Package ‘qrNLMM’

May 21, 2025

Type Package

Title Quantile Regression for Nonlinear Mixed-Effects Models

Version 4.0

Date 2025-05-21

Maintainer Christian E. Galarza <`cgalarza88@gmail.com`>

Imports mvtnorm, lqr, quantreg, psych, ald, progress

Description Quantile regression (QR) for Nonlinear
Mixed-Effects Models via the asymmetric Laplace distribution (ALD).
It uses the Stochastic Approximation of the EM (SAEM) algorithm for
deriving exact maximum likelihood estimates and full inference result
is
for the fixed-effects and variance components.
It also provides prediction and graphical summaries for assessing the algorithm
convergence and fitting results.

License GPL (>= 2)

NeedsCompilation no

Author Christian E. Galarza [aut, cre],
Victor H. Lachos [aut]

Repository CRAN

Date/Publication 2025-05-21 18:10:02 UTC

Contents

`group.plots` *Plot function for grouped data*

Description

Functions for plotting a profiles plot for grouped data.

Usage

```
group.plot(x,y,groups,...)
group.lines(x,y,groups,...)
group.points(x,y,groups,...)
```

Arguments

`y`	the response vector of dimension N where N is the total of observations.
`x`	vector of longitudinal (repeated measures) covariate of dimension N. For example: Time, location, etc.
`groups`	factor of dimension N specifying the partitions of the data over which the random effects vary.
`...`	additional graphical arguments passed to `matplot`. See `par`.

Author(s)

Christian E. Galarza <<`chedgala@espol.edu.ec`>> and Victor H. Lachos <<`hlachos@uconn.edu`>>

See Also

`Soybean`, `HIV`, `QRNLMM`

Examples

```
## Not run:
#A full profile plot for Soybean data

data(Soybean)
attach(Soybean)

group.plot(x = Time,y = weight,groups = Plot,type="b",
             main="Soybean profiles",xlab="time (days)",
             ylab="mean leaf weight (gr)")

#Profile plot by genotype

group.plot(x = Time[Variety=="P"],y = weight[Variety=="P"],
             groups = Plot[Variety=="P"],type="l",col="blue",
             main="Soybean profiles by genotype",xlab="time (days)",
             ylab="mean leaf weight (gr)")
```

```
group.lines(x = Time[Variety=="F"],y = weight[Variety=="F"],
            groups = Plot[Variety=="F"],col="black")

## End(Not run)
```

`HIV` *HIV viral load study*

Description

The data set belongs to a clinical trial (ACTG 315) studied in previous researches by Wu (2002) and Lachos et al. (2013). In this study, we analyze the HIV viral load of 46 HIV-1 infected patients under antiretroviral treatment (protease inhibitor and reverse transcriptase inhibitor drugs). The viral load and some other covariates were mesured several times days after the start of treatment been 4 and 10 the minimum and maximum number of measures per patient respectively.

Usage

```
data(HIV)
```

Format

This data frame contains the following columns:

`patid` a numeric vector indicating the patient register number.

`ind` a numeric vector indicating the number patient on which the measurement was made. It represents the subject number in the study.

`day` time in days.

`cd4` cd4 count in cells/mm^3.

`lgviral` viral load in log10 scale.

`cd8` cd8 count in cells/mm^3.

Details

In order to fit the nonlinear data we sugest to use the Nonlinear model proposed by Wu (2002) and also used by Lachos et al. (2013).

Source

Wu, L. (2002). A joint model for nonlinear mixed-effects models with censoring and covariates measured with error, with application to aids studies. Journal of the American Statistical association, 97(460), 955-964.

Lachos, V. H., Castro, L. M. & Dey, D. K. (2013). Bayesian inference in nonlinear mixed-effects models using normal independent distributions. Computational Statistics & Data Analysis, 64, 237-252.

`predict.QRNLMM` *Predict method for Quantile Regression for Nonlinear Mixed-Effects (QRNLMM) fits*

Description

Takes a fitted object produced by `QRNLMM()` and produces predictions given a new set of values for the model covariates.

