, see ps.
dif	difference order of the spline coefficients in the penalty term.
lambda	smoothing parameter value(s), see lambda in ps.
eps	tolerance value.
var.pen	Varying penalty, see ps.
penMatrix	Specified penalty matrix, see pen.matrix in ps.

lambda.ridge	a (typically very small) value, see lambda.ridge gcrq.
dropcList	see dropc in ps.
decomList	see decompose in ps.
vcList	to indicate if the smooth is VC or not, see by in ps.
dropvcList	see ps.
centerList	see center in ps.
ridgeList	see ridge in ps.
ps.matrix.list	nothing relevant for the user.
colmeansB	see center in ps.
Bconstr	see constr.fit in ps.
foldid	vector (optional) to perform cross validation, see the same arguments in gcrq.
nfolds	number of folds for crossvalidation, see the same arguments in gcrq.
cv	returning cv scores; see the same arguments in gcrq.
adjX.constr	logical to shift the linear covariates. Appropriate only with linear terms.
adList	see ad in ps.
it.j	Ignore.
myeps	Ignore.
sparse	logical, meaning if sparse computations have to be used.
...	optional.

Details

These functions are called by gcrq to fit growth charts based on regression quantiles with non-crossing and monotonicity restrictions. The computational methods are based on the package quantreg by R. Koenker and details are described in the reference paper.

Value

A list of fit information.

Author(s)

Vito M. R. Muggeo

See Also

gcrq

Examples

```
##See ?gcrq
```

plot.gcrq *Plot method for gcrq objects*

Description

Displaying the estimated growth charts from a gcrq fit.

Usage

```
## S3 method for class 'gcrq'
plot(x, term=NULL, add = FALSE, res = FALSE, conf.level=0, axis.tau=FALSE,
   interc=TRUE, se.interc=FALSE, legend = FALSE, select.tau, deriv = FALSE,
   cv = FALSE, transf=NULL, lambda0=FALSE, shade=FALSE, overlap=NULL, rug=FALSE,
  overall.eff=TRUE, grid=NULL, smoos=NULL, split=FALSE, shift=0, type=c("sandw","boot"),
   n.points=NULL, ...)
```

Arguments

x	a fitted "gcrq" object.
term	the variable name or its index in the formula entering the model. Can be vector. Both linear ad spline terms (i.e. included in the model via ps) can be specified and relevant fitted quantile curves (as optionally specified by select.tau) will be plotted. If the model includes both linear and smooth terms, the smooth terms are counted and drawn *first*: therefore if the model formula is y~z+ps(x), term=1 refers to the smooth term. If NULL, all smooth terms are plotted according to the split argument. If the model includes multiple quantile curves and axis.tau=TRUE, term=1 refers to the model intercept (if in the model). The variable name should be reported within 'ps()', e.g. 'ps(age)' regardless of additional arguments specified in ps.
interc	Should the smooth term be plotted along with the model intercept (provided it is included in the model)? Of course such argument is ignored if the smooth term has been called via ps(, dropc=FALSE) and the plot always includes implicitly the 'intercept'. Note that interc=TRUE is requested to display the noncrossing curves (if multiple quantile curves are being plotted).
se.interc	logical. If TRUE the standard errors of fitted quantile curves account for uncertainty of the model intercept (provided it is included in the model). If FALSE, then the uncertainty relates purely to the (usually centred) smooth itself. Ignored if conf.level=0.
add	logical. If TRUE the fitted quantile curves are added on the current plot.
res	logical. If TRUE 'partial residuals' are also displayed on the plot. Borrowing terminology from GLM, partial residuals for covariate X_j are defined as fitted values corresponding to X_j + residuals (from the actual fit). If there is a single covariate, the partial residuals correspond to observed data. If multiple quantile curves have been estimated, the fitted values coming from the 'middle' quantile curve are employed to compute the partial residuals. 'Middle'

means 'corresponding to the τ_k closest to 0.50'. I don't know if that is the best choice.

