Package 'plaqr'

October 14, 2022

Title Partially Linear Additive Quantile Regression

Description Estimation, prediction, thresholding, transformation, and plotting for partially linear additive quantile regression. Intuitive functions for fitting and plotting partially linear additive quantile regression models. Uses and works with functions from the 'quantreg' package.

Version 2.0

Maintainer Adam Maidman <maidm004@umn.edu>

Depends R (>= 3.0), quantreg, splines

License GPL (>= 2)

Repository CRAN

Date/Publication 2017-08-08 18:35:59 UTC

NeedsCompilation no

Author Adam Maidman [cre, aut]

R topics documented:

bic 2
nonlinEffect 3
plaqr 4
plot.plaqreffect 6
predictInt 7
print.plaqreffect 8
print.thresh 9
simData 9
threshold 10
transform_plaqr 11
trans_parameter 14

Index **15**

bic *BIC for the Partially Linear Additive Quantile Regression Model*

Description

Returns the BIC for the partially linear additive quantile regression model from Lee, Noh, and Park (2014).

Usage

```
bic(fit, ...)
```

Arguments

fit	a "plaqr" object obtained from a call to plaqr
...	additional parameters which will be ignored

Value

BIC value

Author(s)

Adam Maidman

References

Lee, E. R., Noh, H., and Park, B. U. (2014). Model selection via bayesian information criterion for quantile regression models. Journal of the American Statistical Association 109, 216-229.

Examples

```
data(simData)

ss <- vector("list", 2)

ss[[2]]$degree <- 3
fit1 <- plaqr(y~., nonlinVars=~z1+z2, data=simData, splinesettings=ss)

ss[[2]]$degree <- 4
fit2 <- plaqr(y~., nonlinVars=~z1+z2, data=simData, splinesettings=ss)

ss[[2]]$degree <- 5
fit3 <- plaqr(y~., nonlinVars=~z1+z2, data=simData, splinesettings=ss)

bic(fit1)
bic(fit2)
bic(fit3)
```

nonlinEffect *Nonlinear Effects Plots*

Description

Returns an object of class "plaqreffect" which represents the effect plot(s) of the nonlinear term(s) of a "plaqr" object from the plaqr function. A "plaqreffect" object should be plotted using the plot function.

Usage

```
nonlinEffect(fit, select=NULL, renames=NULL)
```

Arguments

fit	a "plaqr" object.
select	a character vector with entries matching nonlinear terms in fit.
renames	a character vector with length equal to the number of nonlinear terms in select (if select is NULL, the length must be equal to the number of nonlinear terms in fit). The first entry renames the first nonlinear term for plotting purposes, and so on. Note that select can reorder the nonlinear terms (see the examples).

Value

A returned "plaqreffect" object to be used with the "plot" function. Each nonlinear term is associated with a list containing information for plotting. See the examples for accessing the list.

Author(s)

Adam Maidman

Examples

```
data(simData)
fit <- plaqr(y~.,~z1+z2,data=simData)

eff1 <- nonlinEffect(fit)
eff1
plot(eff1)

eff2 <- nonlinEffect(fit, select=c("z1","z2"), renames=c("Length", "Height"))
eff2
plot(eff2)

eff3 <- nonlinEffect(fit, select=c("z2","z1"), renames=c("Height", "Length"))
eff3
eff3$z1
eff3$z2
```

```
plot(eff3)

par(mfrow=c(1,2))
plot(eff3)
```

---

plaqr *Partially Linear Additive Quantile Regression*

Description

Returns an object of class "plaqr" and "rq" that represents a quantile regression fit. A nonlinear term z is transformed using bs(z) before fitting the model. The formula of the model (as it appears in R) becomes y~ x1 + x2 + bs(z1) + bs(z2) where bs(z1) is a B-spline.