Usage

```
## S3 method for class 'QRNLMM'
predict(object,x = NULL,groups = NULL,covar = NULL, y = NULL,MC = 1000,...)
```

Arguments

`object`	a fitted QRNLMM object as produced by QRNLMM().
`x`	vector of longitudinal (repeated measures) covariate of dimension N. For example: Time, location, etc.
`groups`	factor of dimension N specifying the partitions of the data over which the random effects vary.
`covar`	a matrix of dimension $N \times r$ where r represents the number of covariates.
`y`	the response vector of dimension N where N is the total of observations. Optional. See details.
`MC`	number of MC replicates for the computation of new individual values (only when `y` is provided). By default `MC = 1000`. See details.
`...`	additional arguments affecting the predictions produced.

Details

Prediction for QRNLMM objects can be performed under three different scenarios:

1. `predict(object)`: if no newdata is provided, fitted values for the original dataset is returned. Please refer to the `fitted.values` value in `QRNLMM`.
2. `predict(object,x,groups,covar = NULL)`: if new data is provided, but only the independent variables (no response), population curves are provided. If no covariates are provided, the predicted curves will be the same.
3. `predict(object,x,groups,covar = NULL,y)` if the response values are provided, a Metropolis-Hastings algorithm (with `MC` replicates and `thin = 5`) is performed in order to compute the random-effects for new subjects. The method is based on Galarza et.al. (2020).

Value

A `data.frame` containing the predicted values, one column per quantile.

Note

For scenario 3, results may vary a little each time. For more precision, please increase `MC`.

Author(s)

Christian E. Galarza <<`chedgala@espol.edu.ec`>> and Victor H. Lachos <<`hlachos@uconn.edu`>>

References

Galarza, C.E., Castro, L.M., Louzada, F. & Lachos, V. (2020) Quantile regression for nonlinear mixed effects models: a likelihood based perspective. Stat Papers 61, 1281-1307. doi:10.1007/s003620180988y

Delyon, B., Lavielle, M. & Moulines, E. (1999). Convergence of a stochastic approximation version of the EM algorithm. Annals of Statistics, pages 94-128.