`conf.level` logical. If larger than zero, pointwise confidence intervals for the fitted quantile curve are also shown (at the confidence level specified by `conf.level`). Such confidence intervals are independent of the possible intercept accounted for via the `intercept` argument. See `type` to select different methods (bootstrap or sandwich) to compute the standard errors.

`axis.tau` logical. If `TRUE`, the estimated coefficient `term` is plotted against the probability values. This graph could be useful if the model has been estimated at several `tau` values.

`legend` logical. If `TRUE` a legend is drawn on on the right side of the plot.

`select.tau` an optional numeric vector to draw only some of the fitted quantiles. Percentile values or integers 1 to `length(tau)` may be supplied.

`deriv` logical. If `TRUE` the first derivative of the fitted curves are displayed.

`cv` logical. If `TRUE` and the `"gcrq"` object contains a single smooth term wherein lambda has been selected via CV, then the cross-validation scores against the lambda values are plotted.

`transf` An optional character string (with `"y"` as argument) meaning a function to apply to the predicted values (and possibly residuals) before plotting. E.g. `"(exp(y)-0.1)"`. If `NULL` (default) it is taken as the inverse of function `transf` (*if*) supplied in `gcrq`. See argument `"transf"` in `gcrq()`. If `transf` has been specified in `gcrq()`, use `transf="y"` to force plotting on the transformed scale, i.e. without back transforming.

`lambda0` logical. If `cv=TRUE`, should the CV plot include also the first CV value? Usually the first CV value is at lambda=0, and typically it is much bigger than the other values making the plot not easy to read. Default to `FALSE` not to display the first CV value in the plot.

`shade` logical. If `TRUE` and `conf.level`>0, the pointwise confidence intervals are portrayed via shaded areas.

`overlap` NULL or numeric (scalar or vector). If provided and different from `NULL`, it represents the abscissa values (on the covariate scale) where the legends (i.e. the probability values) of each curve are set. It will be recycled, if its length differs from the number of quantile curves. If unspecified (i.e. `overlap=NULL`), the legends are placed outside the fitted lines on the right side. If specified, `legend=TRUE` is implicitly assumed.

`rug` logical. If `TRUE`, the covariate distribution is displayed as a rug plot at the foot of the plot. Default to `FALSE`.

`overall.eff` logical. If the smooth term has been called via `ps(.., decom=TRUE)`, by specifying `overall.eff=TRUE` the overall smooth effect is drawn, otherwise only the penalized part is portrayed (always *without* intercept).

`grid` if provided, a grid of horizontal and vertical lines is drawn. `grid` has to be a list with the following components `x,y,col,lty,lwd`. If `x` (`y`) is a vector, the vertical (horizontal) lines are drawn at these locations. If `x` (`y`) is a scalar, the vertical (horizontal) lines are drawn at `x` (`y`) equispaced values. `col, lty,lwd` refer to the lines to be drawn.

smoos	logical, indicating if the residuals (provided that res=TRUE) will be drawn using a *smoothed* scatterplot. If NULL (default) the smoothed scatterplot will be employed when the number of observation is larger than 10000.
split	logical. If there are multiple terms (both smooth and linear) and split=TRUE, plot.gcrq() tries to split the plotting area in 2 columns and number of rows depending on the number of smooths. If split=FALSE, the plots are produced on the current device according to the current graphics settings. Ignored if there is single smooth term.
shift	Numerical value(s) to be added to the curve(s) to be plotted. If vector with length equal to the number of quantile curves to plot, the shift[j] is added to the jth quantile curve.
type	If conf.level>0, which covariance matrix should be used to compute and to portray the pointwise confidence intervals? 'boot' means case-resampling bootstrap (see n.boot in gcrq(), 'sandw' mean via the sandwich formula.
n.points	On how many values the plotted lines should rely on? If NULL, 100 values which suffice most of times. Increasing n.points can be useful when adaptive smoothing is used.
...	Additional graphical parameters: xlab, ylab, ylim, and xlim (effective when add=FALSE); lwd, lty, and col for the fitted quantile lines; col<0 means color palette for the different curves; cex and text.col for the legend (if legend=TRUE or overlap is specified); cex.p, col.p, and pch.p for the points (if res=TRUE). When axis.tau=TRUE, all arguments accepted by plot(), points(), matplot(), and matpoints() but pch, type, xlab, ylab, lty.

