Quantile Regression Forests - An R-Vignette

Lukas Schiesser

1 Introduction

The following few pages try to give a more detailed guideline to the use of quantile regression forests in R.
After installing the package it can be loaded by the command:

```
> library(quantregForest)
```

Datasets Two datasets will be considered to illustrate how and when to use which methods for quantile regression forests. As example of a small dataset *Ozone* from the package `gss` (Gu, 2014) with 330 observations of 10 variables and secondly a large data set named *CASP* with 45′370 available observations of 9 variables from the UCI Machine Learning Repository (Lichman, 2013) are used. In the following even not all observations of the second set are used since the desired effect can already be highlighted with less.

2 Quantile Regression Forests

The first step when working with quantile regression forests is to grow such a forest. The help file of the function `quantregForest`

```
> help(quantregForest)
```

specifies the right format for the input. The dataset has to be divided into predictor variables and a response variable. The predictor variables have to be made available either as a matrix or a data frame; the response has to be a numeric vector. Moreover the response has to be continuous. Binary or count responses are not allowed. The other input arguments are ignored for the moment and discussed later.

The dataset *Ozone* consists of 9 predictor variables and the response variable *upo3* which is stored in the first column:

```
> data(ozone,package="gss")
> xozone <- ozone[-1]
> yozone <- ozone$upo3
```

Now predictors `xozone` and response `yozone` can be given as input to `quantregForest`:

```
> qrfozone <- quantregForest(xozone,yozone)
```

The function `quantregForest` returns an object of class `quantregForest`, for which `print`, `plot` and `predict` methods are available. In `qrfozone` the following information is stored:

```
> print(qrfozone) # or simply just type qrfozone

Call:
 quantregForest(x = xozone, y = yozone)

                     Number of trees: 100
No. of variables tried at each split: 3
```

The command `print` produces the output given above. It provides a summary over the input which was given to `quantregForest`.

```
> qrfozone$origNodes
```

This command returns the calculated nodes for the grown forest (330 values for each of the 100 trees), observation 1 lies in the nodes in row 1, observation 2 in the nodes in row 2 etc. This is a difference to `randomForest`, where per default these values are not saved.
To obtain the values of the response variable used to fit the model we can write:

```
> qrfozone$origObs
```

This information is saved because it is used by the prediction algorithm.
The input parameter `ntree` determines how many trees are grown in the random forest on which quantile regression forests are based on. Empirical evidence suggests that the performance of the prediction remains good even when using only few trees. Therefore the default setting in the current version is 100 trees. More parameters for tuning the growth of the trees are `mtry` and `nodesize`. `mtry` sets the number of variables to try for each split when growing the tree. The same default is used as in `randomForest`, which is one third of the number of predictors. The parameter `nodesize` fixes the minimal number of instances in each terminal node, determining how many observations at least lie in the same node. The default setting here is 10. As for the number of trees, varying this parameter does in general not make a big difference (quantile regression forests are very stable concerning these parameters) thus the default setting can often be used.
In analogy to `randomForest` there exists an input argument `importance` to compute a variable importance measure and related to that an input argument `quantiles`. These are discussed further in Section 4.
To summarize, growing quantile regression forests is basically the same as growing random forests but more information on the nodes is stored. The most important part of the package is the prediction function which is discussed in the next section.

3 Prediction

The prediction function in the current use has five input arguments with the following defaults:

```
> predict(object, newdata=NULL,
+                 quantiles=c(0.1,0.5,0.9),
+                 all=FALSE, obs=1)
```

`object` has to be of class `quantregForest`, i.e. a quantile regression forest grown as described in Section 2.
Consider first the simple input

```
> predict(qrfozone)
```

where only the input `object` is set as the quantile regression forest grown for the *Ozone* data. In this form, the function `predict` performs out-of-bag prediction on the dataset *Ozone*, i.e. for each of the grown trees prediction for the data points which were not used for fitting the tree is done (no new data is involved). The output is a 330×3 matrix with the predicted *0.1, 0.5* and *0.9* quantiles. If we only want to calculate one specific quantile, for example the median, we could type:

```
> predict(qrfozone,quantiles=0.5)
```

The input for `quantiles` can be an arbitrary vector with values between 0 and 1. The default is *(0.1, 0.5, 0.9)* as seen above.
To predict quantiles for new data the input `newdata` has to be changed to a matrix or data frame with new observations in the rows.
Consider thus prediction for *Ozone* when only growing the quantile regression forest on the first 329 data points and use the 330th observation as new sample point:

```
> xozone329 <- ozone[-330,-1]
> yozone329 <- ozone$upo3[-330]
> qrfozone329 <- quantregForest(xozone329,yozone329)
> predict(qrfozone329,quantiles=0.5,newdata=ozone[330,-1])

     quantile= 0.5
[1,]            4
```