Usage

```
plaqr(formula, nonlinVars=NULL, tau=.5, data=NULL, subset,
            weights, na.action, method = "br", model = TRUE,
            contrasts = NULL, splinesettings=NULL, ...)
```

Arguments

formula	a formula object, with the response on the left of a ~ operator, and the linear terms, separated by + operators, on the right. Any terms on the right of the ~ operator that also appear in nonlinVars will be included in the model as spline terms, not linear terms.
nonlinVars	a one-sided formula object, with a ~ operator to the left of the nonlinear terms seperated by + operators. A term appearing in both formula and nonlinVars will be treated as a nonlinear term. If nonlinVars is not NULL, then an intercept will automatically be included in the model (despite a -1 or 0 term included in formula).
tau	the quantile to be estimated, this is a number strictly between 0 and 1 (for now).
data	a data.frame in which to interpret the variables named in the formula, or in the subset and the weights argument. If this is missing, then the variables in the formula should be on the search list. This may also be a single number to handle some special cases – see below for details.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	vector of observation weights; if supplied, the algorithm fits to minimize the sum of the weights multiplied into the absolute residuals. The length of weights must be the same as the number of observations. The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous.
na.action	a function to filter missing data. This is applied to the model.frame after any subset argument has been used. The default (with na.fail) is to create an error if any missing values are found. A possible alternative is na.omit, which deletes observations that contain one or more missing values.

model	if TRUE then the model frame is returned. This is essential if one wants to call summary subsequently.
method	the algorithmic method used to compute the fit. There are several options: The default method is the modified version of the Barrodale and Roberts algorithm for l_1-regression, used by l1fit in S, and is described in detail in Koenker and d'Orey(1987, 1994), default = "br". This is quite efficient for problems up to several thousand observations, and may be used to compute the full quantile regression process. It also implements a scheme for computing confidence intervals for the estimated parameters, based on inversion of a rank test described in Koenker(1994). For larger problems it is advantagous to use the Frisch–Newton interior point method "fn". And very large problems one can use the Frisch–Newton approach after preprocessing "pfn". Both of the latter methods are described in detail in Portnoy and Koenker(1997). There is a fifth option "fnc" that enables the user to specify linear inequality constraints on the fitted coefficients; in this case one needs to specify the matrix R and the vector r representing the constraints in the form $Rb \geq r$. See the examples. Finally, there are two penalized methods: "lasso" and "scad" that implement the lasso penalty and Fan and Li's smoothly clipped absolute deviation penalty, respectively. These methods should probably be regarded as experimental.
contrasts	a list giving contrasts for some or all of the factors default = NULL appearing in the model formula. The elements of the list should have the same name as the variable and should be either a contrast matrix (specifically, any full-rank matrix with as many rows as there are levels in the factor), or else a function to compute such a matrix given the number of levels.
splinesettings	a list of length equal to the number of nonlinear effects containing arguments to pass to the bs function for each term. Each element of the list is either NULL or a list with named elements correpsonding to the arguments in bs. If not NULL, the first element of splinesettings corresponds to the first nonlinear effect and so on.
...	additional arguments for the fitting routines (see the rq function in the 'quantreg' package).

Value

Returns the following:

coefficients	Coefficients from the fitted model
x	optionally the model matrix, if x=TRUE.
y	optionally the response, if y=TRUE.
residuals	the residuals from the fit.
dual	the vector dual variables from the fit.
fitted.values	fitted values from the fit.
formula	the formula that was used in the rq function.
rho	the value of the objective function at the solution.
model	optionally the model frame, if model=TRUE
linear	the linear terms used in the model fit.
nonlinear	the nonlinear terms used in the model fit.
z	the values of the nonlinear terms.

Author(s)

Adam Maidman

References

Hastie, T. J. (1992) Generalized additive models. Chapter 7 of Statistical Models in S eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

Koenker, R. W. (2005). Quantile Regression, Cambridge U. Press.

Sherwood, B. and Wang, L. (2016). Partially linear additive quantile regression in ultra-high dimension. The Annals of Statistics 44, 288-317.

Maidman, A., Wang, L. (2017). New Semiparametric Method for Predicting High-Cost Patients. Preprint.