Details

Takes a "gcrq" object and diplays the fitted quantile curves as a function of the covariate specified in term. If conf.level>0 pointwise confidence intervals are also displayed. When the object contains the component cv, plot.gcrq can display cross-validation scores against the lambda values, see argument cv. If a single quantile curve is being displayed, the default 'ylab' includes the relevant edf value (leaving out the basis intercept). If axis.tau=TRUE and the fit includes several quantile curves, plot.gcrq() portrays the estimated coefficients versus the probability values. If term refers to a categorical variable, the point estimates against the categories are plotted (conf.level is ignored).

Value

The function simply generates a new plot or adds fitted curves to an existing one.

Note

Plotting non-crossing curves could depend on the arguments 'interc' and 'shift', in turn depending on how the model has been specified. Take care about that!

Author(s)

Vito M. R. Muggeo

See Also

gcrq, predict.gcrq

Examples

```
## Not run:
## use the fits from ?gcrq
#The additive model
plot(o, res=TRUE, col=2, conf.level=.9, shade=TRUE, split=TRUE)


par(mfrow=c(2,2))
plot(m5, select.tau=c(.1,.5,.9), overlap=0.6, legend=TRUE)
plot(m5, grid=list(x=8,y=5), lty=1) #a 8 times 5 grid..
plot(m7, cv=TRUE) #display CV score versus lambda values
plot(m7, res=TRUE, grid=list(x=5, y=8), col=4) #fitted curves at the best lambda value


## End(Not run)
```

---

predict.gcrq *Prediction for "gcrq" objects*

Description

Takes a "gcrq" objects and computes fitted values

Usage

```
## S3 method for class 'gcrq'
predict(object, newdata, se.fit=FALSE, transf=NULL, xreg,
                      type=c("sandw","boot"), ...)
```

Arguments

object	a fitted "gcrq" object.
newdata	a dataframe including *all* the covariates of the model. The smooth term is represented by a covariate and proper basis functions will be build accordingly. If omitted, the fitted values are used. Ignored if xreg is provided.
se.fit	logical. If TRUE, standard errors of the fitted quantiles are computed using the bootstrap or the sandwich covariance matrix, according to the argument type.
transf	An optional character string (with "y" as argument) meaning a function to apply to the predicted values. E.g. "(exp(y)-0.1)". If NULL (default) it is taken as the inverse of function transf (*if*) supplied in gcrq. The standard errors (provided se.fit=TRUE has been set) are adjusted accordingly via the Delta method. See argument "transf" in gcrq(). If transf has been specified in gcrq(), use transf="y" to force predictions on the transformed scale, i.e. without back transforming.

xreg	the design matrix for which predictions are requested. If provided, xreg has to include the basis functions of the B-spline.
type	If se.fit=TRUE, which cov matrix should be used? 'boot' means case-resampling bootstrap (see n.boot in gcrq()), 'sandw' mean via the sandwich formula.
...	arguments passed to other functions

Details

predict.gcrq computes fitted quantiles as a function of observations included in newdata or xreg. Either newdata or xreg have to be supplied, but newdata is ignored when xreg is provided.

Value

If se.fit=FALSE, a matrix of fitted values with number of rows equal to number of rows of input data and number of columns depending on the number of fitted quantile curves (i.e length of taus). If se.fit=TRUE, a list of matrices (fitted values and standard errors).

Author(s)

Vito M.R. Muggeo

See Also

gcrq, plot.gcrq

Examples

```
##see ?gcrq
## predict(m1, newdata=data.frame(x=c(.3,.7)))
```

---

print.gcrq *Print method for the gcrq class*

Description

Printing the most important feautures of a gcrq model.