See Also

`QRNLMM`,`group.plots`,`group.lines`,`Soybean`, `HIV`, `lqr`

Examples

```
  ## Not run:

#A model for comparing the two genotypes  (with covariates)

data(Soybean)
attach(Soybean)

y     = weight          #response
x     = Time            #time
covar = c(Variety)-1    #factor genotype (0=Forrest, 1=Plan Intro)

#Expression for the three parameter logistic curve with a covariate

exprNL = expression((fixed[1]+(fixed[4]*covar[1])+random[1])/
                      (1 + exp(((fixed[2]+random[2])- x)/
                      (fixed[3]+random[3]))))

#Initial values for fixed effects
initial = c(max(y),0.6*max(y),0.73*max(y),3)

# A quantile regression for percentiles p = c(0.05,0.50,0.95)

#Take your sit and some popcorn

results = qrNLMM::QRNLMM(
  y = y,
  x = x,
  groups = Plot,
  initial = initial,
  exprNL = exprNL,
```

```
  covar = covar,
  p= c(0.05,0.50,0.95),# quantiles to estimate
  MaxIter = 50,M = 15, # to accelerate
  verbose = FALSE      # show no output
)

#######################################################################
# Predicting
#######################################################################

# now we select two random subjects from original data
set.seed(19)
index = Plot %in% sample(Plot,size = 2)
index2 = c(1,diff(as.numeric(Plot)))>0 & index

# 1. Original dataset
#####################

prediction = predict(object = results)
head(prediction) # fitted values

if(TRUE){
  group.plot(x = Time[index],
             y = weight[index],
             groups = Plot[index],
             type="b",
             main="Soybean profiles",
             xlab="time (days)",
             ylab="mean leaf weight (gr)",
             col= ifelse(covar[index2],"gray70","gray90"),
             ylim = range(prediction[index,]),
             lty = 1
  )

  legend("bottomright",
         legend = c("Forrest","Plan Intro"),
         bty = "n",col = c(4,2),lty = 1)

  # predictions for these two plots

  group.lines(x = Time[index], # percentile 5
              y = prediction[index,1],
              groups = Plot[index],
              type = "l",
              col=ifelse(covar[index2],"red","blue"),
              lty = 2
  )

  group.lines(x = Time[index], # median
              y = prediction[index,2],
              groups = Plot[index],
              type = "l",
              col="black",
```

```
              lty = 2)

  group.lines(x = Time[index], # percentile 95
              y = prediction[index,3],
              groups = Plot[index],
              type = "l",
              col=ifelse(covar[index2],"red","blue"),
              lty = 2)

  legend("topleft",
         legend = c("p = 5","p = 50","p = 95"),
         col = c(4,1,4),lty = c(2,2,2),bty = "n")
}

# 2. New covariates with no response
####################################

# For the two randomly selected plots (index == TRUE)

# newdata
newdata = data.frame(new.groups = Plot[index],
                new.x = Time[index],
                new.covar = covar[index])

newdata
attach(newdata)

prediction2 = predict(object = results,
                      groups = new.groups,
                      x = new.x,
                      covar = new.covar)

# population curves

if(TRUE){
  group.plot(x = new.x, # percentile 5
             y = prediction2[,1],
             groups = new.groups,
             type = "l",
             col=ifelse(covar[index2],"red","blue"),
             lty = 2,
             main="Soybean profiles",
             xlab="time (days)",
             ylab="mean leaf weight (gr)",
             ylim = range(prediction2)
  )

  legend("bottomright",
         legend = c("Forrest","Plan Intro"),
         bty = "n",col = c(4,2),lty = 1)

  # predictions for these two plots
```

```
  group.lines(x = new.x, # median
              y = prediction2[,2],
              groups = new.groups,
              type = "l",
              col="black",
              lty = 1)

  group.lines(x = new.x, # percentile 95
              y = prediction2[,3],
              groups = new.groups,
              type = "l",
              col=ifelse(covar[index2],"red","blue"),
              lty = 2)

  legend("topleft",
         legend = c("p = 5","p = 50","p = 95"),
         col = c(4,1,4),lty = c(2,1,2),bty = "n")

  segments(x0 = new.x[new.covar==1],
           y0 = prediction2[new.covar==1,1],
           y1 = prediction2[new.covar==1,3],
           lty=2,col = "red")

  segments(x0 = new.x[new.covar==0],
           y0 = prediction2[new.covar==0,1],
           y1 = prediction2[new.covar==0,3],
           lty=2,col = "blue")
}

# 3. New covariates + response
####################################

# newdata
newdata2 = data.frame(new.groups = Plot[index],
                      new.x = Time[index],
                      new.covar = covar[index],
                      new.y = weight[index])

newdata2
attach(newdata2)

prediction2 = predict(object = results,
                       groups = new.groups,
                       x = new.x,
                       covar = new.covar,
                       y = new.y)

# individual curves (random-effects to be computed)

if(TRUE){
  group.plot(x = Time[index],
             y = weight[index],
             groups = Plot[index],
```

```
              type="b",
              main="Soybean profiles",
              xlab="time (days)",
              ylab="mean leaf weight (gr)",
              col= ifelse(covar[index2],"gray70","gray90"),
              ylim = range(prediction[index,]),
              lty = 1
  )

  legend("bottomright",
         legend = c("Forrest","Plan Intro"),
         bty = "n",col = c(4,2),lty = 1)

  # predictions for these two plots

  group.lines(x = new.x, # percentile 5
              y = prediction2[,1],
              groups = new.groups,
              type = "l",
              col=ifelse(covar[index2],"red","blue"),
              lty = 2)

  group.lines(x = new.x, # median
              y = prediction2[,2],
              groups = new.groups,
              type = "l",
              col="black",
              lty = 1)

  group.lines(x = new.x, # percentile 95
              y = prediction2[,3],
              groups = new.groups,
              type = "l",
              col=ifelse(covar[index2],"red","blue"),
              lty = 2)

  legend("topleft",
         legend = c("p = 5","p = 50","p = 95"),
         col = c(4,1,4),lty = c(2,1,2),bty = "n")
}

## End(Not run)
```

`QRNLMM` *Quantile Regression for Nonlinear Mixed-Effects Models*

Description

Performs a quantile regression for a NLMEM using the Stochastic-Approximation of the EM Algorithm (SAEM) for an unique or a set of quantiles.