Examples

```
data(simData)

ss <- vector("list", 2)
ss[[2]]$degree <- 5
ss[[2]]$Boundary.knots <- c(-1, 1)

plaqr(y~., nonlinVars=~z1+z2, data=simData)
#same as plaqr(formula= y~x1+x2+x3, nonlinVars=~z1+z2, data=simData)

plaqr(y~0, nonlinVars=~z1+z2, data=simData, splinesettings=ss) #no linear terms in the model

plaqr(y~., data=simData) #all linear terms
```

---

plot.plaqreffect *Nonlinear Effect Plot for a Partially Linear Additive Quantile Regression Model*

Description

Makes nonlinear effect plots for the nonlinear effects in a fit returned from the `nonlinEffect` function. Note: you cannot use this function to plot a `"plaqr"` object.

Usage

```
## S3 method for class 'plaqreffect'
plot(x, select=NULL, rug = TRUE, jit = TRUE, titles = NULL, pages = 0, type="l", ...)
```

Arguments

x	a plaqreffect object returned from nonlinEffect.
select	vector of indices of nonlinear terms in x to be plotted, by default all.
rug	if TRUE, a rugplot for the x-coordinate is plotted.
jit	if TRUE, the x-values of the rug plot are jittered.
titles	title(s) as vector of character strings, by default titles are chosen for each plot as "Effect of CovariateName (tau=tau)".
pages	number of pages desired for the plots.
type	the type of plot that should be drawn.
...	additional arguments for the plotting algorithm.

Author(s)

Adam Maidman

Examples

```
data(simData)
fit <- plaqr(y~.,~z1+z2,data=simData)
eff <- nonlinEffect(fit, select=c("z1","z2"), renames=c("Length", "Height"))
eff

plot(eff)
plot(eff, select=1, col="red")
plot(eff, select=c(2,1), titles=c("Effect Z1","Effect Z2"))
plot(eff, select=1, col="red", lwd=4)

par(mfrow=c(1,2))
plot(eff)
```

---

predictInt *Prediction Inteval for Quantile Regression*

Description

Predicts future values using the median and finds a prediction interval for future values using an upper and lower quantile. The lower quantile is (1-level)/2 and the upper quantile is .5 + level/2.

Usage

```
predictInt(fit, level=.95, newdata=NULL, ...)
```

Arguments

fit	a fitted model of class "plaqr" or "rq" to be used for prediction.
level	the prediction level required. The lower quantile is (1-level)/2 and the upper quantile is .5 + level/2.
newdata	an optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
...	additional argument(s) for methods.

Value

a matrix with columns giving the predicted median and lower and upper prediction bounds.

Author(s)

Adam Maidman

Examples

```
data(simData)
fit <- plaqr(y~.,~z1+z2,data=simData)
predictInt(fit, level=.95)
```

---

print.plaqreffect *Print a plaqreffect object*

Description

Print an object generated by nonlinEffect.

Usage

```
## S3 method for class 'plaqreffect'
print(x, ...)
```

Arguments

x	an object returned from nonlinEffect.
...	optional arguments.

Author(s)

Adam Maidman

print.thresh *Print a thresh Object*

Description

Print an object generated by threshold.

Usage

```
## S3 method for class 'thresh'
print(x,...)
```

Arguments

x	an object returned from threshold.
...	optional arguments.

Author(s)

Adam Maidman

simData *Simulated Data*

Description

A simulated data set to illustrate the functions in this package.

```
set.seed(4)
x1 <- rbinom(100, 1,.5)
x2 <- rnorm(100)
x3 <- rnorm(100)
z1 <- runif(100, 0, 1)
z2 <- runif(100, -1, 1)
y <- 3*x1 +1.5*x2 + 2*x3 + 5*sin(2*pi*z1) + 5*z2^3 + rnorm(100)
simData <- data.frame(y,x1,x2,x3,z1,z2)
```

Usage

```
data(simData)
```

Format

A data frame with 100 observations on the following 6 variables.

y response: expenditure

x1 male/female (a linear term)

x2 distance north/south from center (a linear term)

x3 distance east/west from center (a linear term)

z1 income/(max income) (a nonlinear term)

z2 spending habits on a -1 to 1 scale (frugal to lavish) (a nonlinear term)

`threshold` *Classifying a Numerical Response Using a Threshold*

Description

Classification of a numerical response into a "high" class and "low" class using a threshold. This function can be used with any model that has a numerical outcome and allows for prediction using the `predict` function.

