Package ‘quantregForest’

October 7, 2024

Type Package

Title Quantile Regression Forests

Version 1.3-7.1

Date 2017-12-16

Depends randomForest, RColorBrewer

Imports stats, parallel

Suggests gss, knitr, rmarkdown

Description Quantile Regression Forests is a tree-based ensemble
method for estimation of conditional quantiles. It is
particularly well suited for high-dimensional data. Predictor
variables of mixed classes can be handled. The package is
dependent on the package 'randomForest', written by Andy Liaw.

License GPL

NeedsCompilation yes

URL `https://github.com/lorismichel/quantregForest`

BugReports `https://github.com/lorismichel/quantregForest/issues`

RoxygenNote 6.0.1

Repository CRAN

Date/Publication 2024-10-07 08:21:02 UTC

Author Nicolai Meinshausen [aut],
Loris Michel [cre]

Maintainer Loris Michel <`michel@stat.math.ethz.ch`>

Contents

predict.quantregForest *Prediction method for class quantregForest*

Description

Prediction of test data with quantile regression forests.

Usage

```
## S3 method for class 'quantregForest'
predict(object, newdata = NULL,
                         what = c(0.1, 0.5, 0.9), ...)
```

Arguments

`object`	An object of class `quantregForest`
`newdata`	A data frame or matrix containing new data. If left at default `NULL`, the out-of-bag predictions (OOB) are returned, for which the option `keep.inbag` has to be set to `TRUE` at the time of fitting the object.
`what`	Can be a vector of quantiles or a function. Default for `what` is a a vector of quantiles (with numerical values in [0,1]) for which the conditional quantile estimates should be returned. If a function it has to take as argument a numeric vector and return either a summary statistic (such as `mean`,`median` or `sd` to get conditional mean, median or standard deviation) or a vector of values (such as with `quantiles` or via `sample`) or a function (for example with `ecdf`).
`...`	Additional arguments (currently not in use).

Value

A vector, matrix or list.

If `what` is a vector with desired quantiles (the default is `what=c(0.1,0.5,0.9)`), a matrix with one column per requested quantile returned.

If just a single quantile is specified (for example via `what=0.5`), a vector is returned.

If `what` is a function with numerical return value (for example via `what=function(x) sample(x,10,replace=TRUE)` to sample 10 new observations from conditional distribution), the output is also a matrix (or vector if just a scalar is returned).

If `what` has a function as output (such as `what=ecdf`), a list will be returned with one element per new sample point (and the element contains the desired function).

Author(s)

Nicolai Meinshausen, Christina Heinze

See Also

quantregForest, predict.quantregForest

Examples

```
################################################
##  Load air-quality data (and preprocessing) ##
################################################

data(airquality)
set.seed(1)


## remove observations with mising values
airquality <- airquality[ !apply(is.na(airquality), 1,any), ]

## number of remining samples
n <- nrow(airquality)


## divide into training and test data
indextrain <- sample(1:n,round(0.6*n),replace=FALSE)
Xtrain     <- airquality[ indextrain,2:6]
Xtest      <- airquality[-indextrain,2:6]
Ytrain     <- airquality[ indextrain,1]
Ytest      <- airquality[-indextrain,1]




################################################
##     compute Quantile Regression Forests    ##
################################################

qrf <- quantregForest(x=Xtrain, y=Ytrain)
qrf <- quantregForest(x=Xtrain, y=Ytrain, nodesize=10,sampsize=30)


## predict 0.1, 0.5 and 0.9 quantiles for test data
conditionalQuantiles  <- predict(qrf,  Xtest)
print(conditionalQuantiles[1:4,])

## predict 0.1, 0.2,..., 0.9 quantiles for test data
conditionalQuantiles  <- predict(qrf, Xtest, what=0.1*(1:9))
print(conditionalQuantiles[1:4,])

## estimate conditional standard deviation
conditionalSd <- predict(qrf,  Xtest, what=sd)
print(conditionalSd[1:4])

## estimate conditional mean (as in original RF)
```

```
conditionalMean <- predict(qrf,  Xtest, what=mean)
print(conditionalMean[1:4])

## sample 10 new observations from conditional distribution at each new sample
newSamples <- predict(qrf, Xtest,what = function(x) sample(x,10,replace=TRUE))
print(newSamples[1:4,])


## get ecdf-function for each new test data point
## (output will be a list with one element per sample)
condEcdf <- predict(qrf,  Xtest, what=ecdf)
condEcdf[[10]](30) ## get the conditional distribution at value 30 for i=10
## or, directly, for all samples at value 30 (returns a vector)
condEcdf30 <- predict(qrf, Xtest, what=function(x) ecdf(x)(30))
print(condEcdf30[1:4])

## to use other functions of the package randomForest, convert class back
class(qrf) <- "randomForest"
importance(qrf) ## importance measure from the standard RF
```