Usage

```
QRNLMM(y,x,groups,initial,exprNL,covar=NA,p=0.5,precision=0.0001,MaxIter=500,
    M=20,cp=0.25,beta=NA,sigma=NA,Psi=NA,show.convergence=TRUE,CI=95,verbose=TRUE)
```

Arguments

`y`	the response vector of dimension N where N is the total of observations.
`x`	vector of longitudinal (repeated measures) covariate of dimension N. For example: Time, location, etc.
`groups`	factor of dimension N specifying the partitions of the data over which the random effects vary.
`initial`	an numeric vector, or list of initial estimates for the fixed effects. It must be provide adequately (see `details` section) in order to ensure a proper convergence.
`exprNL`	expression containing the proposed nonlinear function. It can be of class `character` or `expression`. It must have a defined structure defined in the `details` section in order to be correctly read by the derivate R function `deriv`.
`covar`	a matrix of dimension $N \times r$ where r represents the number of covariates.
`p`	unique quantile or a set of quantiles related to the quantile regression.
`precision`	the convergence maximum error.
`MaxIter`	the maximum number of iterations of the SAEM algorithm. Default = 500.
`M`	Number of Monte Carlo simulations used by the SAEM Algorithm. Default = 20. For more accuracy we suggest to use `M=20`.
`cp`	cut point $(0 \leq cp \leq 1)$ which determines the percentage of initial iterations with no memory.
`beta`	fixed effects vector of initial parameters, if desired.
`sigma`	dispersion initial parameter for the error term, if desired.
`Psi`	Variance-covariance random effects matrix of initial parameters, if desired.
`show.convergence`	if `TRUE`, it will show a graphical summary for the convergence of the estimates of all parameters for each quantile in order to assess the convergence.
`CI`	Confidence to be used for the Confidence Interval when a grid of quantiles is provided. Default=95.
`verbose`	if `TRUE`, an output summary is printed.

Details

This algorithm performs the SAEM algorithm proposed by Delyon et al. (1999), a stochastic version of the usual EM Algorithm deriving exact maximum likelihood estimates of the fixed-effects and variance components. Covariates are allowed, the longitudinal (repeated measures) coded `x` and a set of covariates `covar`.

$Aboutinitialvalues$: Estimation for fixed effects parameters envolves a Newton-Raphson step. In adition, NL models are highly sensitive to initial values. So, we suggest to set of intial values quite good, this based in the parameter interpretation of the proposed NL function.

$Aboutthenonlinearexpression$: For the NL expression `exprNL` just the variables `x`, `covar`, `fixed` and `random` can be defined. Both `x` and `covar` represent the covariates defined above. The fixed effects must be declared as `fixed[1]`, `fixed[2]`,..., `fixed[d]` representing the first, second and dth fixed effect. Exactly the same for the random effects and covariates where the term `fixed` should be replace for `random` and `covar` respectively.

For instance, if we use the exponential nonlinear function with two parameters, each parameter represented by a fixed and a random effect, this will be defined by

$$y_{ij} = (\beta_1 + b_1) \exp^{-(\beta_2 + b_2) x_{ij}}$$

and the `exprNL` should be a character or and expression defined by

`exprNL = "(fixed[1]+random[1])*exp(-(fixed[2]+random[2])*x)"`

or

`exprNL = expression((fixed[1]+random[1])*exp(-(fixed[2]+random[2])*x)).`

If we are interested in adding two covariates in order to explain on of the parameters, the covariates `covar[1]` and `covar[2]` must be included in the model. For example, for the nonlinear function

$$y_{ij} = (\beta_1 + \beta_3 * covar1_{ij} + b_1) \exp^{-(\beta_2 + \beta_4 * covar2_{ij} + b_2) x_{ij}}$$

the `exprNL` should be

`exprNL = "(fixed[1]+fixed[3]*covar[1]+random[1])*exp(-(fixed[2]+fixed[4]*covar[2]+random[2])*x)"`

or

`exprNL = expression((fixed[1]+fixed[3]*covar[1]+random[1])*exp(-(fixed[2]+ fixed[4]*covar[2]+random[2`

Note that the mathematical function `exp` was used. For derivating the `deriv` R function recognizes in the `exprNL` expression the arithmetic operators +, -, *, / and ^, and the single-variable functions `exp`, `log`, `sin`, `cos`, `tan`, `sinh`, `cosh`, `sqrt`, `pnorm`, `dnorm`, `asin`, `acos`, `atan`, `gamma`, `lgamma`, `digamma` and `trigamma`, as well as `psigamma` for one or two arguments (but derivative only with respect to the first).