Usage

```
threshold(fit, t, newdata=NULL, ...)
```

Arguments

`fit`	any model with a numerical response.
`t`	the desired threshold value. All values above `t` will be labeled "1"and all values below `t` will be labeled "0".
`newdata`	an optional data frame in which to look for variables with which to predict. If omitted, no prediction is done.
`...`	additional argument(s) for methods in the `predict` function.

Value

`pred.class`	if `newdata` is not `NULL`, then `pred.class` is a vector of predicted classes for `newdata`. If `newdata` is `NULL`, then `pred.class` is `NULL`.
`t`	the threshold.
`train.class`	a vector of the predicted classes of the data used in `fit`.
`true.class`	a vector of the true classes of the data used in `fit`.
`train.error`	a scalar equal to the `mean(train.class != true.class)`.
`true.high`	the number of observations in class"1" using the data used in `fit`.
`true.low`	the number of observations in class "1" using the data used in `fit`.

false.high	the number of observations truly in class "0", but predicted to be in class "1" using the data used in fit.
false.low	the number of observations truly in class "1", but predicted to be in class "1" using the data used in fit.
call	the call of fit.
formula	the formula used in fit.

Author(s)

Adam Maidman

Examples

```
data(simData)
fit <- plaqr(y~.,~z1+z2,data=simData)
testdata <- .5*simData[4,2:6]
trh <- threshold(fit, t=9, newdata=testdata)
trh$pred.class
trh
```

---

transform_plaqr *Transformation for Partially Linear Additive Quantile Regression*

Description

Returns the estimated transformation parameter for the one-parameter symmetric transformation (Geraci and Jones, 2015). Confidence intervals for the transformation parameter can also be created using the bootstrap. The response variable must be strictly positive; a constant can be added to the variable to ensure that all values are positive.

Usage

```
transform_plaqr(formula, nonlinVars=NULL, tau=.5, data=NULL, lambda=seq(0,1,by=.05),
                confint=NULL, B=99, subset, weights, na.action, method = "br",
                contrasts = NULL, splinesettings=NULL)
```

Arguments

formula	a formula object, with the response on the left of a ~ operator, and the linear terms, separated by + operators, on the right. Any terms on the right of the ~ operator that also appear in nonlinVars will be included in the model as spline terms, not linear terms.
nonlinVars	a one-sided formula object, with a ~ operator to the left of the nonlinear terms seperated by + operators. A term appearing in both formula and nonlinVars will be treated as a nonlinear term. If nonlinVars is not NULL, then an intercept will automatically be included in the model (despite a -1 or 0 term included in formula).

`tau` the quantile to be estimated, this is a number strictly between 0 and 1 (for now).

`data` a data.frame in which to interpret the variables named in the formula, or in the subset and the weights argument. If this is missing, then the variables in the formula should be on the search list. This may also be a single number to handle some special cases – see below for details.

`lambda` a real-valued sequence of possible transformation parameters. 0 corresponds to the log transformation and 1 corresponds to the identity. The transformation is symmetric so a negative transformation parameter is redundant and can be avoided. See Geraci and Jones (2015) for more information on the one-parameter, symmetric transformation.

`confint` a `confint` confidence interval for the transformation parameter will be created if `confint` is a number between 0 and 1 (otherwise automatically creates 95% CI). Otherwise, no confidence interval will be created. The bootstrap is used to create the confidence interval.

`B` the number of bootstrap replications for the confidence interval. If no confidence interval is being created, this argument is ignored.

`subset` an optional vector specifying a subset of observations to be used in the fitting process.

`weights` vector of observation weights; if supplied, the algorithm fits to minimize the sum of the weights multiplied into the absolute residuals. The length of weights must be the same as the number of observations. The weights must be nonnegative and it is strongly recommended that they be strictly positive, since zero weights are ambiguous.

`na.action` a function to filter missing data. This is applied to the model.frame after any subset argument has been used. The default (with `na.fail`) is to create an error if any missing values are found. A possible alternative is `na.omit`, which deletes observations that contain one or more missing values.