Usage

```
## S3 method for class 'gcrq'
print(x, digits = max(3, getOption("digits") - 4), ...)
```

Arguments

x	object of class gcrq
digits	number of digits to be printed
...	arguments passed to other functions

Author(s)

Vito M.R. Muggeo

See Also

summary.gcrq

---

ps *Specifying a smooth term in the gcrq formula.*

Description

Function used to define the smooth term (via P-splines) within the gcrq formula. The function actually does not evaluate a (spline) smooth, but simply it passes relevant information to proper fitter functions.

Usage

```
ps(..., lambda = -1, d = 3, by=NULL, ndx = NULL, deg = 3, knots=NULL,
    monotone = 0, concave = 0, var.pen = NULL, pen.matrix=NULL, dropc=TRUE,
    center=TRUE, K=NULL, decom=FALSE, constr.fit=TRUE, shared.pen=FALSE,
    st=FALSE, ad=0)
```

Arguments

...	The covariate supposed to have a nonlinear relationship with the quantile curve(s) being estimated. A B-spline is built, and a (difference) penalty is applied. In growth charts this variable is typically the age. If the covariate is a factor, category-specific coefficients are estimated subject to a lasso penalty. See the last example in ?gcrq. A matrix of (continuous) covariates can be also supplied to perfom variable selection (among its columns).
lambda	A supplied smoothing parameter for the smooth term. If it is negative scalar, the smoothing parameter is estimated iteratively as discussed in Muggeo et al. (2021). If a positive scalar, it represents the actual smoothing parameter. If it is a vector, cross validation is performed to select the 'best' value. See Details in gcrq.
d	The difference order of the penalty. Default to 3 Ignored if pen.matrix is supplied.
by	if different from NULL, a numeric or factor variable of the same dimension as the covariate in ... If numeric the elements multiply the smooth (i.e. a varying coefficient model); if factor, a smooth is fitted for each factor level. Usually the variable by is also included as main effect in the formula, see examples in gcrq. When by includes a factor, the formula should include the model intecept, i.e. y~g+ps(x,by=g) and not y~ 0+g+ps(x,by=g).

ndx — The number of intervals of the covariate range used to build the B-spline basis. Non-integer values are rounded by `round()`. If `NULL`, default, it is taken $min(n/4, 9)$ (versions <=1.1-0 it was $min(n/4, 40)$, the empirical rule of Ruppert). It could be reduced further (but no less than 5 or 6, say) if the sample size is not large and the default value leads to some error in the fitting procedure, see section `Note` in `gcrq`. Likewise, if the underlying relationship is strongly nonlinear, `ndx` could be increased. The returned basis wil have 'ndx+deg-1' (if `dropc=TRUE`) basis functions.

deg — The degree of the spline polynomial. Default to 3. The B-spline basis is composed by `ndx+deg` basis functions and if `dropc=TRUE` the first column is removed for identifiability (and the model intercept is estimated without any penalty).

knots — The knots locations. If `NULL`, equispaced knots are set. Note if predictions outside the observed covariate range have to be computed (via `predict.gcrq`), the knots should be set enought outside the observed range.

monotone — Numeric value to set up monotonicity restrictions on the first derivative of fitted smooth function

- '0' = no constraint (default);
- '1' = non-decreasing smooth function;
- '-1' = non-increasing smooth function.

concave — Numeric value to set up monotonicity restrictions on the second derivative of fitted smooth function

- '0' = no constraint (default);
- '1' = concave smooth function;
- '-1' = convex smooth function.

var.pen — A character indicating the varying penalty. See Details.

pen.matrix — if provided, a penalty matrix A, say, such that the penalty in the objective function, apart from the smoothing parameter, is $||Ab||_1$ where b is the spline coefficient vector being penalized.

dropc — logical. Should the first column of the B-spline basis be dropped for the basis identifiability? Default to `TRUE`. Note, if `dropc=FALSE` is set, it is necessary to omit the model intercept AND not to center the basis, i.e. `center=FALSE`. Alternatively, both a full basis and the model intercept may be included by adding a small ridge penalty via `lambda.ridge>0`.

center — logical. If `TRUE` the smooth effects are 'centered' over the covariate values, i.e. $\sum_i \hat{f}(x_i) = 0$.