Per default only one observation per node is used for prediction. This can be set with the input argument `all` with default `all=FALSE` (one observation per node used) and when setting `all=TRUE`, all observation per node are used. The use of only one observation per node is of advantage especially when working with large datasets since the algorithm can be very slow otherwise. Numerical experiments suggest that the performance remains good.
Nevertheless, this option should be handled with care in cases with big datasets

and few new sample points as input for `newdata` where setting `all=FALSE` may be significantly slower than choosing `all=TRUE`.
The advantage of the fast implementation using only one observation per node for prediction can for example be seen when performing out-of-bag prediction on the large dataset *CASP* with response variable *RMSD* and 8 predictors when $10'000$ observations are used to fit the model and the quantiles for $1'000$ new sample points are predicted:

```
> casp=read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/00265/CASP.csv")
> xcasp <- casp[1:10000,-1]
> ycasp <- casp$RMSD[1:10000]
> qrfcasp <- quantregForest(xcasp,ycasp)

> system.time(predict(qrfcasp,quantiles=0.5,newdata=casp[10001:11000,-1]))

   user  system elapsed
  10.95    1.17   12.21
```

As comparison consider the same setting but now all observations are used for prediction (`all=TRUE`):

```
> system.time(predict(qrfcasp,quantiles=0.5,newdata=casp[10001:11000,-1],all=TRUE))

   user  system elapsed
  35.53    0.48   36.03
```

We observe that the execution time in this case is indeed much longer when using all observations instead of only one observation per node for prediction.
Finally, there exists the option to change the number of observations per node used for prediction by the input argument `obs`, which is only available for default setting `all=FALSE`. The value 3 is chosen as example:

```
> predict(qrfozone,quantiles=0.5,obs=3)
```

The default of `obs` is 1. Numerical experiments have shown that the results are already satisfying for `obs=1`, so it is recommended to use the default here.
The question remains when to use which method, i.e. use all observations for prediction or only several observations per node. It depends mainly on the number of observations in the training dataset (n_{train}) and the number of observations in test data (n_{test}). To summarize, consider the following cases:

- Predict the quantiles for n_{train} small and arbitrary n_{test} or out-of-bag prediction: `all=FALSE` (default) or `all=TRUE` can be used since the results do not differ much and both options are fast.
- Predict the quantiles for n_{train} large and n_{test} small (e.g. single new data point): Use `all=TRUE` since it is much faster.
- Predict the quantiles for n_{train} large and n_{test} large or out-of-bag prediction: Use `all=FALSE` (default) since it is remarkably faster.

```
> plot(qrfozone)
```

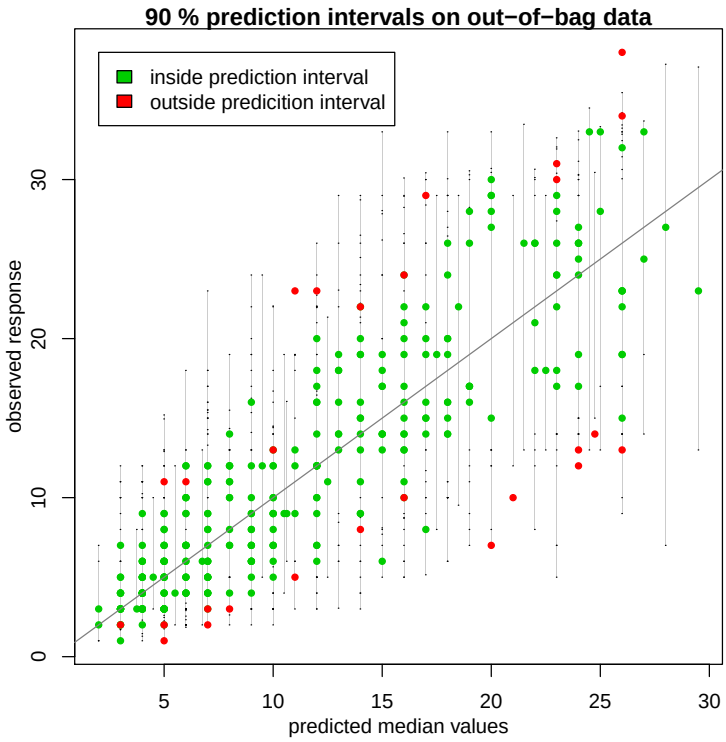

Figure 1: Estimated 90 % prediction intervals for the *Ozone* dataset

A method to plot data for class `quantregForest` is also made available in the package. With the command `plot` the 90 % prediction interval for out-of-bag prediction is plotted. The input `object` has to be of class `quantregForest`, see Figure 1 for the *Ozone* dataset. The red dots mark the observations which lie outside the prediction interval, the green ones lie inside. The grey bars represent the prediction intervals.
Again, per default `all=FALSE` is used which is reasonable since out-of-bag prediction is performed. Nevertheless, the option `all=TRUE` exists although it is recommended to use the default. The option to change the number of observations used for prediction is also available, but again it is recommended to use the default.

```
> plot(qrfozone,all=TRUE)
> plot(qrfozone,obs=3)
```