`quantregForest` *Quantile Regression Forests*

Description

Quantile Regression Forests infer conditional quantile functions from data

Usage

```
quantregForest(x,y, nthreads=1, keep.inbag=FALSE, ...)
```

Arguments

`x`	A matrix or data.frame containing the predictor variables.
`y`	The response variable.
`nthreads`	The number of threads to use (for parallel computation).
`keep.inbag`	Keep information which observations are in and out-of-bag? For out-of-bag predictions, this argument needs to be set to `TRUE`.
`...`	Other arguments passed to `randomForest` such as `nodesize` or `mtry` etc.

Details

The object can be converted back into a standard `randomForest` object and all the functions of the `randomForest` package can then be used (see example below).

The response y should in general be numeric. However, some use cases exists if y is a factor (such as sampling from conditional distribution when using for example what=function(x) sample(x,10)). Trying to generate quantiles will generate an error if y is a factor, though.

Parallel computation is invoked by setting the value of nthreads to values larger than 1 (for example to the number of available CPUs). The argument only has an effect under Linux and Mac OSX and is without effect on Windows due to restrictions on forking.

Value

A value of class quantregForest, for which print and predict methods are available. Class quantregForest is a list of the following components additional to the ones given by class randomForest:

call	the original call to quantregForest
valuesNodes	a matrix that contains per tree and node one subsampled observation

Author(s)

Nicolai Meinshausen, Christina Heinze

References

N. Meinshausen (2006) "Quantile Regression Forests", Journal of Machine Learning Research 7, 983-999 https://jmlr.csail.mit.edu/papers/v7/

See Also

predict.quantregForest

Examples

```
################################################
##  Load air-quality data (and preprocessing) ##
################################################

data(airquality)
set.seed(1)


## remove observations with mising values
airquality <- airquality[ !apply(is.na(airquality), 1,any), ]

## number of remining samples
n <- nrow(airquality)


## divide into training and test data
indextrain <- sample(1:n,round(0.6*n),replace=FALSE)
Xtrain     <- airquality[ indextrain,2:6]
Xtest      <- airquality[-indextrain,2:6]
Ytrain     <- airquality[ indextrain,1]
Ytest      <- airquality[-indextrain,1]
```

```
################################################
##     compute Quantile Regression Forests    ##
################################################

qrf <- quantregForest(x=Xtrain, y=Ytrain)
qrf <- quantregForest(x=Xtrain, y=Ytrain, nodesize=10,sampsize=30)


## for parallel computation use the nthread option
## qrf <- quantregForest(x=Xtrain, y=Ytrain, nthread=8)

## predict 0.1, 0.5 and 0.9 quantiles for test data
conditionalQuantiles  <- predict(qrf,  Xtest)
print(conditionalQuantiles[1:4,])

## predict 0.1, 0.2,..., 0.9 quantiles for test data
conditionalQuantiles  <- predict(qrf, Xtest, what=0.1*(1:9))
print(conditionalQuantiles[1:4,])

## estimate conditional standard deviation
conditionalSd <- predict(qrf,  Xtest, what=sd)
print(conditionalSd[1:4])

## estimate conditional mean (as in original RF)
conditionalMean <- predict(qrf,  Xtest, what=mean)
print(conditionalMean[1:4])

## sample 10 new observations from conditional distribution at each new sample
newSamples <- predict(qrf, Xtest,what = function(x) sample(x,10,replace=TRUE))
print(newSamples[1:4,])


## get ecdf-function for each new test data point
## (output will be a list with one element per sample)
condEcdf <- predict(qrf,  Xtest, what=ecdf)
condEcdf[[10]](30) ## get the conditional distribution at value 30 for i=10
## or, directly, for all samples at value 30 (returns a vector)
condEcdf30 <- predict(qrf, Xtest, what=function(x) ecdf(x)(30))
print(condEcdf30[1:4])

## to use other functions of the package randomForest, convert class back
class(qrf) <- "randomForest"
importance(qrf) ## importance measure from the standard RF


#####################################
## out-of-bag predictions and sampling
##################################

## for with option keep.inbag=TRUE
qrf <- quantregForest(x=Xtrain, y=Ytrain, keep.inbag=TRUE)
```

```
## or use parallel version
## qrf <- quantregForest(x=Xtrain, y=Ytrain, nthread=8)

## get quantiles
oobQuantiles <- predict( qrf, what= c(0.2,0.5,0.8))

## sample from oob-distribution
oobSample <- predict( qrf, what= function(x) sample(x,1))
```

Index