K — A scalar tuning the selection of wiggliness of the smoothed curve when λ has to be estimated (i.e. `lambda<0` is set). The larger `K`, the smoother the curve. Simulations suggest `K=2` for the smoothing, and `K=log(n/p^(2/3))` for variable selection and random intercepts (`p` is the number of variables or number of subjects). See details.

decom — logical. If `TRUE`, the B-spline B (with a d order difference penalty) is decomposed into truncated power functions namely unpenalized polynomial terms up to degree d-1, and additional terms $Z = BD'(DD')^{-1}$. Only the coefficients of Z are penalized via an identity matrix, i.e. a lasso penalty. Currently `decom=TRUE` does not work with shape (monotonicity and concavity) restrictions and noncrossing constraints.

`constr.fit` logical. If `monotone` or `concave` are different from 0, `constr.fit=TRUE` means that these constraints are set on the fitted quantiles rather than on the spline coefficients.

`shared.pen` logical. If `TRUE` and the smooth is a VC term with a factor specified in `by`, the smooths in each level of the factor share the same smoothing parameter.

`st` logical. If `TRUE` the variable(s) are standardized via the `scale()` function. Typically used for variable selection via lasso, i.e. when a matrix of covariates is passed in `ps()`.

`ad` a positive number to carry out a form of *adaptive* lasso. More specifically, at each step of the iterative algorithm, the penalty is $\lambda \sum_j w_j |\beta_j|$ where $w_j = |\tilde{\beta}_j|^{-\texttt{ad}}$ and $\tilde{\beta}_j$ are estimates coming from the previous iteration with a different value of λ. `ad=0` means the standard lasso and `ad=1` adaptive lasso (with weights updated during the iterative process.

Details

If a numeric variable has been supplied, `ps()` builds a B-spline basis with number of columns equal to `ndx+deg` (or `length(knots)-deg-1`). However, unless `dropc=FALSE` is specified, the first column is removed for identifiability, and the spline coefficients are penalized via differences of order `d`; `d=0` leads to a penalty on the coefficients themselves. If `pen.matrix` is supplied, `d` is ignored. Since versions 1.5-0 and 1.6-0, a factor or matrix can be supplied.

`lambda` is the tuning parameter, fixed or to be estimated. When `lambda=0` an unpenalized (and typically wiggly) fit is obtained, and as lambda increases the curve gets smoother till a `d-1` degree polynomial. At 'intermediate' lambda values, the fitted curve is a *piecewise* polynomial of degree `d-1`.

It is also possible to put a varying penalty via the argument `var.pen`. Namely for a constant smoothing (`var.pen=NULL`) the penalty is $\lambda \sum_k |\Delta_k^d|$ where Δ_k^d is the k-th difference (of order `d`) of the spline coefficients. For instance if $d = 1$, $|\Delta_k^1| = |b_k - b_{k-1}|$ where the b_k are the spline coefficients. When a varying penalty is set, the penalty becomes $\lambda \sum_k |\Delta_k^d| w_k$ where the weights w_k depend on `var.pen`; for instance `var.pen="((1:k)^2)"` results in $w_k = k^2$. See models `m6` and `m6a` in the examples of `gcrq`.

If `decom=TRUE`, the smooth can be plotted with or without the fixed part, see `overall.eff` in the function `plot.gcrq`.

Value

The function simply returns the covariate with added attributes relevant to smooth term.

Note

For shape-constrained fits, use `constr.fit=FALSE` only if you are using a single full and uncentred basis, namely something like
`gcrq(y~0+ps(x, center=FALSE, dropc=FALSE, monotone=1, constr.fit=FALSE),..)`.

Author(s)

Vito M. R. Muggeo

References

Muggeo VMR, Torretta F, Eilers PHC, Sciandra M, Attanasio M (2021). Multiple smoothing parameters selection in additive regression quantiles, *Statistical Modelling*, 21, 428-448.