`method` the algorithmic method used to compute the fit. There are several options: The default method is the modified version of the Barrodale and Roberts algorithm for l_1-regression, used by `l1fit` in S, and is described in detail in Koenker and d'Orey(1987, 1994), default = `"br"`. This is quite efficient for problems up to several thousand observations, and may be used to compute the full quantile regression process. It also implements a scheme for computing confidence intervals for the estimated parameters, based on inversion of a rank test described in Koenker(1994). For larger problems it is advantagous to use the Frisch–Newton interior point method `"fn"`. And very large problems one can use the Frisch–Newton approach after preprocessing `"pfn"`. Both of the latter methods are described in detail in Portnoy and Koenker(1997). There is a fifth option `"fnc"` that enables the user to specify linear inequality constraints on the fitted coefficients; in this case one needs to specify the matrix `R` and the vector `r` representing the constraints in the form $Rb \geq r$. See the examples. Finally, there are two penalized methods: `"lasso"` and `"scad"` that implement the lasso penalty and Fan and Li's smoothly clipped absolute deviation penalty, respectively. These methods should probably be regarded as experimental.

`contrasts` a list giving contrasts for some or all of the factors default = `NULL` appearing in the model formula. The elements of the list should have the same name as the

variable and should be either a contrast matrix (specifically, any full-rank matrix with as many rows as there are levels in the factor), or else a function to compute such a matrix given the number of levels.

`splinesettings` a list of length equal to the number of nonlinear effects containing arguments to pass to the `bs` function for each term. Each element of the list is either `NULL` or a list with named elements correpsonding to the arguments in `bs`. If not `NULL`, the first element of splinesettings corresponds to the first nonlinear effect and so on.

Value

Returns the following:

`parameter`	The transformation parameter
`Y`	The values of the transformed response
`confint`	If a confidence interval is created, this is the confidence interval for the transformation parameter. Otherwise, `NULL`.
`U`	If a confidence interval is created, a `B` by `n` matrix containing the indices used in each bootstrap sample. Otherwise, `NULL`.
`P`	If a confidence interval is created, a `B` length vector containing the transformation parameter estimated in each bootstrap sample. Otherwise, `NULL`.

Author(s)

Adam Maidman

References

Geraci, M. and Jones, M. (2015). Improved transformation-based quantile regression. Canadian Journal of Statistics 43, 118-132.

Maidman, A., Wang, L. (2017). New Semiparametric Method for Predicting High-Cost Patients. Preprint.

Examples

```
data(simData)

simData$Y <- exp(simData$y)

transform_plaqr(Y~x1+x2+x3, nonlinVars=~z1+z2, data=simData)

transform_plaqr(Y~x1+x2+x3, nonlinVars=~z1+z2, confint=.95, data=simData)
```

`trans_parameter` *Transformation of the Response Variable*

Description

Transform the response variable using the one-paremter, symmetric transformation of Geraci and Jones (2015).

Usage

```
trans_parameter(x, parameter, inverse=FALSE)
```

Arguments

`x`	a vector of values to be transformed (the response variable)
`parameter`	a real-valued transformation parameter. 0 corresponds to the log transformation and 1 corresponds to the identity. See Geraci and Jones (2015) for more information on the one-parameter, symmetric transformation.
`inverse`	If TRUE, the inverse transformation is done to transform the variable back to the original scale. If FALSE, the standard transformation is computed.

Value

Returns a vector of the transformed (or back-transformed) variable.

Author(s)

Adam Maidman

References

Geraci, M. and Jones, M. (2015). Improved transformation-based quantile regression. Canadian Journal of Statistics 43, 118-132.

Maidman, A., Wang, L. (2017). New Semiparametric Method for Predicting High-Cost Patients. Preprint.

Examples

```
data(simData)
simData$Y <- exp(simData$y)

tparam <- transform_plaqr(Y~x1+x2+x3, nonlinVars=~z1+z2, data=simData)

simData$newy <- trans_parameter(simData$Y, tparam$parameter)

fit <- plaqr(newy~x1+x2+x3, nonlinVars=~z1+z2, data=simData)

trans_parameter( predictInt(fit), tparam$parameter, inverse=TRUE)
```

Index

bic, 2

nonlinEffect, 3

plaqr, 4
plot.plaqreffect, 6
predictInt, 7
print.plaqreffect, 8
print.thresh, 9

simData, 9

threshold, 10
trans_parameter, 14
transform_plaqr, 11