$Generaldetails$: When a grid of quantiles is provided, a graphical summary with point estimates and Confidence Intervals for model parameters is shown and also a graphical summary for the convergence of these estimates (for each quantile), if `show.convergence=TRUE`.

If the convergence graphical summary shows that convergence has not be attained, it's suggested to increase the total number of iterations `MaxIter`.

About the cut point parameter `cp`, a number between 0 and 1 $(0 \leq cp \leq 1)$ will assure an initial convergence in distribution to a solution neighborhood for the first `cp*MaxIter` iterations and an almost sure convergence for the rest of the iterations. If you do not know how SAEM algorithm works, these parameters SHOULD NOT be changed.

This program uses progress bars that will close when the algorithm ends. They must not be closed before, if not, the algorithm will stop.

Value

The function returns a list with two objects

`conv`	A two elements list with the matrices `teta` and `se` containing the point estimates and standard error estimate for all parameters along all iterations.

The second element of the list is `res`, a list of 13 elements detailed as

`p`	quantile(s) fitted.
`iter`	number of iterations.
`criteria`	attained criteria value.
`nlmodel`	the proposed nonlinear function.
`beta`	fixed effects estimates.
`weights`	random effects weights (b_i).
`sigma`	scale parameter estimate for the error term.
`Psi`	Random effects variance-covariance estimate matrix.
`SE`	Standard Error estimates.
`table`	Table containing the inference for the fixed effects parameters.
`loglik`	Log-likelihood value.
`AIC`	Akaike information criterion.
`BIC`	Bayesian information criterion.
`HQ`	Hannan-Quinn information criterion.
`fitted.values`	vector containing the fitted values
`residuals`	vector containing the residuals.
`time`	processing time.

Note

If a grid of quantiles was provided, the result is a list of the same dimension where each element corresponds to each quantile as detailed above.

Author(s)

Christian E. Galarza <<`chedgala@espol.edu.ec`>> and Victor H. Lachos <<`hlachos@uconn.edu`>>

References

Galarza, C.E., Castro, L.M., Louzada, F. & Lachos, V. (2020) Quantile regression for nonlinear mixed effects models: a likelihood based perspective. Stat Papers 61, 1281-1307. doi:10.1007/s003620180988y

Delyon, B., Lavielle, M. & Moulines, E. (1999). Convergence of a stochastic approximation version of the EM algorithm. Annals of Statistics, pages 94-128.