For a general discussion on using B-spline and penalties in regression model see

Eilers PHC, Marx BD. (1996) Flexible smoothing with B-splines and penalties. Statistical Sciences, 11:89-121.

See Also

gcrq, plot.gcrq

Examples

```
##see ?gcrq

##gcrq(y ~ ps(x),..) #it works (default: center = TRUE, dropc = TRUE)
##gcrq(y ~ 0 + ps(x, center = TRUE, dropc = FALSE)) #it does NOT work
##gcrq(y ~ 0 + ps(x, center = FALSE, dropc = FALSE)) #it works
```

SiChildren *Age, height and weight in a sample of Italian children*

Description

Age, height and weight in a sample of 1424 Italian children born in Sicily in the eighties

Usage

```
data("SiChildren")
```

Format

A data frame with 1424 observations on the following 3 variables.

`age` age in years

`height` child height (in centimeter)

`weight` child weight (in kilo)

Details

Data refer on the usual antropometric measures of Italian boys born in Sicily in the first years of 80s. Data have been kindly provided by prof M. Chiodi

Source

Gattuccio F., and Pirronello S., and Chiodi M (1988) Possibilita' di identificazione di tipologie evolutive del periodo puberale: proposta di una metodica pr finalita' predittive, *Rivista di pediatria preventiva e sociale nipiologia*, 189-199

Examples

```
data(SiChildren)
## see the package vignette for an example using such dataset
```

`summary.gcrq` *Summarizing model fits for growth charts regression quantiles*

Description

summary and print methods for class `gcrq`

Usage

```
## S3 method for class 'gcrq'
summary(object, type=c("sandw","boot"), digits = max(3, getOption("digits") - 3),
    signif.stars =getOption("show.signif.stars"), ...)
```

Arguments

`object`	An object of class "`gcrq`".
`type`	Which covariance matrix should be used to compute the estimate standard errors? '`boot`' means case-resampling bootstrap (see `n.boot` in `gcrq()`), '`sandw`' mean via the sandwich formula.
`digits`	controls number of digits printed in output.
`signif.stars`	Should significance stars be printed?
`...`	further arguments.

Details

`summary.gcrq` returns some information on the fitted quantile curve at different probability values, such as the estimates, standard errors, values of check (objective) function values at solution. Currently there is no `print.summary.gcrq` method, so `summary.gcrq` itself prints results.

The SIC returned by `print.gcrq` and `summary.gcrq` is computed as $\log(\rho_\tau/n)+\log(n)edf/(2n)$, where ρ_tau is the usual asymmetric sum of residuals (in absolute value). For multiple J quantiles it is $\log(\sum_\tau \rho_\tau/(nJ)) + \log(nJ)edf/(2nJ)$. Note that computation of SIC in `AIC.gcrq` relies on the Laplace assumption for the response.

Author(s)

Vito M.R. Muggeo

See Also

gcrq

Examples

```
## see ?gcrq
##summary(o)
```

---

vcov.gcrq *Variance-Covariance Matrix for a Fitted 'gcrq' Model*

Description

Returns the variance-covariance matrix of the parameter estimates of a fitted gcrq model object.

Usage

```
## S3 method for class 'gcrq'
vcov(object, term, type=c("sandw","boot"), ...)
```

Arguments

object	a fitted model object of class "gcrq" returned by gcrq().
term	if specified, the returned covariance matrix includes entries relevant to parameter estimates for that 'term' only. If missing, the returned matrices refer to all model parameter estimates. Currently term is not allowed.
type	Which cov matrix should be returned? 'boot' means case-resampling bootstrap (see n.boot in gcrq()), 'sandw' mean via the sandwich formula.
...	additional arguments.

Details

Bootstrap-based covariance matrix, i.e. type="boot", is computable only if the object fit has been obtained by specifying n.boot>0 in gcrq().

Value

A list (its length equal the length of tau specified in gcrq) of square matrices. Namely the list includes the covariance matrices of the parameter estimates for each regression quantile curve.

Author(s)

Vito Muggeo

See Also

summary.gcrq

Index