4 Variable Importance

A variable importance measure for quantile regression forests can be obtained by the following steps:
First, when growing the tree with `quantregForest` the additional option `importance` has to be set to `TRUE`, e.g. for the dataset *Ozone*:

```
> qrfozone <- quantregForest(xozone,yozone,importance=TRUE)
```

The quantiles for which the measure should be computed can be set by the input argument `quantiles`. The default setting is `quantiles=c(0.1,0.5,0.9)`. Arbitrary vectors with values between 0 and 1 are allowed.
The importance measure can then be extracted either using

```
> qrfozone$importance
```

or the function `importance`, both yield the same output:

```
> importance(qrfozone)
```

```
     quantile= 0.1 quantile= 0.5 quantile= 0.9
vdht      1.501621     0.9888700      5.893826
wdsp      1.848444    -0.5006413      1.057341
hmdt      5.405922     9.8663577     12.283538
sbtp     15.315859    35.1979809     42.123174
ibht     20.115267    11.7133518     11.468469
dgpg      5.530880     7.1951674      9.964748
ibtp     13.222175    32.7651123     52.684763
vsty      6.515244     5.0602011      2.775703
day       6.176072     6.0697588      4.002195
```

The quantiles for which the measure should be returned can be set by the input argument `quantiles`, e.g.

```
> importance(qrfozone,quantile=0.5)
```

Per default the measures for all computed quantiles are returned. Only values for which the measure was already computed with `quantregForest` are allowed.
To visualize the variable importance measure the function `varImpPlot.qrf` can be used, see Figure 2.
Again, the quantiles for which the measure should be visualized can be set by the input argument `quantiles`. The same conditions need to be fulfilled as for `importance`.
Additional options determining the design of the plot are available: `sort` determines if predictors should be sorted increasingly by the value of their importance measure, `which.sort` determines which quantile is relevant for the sorting and `symbols` and `color` determine if the plotted points should be symbols and/or coloured, consider e.g. the following possible input:

```
> varImpPlot.qrf(qrfozone,quantiles=c(0.5,0.9),
+ symbols=FALSE,color=TRUE,which.sort=2)
```

For more details refer to the respective help files.

```
> varImpPlot.qrf(qrfozone)
```

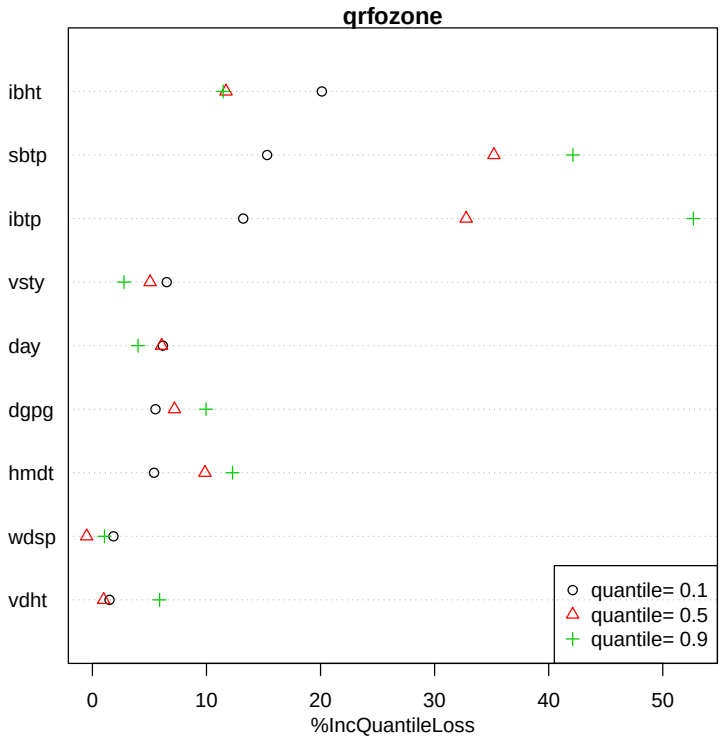

Figure 2: Variable importance plot for *Ozone*

5 Applications

Quantile regression can be used in many different problems where one is not only interested in the conditional mean of a distribution but in the whole distribution. Two applications will be described here as they are highlighted in the original paper on quantile regression forests by Meinshausen (Meinshausen, 2006).