See Also

`Soybean`, `HIV`, `lqr` , `group.plots`

Examples

```
## Not run:
#Using the Soybean data

data(Soybean)
attach(Soybean)

#################################
#A full model (no covariate)

y     = weight          #response
x     = Time            #time

#Expression for the three parameter logistic curve

exprNL = expression((fixed[1]+random[1])/
                    (1 + exp(((fixed[2]+random[2])- x)/(fixed[3]+random[3]))))

#Initial values for fixed effects
initial = c(max(y),0.6*max(y),0.73*max(y))

#A median regression (by default)
median_reg = QRNLMM(y,x,Plot,initial,exprNL)

#Assesing the fit

fxd     = median_reg$res$beta
nlmodel = median_reg$res$nlmodel
weights = median_reg$res$weights
seqc    = seq(min(x),max(x),length.out = 500)

group.plot(x = Time,y = weight,groups = Plot,type="l",
           main="Soybean profiles",xlab="time (days)",
           ylab="mean leaf weight (gr)",col="gray")

for(i in 1:nlevels(Plot)){
  lines(seqc,nlmodel(x = seqc,fixed = fxd,random = weights[i,]),lty=2)
}

# median population curve
lines(seqc,nlmodel(x = seqc,fixed = fxd,random = rep(0,ncol(weights))),
      lwd=3,col="blue")

#########################################
#A model for compairing the two genotypes

y     = weight          #response
x     = Time            #time
```

```
covar = as.numeric(Variety)-1    #factor genotype (0=Forrest, 1=Plan Introduction)

#Expression for the three parameter logistic curve with a covariate in the asymp growth
exprNL = expression((fixed[1]+(fixed[4]*covar[1])+random[1])/
                      (1 + exp(((fixed[2]+random[2])- x)/(fixed[3]+random[3]))))

#Initial values for fixed effects
initial = c(max(y),0.6*max(y),0.73*max(y),3)

# A quantile regression for the three quartiles (just 200 iterations)
box_reg = QRNLMM(y,x,Plot,initial,exprNL,covar,p=c(0.25,0.50,0.75),MaxIter = 200)

# Assing the fit for the first quartile Q1 curve
fxd_q1    = box_reg[[1]]$res$beta
fxd_q2    = box_reg[[2]]$res$beta #median
fxd_q3    = box_reg[[3]]$res$beta
nlmodel = box_reg[[1]]$res$nlmodel
seqc    = seq(min(x),max(x),length.out = 500)


group.plot(x = Time[Variety=="P"],y = weight[Variety=="P"],
           groups = Plot[Variety=="P"],type="l",col="lightblue",
           main="Soybean profiles by genotype",xlab="time (days)",
           ylab="mean leaf weight (gr)")

group.lines(x = Time[Variety=="F"],y = weight[Variety=="F"],
            groups = Plot[Variety=="F"],col="gray")

# Add the three quantile lines
lines(seqc,nlmodel(x = seqc, fixed = fxd_q1, random = rep(0, 3), covar = 1),
      lwd = 2, col = "blue", lty = "dashed") # Q1, dashed
lines(seqc, nlmodel(x = seqc, fixed = fxd_q2, random = rep(0, 3), covar = 1),
      lwd = 2, col = "blue")             # Median, solid
lines(seqc, nlmodel(x = seqc, fixed = fxd_q3, random = rep(0, 3), covar = 1),
      lwd = 2, col = "blue", lty = "dashed") # Q3, dashed

lines(seqc,nlmodel(x = seqc,fixed = fxd_q1,random = rep(0,3),covar=0),
      lwd=2,col="black",lty="dashed") #q1
lines(seqc,nlmodel(x = seqc,fixed = fxd_q2,random = rep(0,3),covar=0),
      lwd=2,col="black")             # Median
lines(seqc,nlmodel(x = seqc,fixed = fxd_q3,random = rep(0,3),covar=0),
      lwd=2,col="black",lty="dashed") #q3


#########################################
#A simple output example

---------------------------------------------------
Quantile Regression for Nonlinear Mixed Model
---------------------------------------------------
Quantile = 0.5
Subjects = 48 ; Observations = 412
```

```
- Nonlinear function

function(x,fixed,random,covar=NA){
  resp = (fixed[1] + random[1])/(1 + exp(((fixed[2] +
          random[2]) - x)/(fixed[3] + random[3])))
  return(resp)}

-----------
Estimates
-----------
- Fixed effects

       Estimate Std. Error   z value Pr(>|z|)
beta 1 18.80029    0.53098  35.40704        0
beta 2 54.47930    0.29571 184.23015        0
beta 3  8.25797    0.09198  89.78489        0

sigma = 0.31569

Random effects Variance-Covariance Matrix matrix
         b1       b2      b3
b1 24.36687 12.27297 3.24721
b2 12.27297 15.15890 3.09129
b3  3.24721  3.09129 0.67193

------------------------
Model selection criteria
------------------------
        Loglik      AIC      BIC       HQ
Value -622.899 1265.798 1306.008 1281.703

-------
Details
-------
Convergence reached? = FALSE
Iterations = 300 / 300
Criteria = 0.00058
MC sample = 20
Cut point = 0.25
Processing time = 22.83885 mins

## End(Not run)
```

`Soybean` *Growth of soybean plants*

Description

The `Soybean` data frame has 412 rows and 5 columns.

Format

This data frame contains the following columns:

Plot a factor giving a unique identifier for each plot.

Variety a factor indicating the variety; Forrest (F) or Plant Introduction #416937 (P).

Year a factor indicating the year the plot was planted.

Time a numeric vector giving the time the sample was taken (days after planting).

weight a numeric vector giving the average leaf weight per plant (g).

Details

These data are described in Davidian and Giltinan (1995, 1.1.3, p.7) as "Data from an experiment to compare growth patterns of two genotypes of soybeans: Plant Introduction #416937 (P), an experimental strain, and Forrest (F), a commercial variety." In order to fit the Nonlinear data we sugest to use the three parameter logistic model as in Pinheiro & Bates (1995).

Source

Pinheiro, J. C. and Bates, D. M. (2000), *Mixed-Effects Models in S and S-PLUS*, Springer, New York. (Appendix A.27)

Davidian, M. and Giltinan, D. M. (1995), *Nonlinear Models for Repeated Measurement Data*, Chapman and Hall, London.

Examples

```
## Not run:
data(Soybean)
attach(Soybean)

#################################
#A full model (no covariate)

y     = weight          #response
x     = Time            #time

#Expression for the three parameter logistic curve

exprNL = expression((fixed[1]+random[1])/(1 + exp(((fixed[2]+random[2])- x)/(fixed[3]+random[3]))))

#Initial values for fixed effects
initial = c(max(y),0.6*max(y),0.73*max(y))

#A median regression (by default)
median_reg = QRNLMM(y,x,Plot,initial,exprNL)

#Assing the fit

fxd     = median_reg$res$beta
nlmodel = median_reg$res$nlmodel
```

```
seqc    = seq(min(x),max(x),length.out = 500)

group.plot(x = Time,y = weight,groups = Plot,type="l",
           main="Soybean profiles",xlab="time (days)",
           ylab="mean leaf weight (gr)",col="gray")

lines(seqc,nlmodel(x = seqc,fixed = fxd,random = rep(0,3)),
      lwd=2,col="blue")

#Histogram for residuals
hist(median_reg$res$residuals,breaks = 20)

#########################################
#A model for compairing the two genotypes

y     = weight          #response
x     = Time            #time
covar = as.numeric(Variety)-1    #factor genotype (0=Forrest, 1=Plan Introduction)

#Expression for the three parameter logistic curve with a covariate in the asymp growth
exprNL = expression((fixed[1]+(fixed[4]*covar[1])+random[1])/
                      (1 + exp(((fixed[2]+random[2])- x)/(fixed[3]+random[3]))))

#Initial values for fixed effects
initial = c(max(y),0.6*max(y),0.73*max(y),3)

# A quantile regression for the three quartiles ()
box_reg = QRNLMM(y,x,Plot,initial,exprNL,covar,p=c(0.25,0.50,0.75),MaxIter = 20)

# Assing the fit for the first quartile Q1 curve
fxd_q1    = box_reg[[1]]$res$beta
fxd_q2    = box_reg[[2]]$res$beta #median
fxd_q3    = box_reg[[3]]$res$beta
nlmodel = box_reg[[1]]$res$nlmodel
seqc    = seq(min(x),max(x),length.out = 500)


group.plot(x = Time[Variety=="P"],y = weight[Variety=="P"],
           groups = Plot[Variety=="P"],type="l",col="lightblue",
           main="Soybean profiles by genotype",xlab="time (days)",
           ylab="mean leaf weight (gr)")

group.lines(x = Time[Variety=="F"],y = weight[Variety=="F"],
            groups = Plot[Variety=="F"],col="gray")

# Add the three quantile lines
lines(seqc,nlmodel(x = seqc, fixed = fxd_q1, random = rep(0, 3), covar = 1),
      lwd = 2, col = "blue", lty = "dashed") # Q1, dashed
lines(seqc, nlmodel(x = seqc, fixed = fxd_q2, random = rep(0, 3), covar = 1),
      lwd = 2, col = "blue")             # Median, solid
lines(seqc, nlmodel(x = seqc, fixed = fxd_q3, random = rep(0, 3), covar = 1),
      lwd = 2, col = "blue", lty = "dashed") # Q3, dashed
```

```
lines(seqc,nlmodel(x = seqc,fixed = fxd_q1,random = rep(0,3),covar=0),
      lwd=2,col="black",lty="dashed") #q1
lines(seqc,nlmodel(x = seqc,fixed = fxd_q2,random = rep(0,3),covar=0),
      lwd=2,col="black")             # Median
lines(seqc,nlmodel(x = seqc,fixed = fxd_q3,random = rep(0,3),covar=0),
      lwd=2,col="black",lty="dashed") #q3



## End(Not run)
```

Index