Outlier Detection The first application discussed here is outlier detection. For illustrating this purpose, consider the *Ozone* dataset, where three data points are manipulated, those with indices 50, 150 and 200. Then the 99% quantiles are predicted and the indices of observations larger than the respective quantile are given as output.

```
> ozoneoutliers <- ozone
> ozoneoutliers$upo3[c(50,150,200)] <- ozoneoutliers$upo3[c(50,150,200)]*100
> x <- ozoneoutliers[-1]
> y <- ozoneoutliers$upo3
> qrf <- quantregForest(x,y)
> which(y>predict(qrf,quantile=0.99))

[1]  50 128 150 200 220 252
```

Prediction Intervals As already seen in the Figure 1, it is quite easy to plot prediction intervals when working with quantile regression forests. A main conclusion that can be achieved by visualizing the prediction intervals is the reliability of a prediction, as the length of the prediction intervals differs strongly. As an example consider the *CASP* dataset, where the model is fitted on 10′000 observations and the quantiles for 500 points are predicted. The result is displayed in Figure 3 where we can conclude that for some observations the prediction is more reliable than for others.

```
> quantiles <- c(0.05,0.5,0.95)
> quant <- predict(qrfcasp,quantiles=quantiles,newdata=casp[11000:11500,-1])
> z <- quant[,3]-quant[,1]
> or <- order(z)
> ynew <- casp$RMSD[11000:11500]
> n <- length(ynew)
> # center and order the quantiles
>
> med <- quant[or,2]-quant[or,2]
> upp <- quant[or,3]-quant[or,2]
> low <- quant[or,1]-quant[or,2]
> ytrain <- ynew[or]-quant[or,2]
```

```
> # Plot the centred observations and the prediction intervals
>
> plot(1:n,ynew[or]-quant[or,2],pch=20,xlab="ordered samples",
+ ylab="observed response and prediction
+ intervals(centred)",type="n",main="90% prediction intervals")
> dist <- 0.01
> for (i in 1:n){
+   polygon( c(i-dist,i+dist,i+dist,i-dist),
+   c(upp[i],upp[i],low[i],low[i]) ,col=rgb(0.8,0.8,0.8) ,border=NA)
+ }
> for (i in 1:n){
+   lines(c(i-dist,i+dist) , c(upp[i],upp[i]) )
+   lines(c(i-dist,i+dist) , c(low[i],low[i]) )
+ }
> inpred <- (ytrain<= upp) & (ytrain>=low)
> for (i in  1:n) points(i,ynew[or[i]]-quant[or[i],
+ 2],col=as.numeric(inpred)[i]+2,pch=20)
```

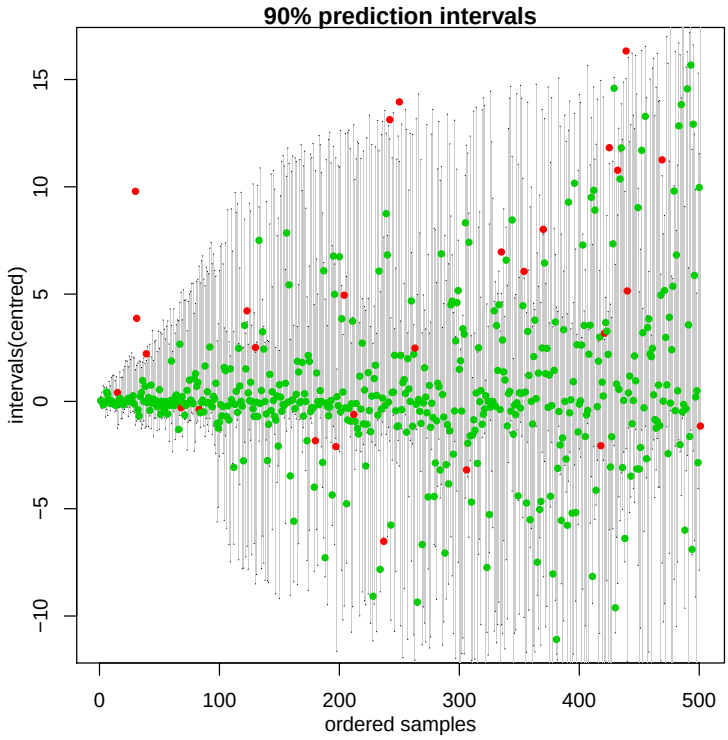

Figure 3: Prediction intervals ordered according to their length for 500 samples of the *CASP* dataset.

References

Gu, C. (2014). Smoothing spline anova models: R package gss. *Journal of Statistical Software 58*(5), 1–25.

Lichman, M. (2013). UCI machine learning repository.

Meinshausen, N. (2006). Quantile regression forests. *J. Mach. Learn. Res. 7*, 983–999.