Ludwig-Maximilians-Universität München

Institut für Statistik

Prediction Inference with Ensemble Methods

Prognoseinferenz mit Ensemble-Methoden

Diplomarbeit
von
Andreas Mayr

Betreuung: Dipl. Stat. Nora Fenske
Prof. Dr. Torsten Hothorn
Abgabetermin: 15. Januar 2010

Acknowledgements

I am deeply grateful for the support of many people that helped me during these last months while I worked on this project. A special thank you to....

- **Dipl. Stat. Nora Fenske** for her excellent supervision, her accurate proof-reading, her support, her commitment and her patience. I also want to thank her for making all her R code about quantile regression available to me.
- **Prof. Dr. Torsten Hothorn** for his excellent supervision and for the trust he put in me by offering me this thesis. I also want to thank him for looking through much of the R code that led to the simulations and for the essential input around the questions of how to interpret and evaluate prediction intervals.
- **Prof. Dr. Helmut Küchenhoff** and his team of the statistical consulting unit (especially **Juliane Manitz** and **André Klima**) for giving me the great opportunity to test our methods in practice with the movie data.
- **Elisabeth Waldmann** and **Juliane Manitz** for proofreading and their fair comments.
- **Dipl. Stat. Michael Obermeier** and **Birgit Oppolzer** for important LaTeX tips and for sharing the office with me while I worked on this thesis.
- **Dr. Rebecca Schennach-Wolff** and **Dr. Florian Seemüller** for the 3 pm coffee therapy every day and their moral support.
- **Dipl. Stat. Benjamin Hofner** for encouraging me to select this topic and mailing me his thesis which I allowed me to use as a guideline concerning many aspects.
- **My friends**, **my family** and **Belén** for their constant support and encouragement.

Andreas Mayr
Munich, January 2010

Contents

Chapter 1

Introduction

Prediction is very difficult, especially about the future.

Niels Bohr (1885-1962)

This famous quotation by the Danish physicist who won the Nobel Prize in 1922 reflects the widespread disbelief in predictions concerning future events. Yet one important goal of modern statistics is not only the analysis of relationships between certain variables but also to quantify and to use the predictive power of data for future or unobserved observations. Nevertheless, these two aspects of data analysis are often highly linked and can hardly be discussed separately.

The question of relationship between variables is surely the most common problem in data analysis. Does the new medical treatment have a significant influence on the patient's blood pressure? One of the most common ways to answer these questions is the usage of regression analysis. Typically, the aim of a regression model is to explain a response variable Y (`blood pressure`) by one or more predictor variables X (`treatment`). One advantage of a regression model is the possibility to adjust for differences between treatment groups (`verum` vs. `placebo`) in other variables (`age`, `weight`). The result in the standard form is a linear additive model explaining the conditional mean of the response variable Y given the observed $X = \boldsymbol{x}$ as follows:

$$\mathbb{E}(Y|X = \boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\beta} = \beta_0 + \beta_1 \cdot \texttt{treatment} + \beta_2 \cdot \texttt{age} + \beta_3 \cdot \texttt{weight}$$

In this case, let us assume that we have data of a clinical trial including 100 patients: $\boldsymbol{y} = (y_1, ..., y_{100})^\top$ and the explaining variables respectively. As the conditional mean minimizes the expected squared error loss, the vector $\boldsymbol{\beta}$ of all parameters can be estimated by ordinary least squares (OLS) as:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y},$$

with the design matrix $\boldsymbol{X}$ containing the values of `treatment`, `age` and `weight` of all 100 patients.

The effect of the treatment can now be interpreted via β_1, adjusted for `age` and `weight` of the patient. The test of the hypothesis $H_0 : \beta_1 = 0$ answers the question about the relationship between the new treatment and the health of the patient in terms of his blood pressure. Once this effect is clear, another question arises: How is the predictive power of this model based on the clinical trial including 100 patients? Can we give an estimation about a new patient's blood pressure after treatment, given only his data at admission to hospital? This is obviously the case, one only needs the $\hat{\boldsymbol{\beta}}$ and the patients treatment, age and weight. Hence, we do make predictions about the future. Nevertheless, how reliable is this prediction?

The best way to answer this question are prediction intervals instead of a single point estimator. The estimated response $\mathbb{E}(Y|X = \boldsymbol{x}_{\text{new}})$ for a new observation is of obvious interest, but the accuracy of the procedure can be better shown by a prediction interval $\text{PI}(\boldsymbol{x}_{\text{new}}) = [a, b]$, where a denotes the lower boundary for new observations with the observed predictor variables $\boldsymbol{x}_{\text{new}}$ and b the upper one. The size of the prediction interval already includes information about the reliability of the point estimator.

The standard form to do this would be to calculate a standard regression model for the conditional mean, and use the assumption of an underlying distribution function to calculate a symmetric prediction interval for this point estimator. This works fine as long as the assumption about the distribution is correct and we have valid information about the variance of the estimation method.

In this thesis, we want to focus on another nonparametric approach, which can be used for every continuous variable Y, without assuming any distribution and without having to estimate the variance of $\hat{Y}$:

Meinshausen (2006) came up with the idea of using quantile regression to construct prediction intervals for new observations. In contrast to standard regression analysis, quantile regression (Koenker, 2005) does not estimate the conditional mean $\mathbb{E}(Y|X = \boldsymbol{x})$ of the response variable, but the conditional τ-quantile $Q_\tau(Y|X = \boldsymbol{x})$ for a given possible multidimensional $X = \boldsymbol{x}$. Following the definition of quantiles, the probability of the response being smaller than $Q_\tau(Y|X = \boldsymbol{x})$ is τ. Meinshausen proposed using this to construct 95% prediction intervals depending on $\boldsymbol{x}_{\text{new}}$. In the following, the conditional quantiles are treated as functions in x, given as:

$$Q_\tau(Y|X = \boldsymbol{x}) := q_\tau(\boldsymbol{x})$$

A new observation $\boldsymbol{x}_{\text{new}}$ is then plugged-in in the conditional quantiles $q_\tau(\cdot)$ that work as boundaries :

$$\text{PI}(\boldsymbol{x}_{\text{new}}) = [q_{0.025}(\boldsymbol{x}_{\text{new}}), q_{0.975}(\boldsymbol{x}_{\text{new}})]$$

The main advantage toward classical prediction intervals focusing on the conditional mean is the fact that we do not have to assume anything about Y, especially no underlying distribution function as we estimate the boundaries by quantile regression.

Going back to the example above, this means that one can compute an interval for the prediction of a new patient's blood pressure based on his data at admission and the proposed treatment. The relevance of this information is obvious. Instead of a single point predictor as the conditional mean, this procedure gives an interval for the patient's blood pressure after treatment. Hence, we do not only make a prediction, but also include information about this predictions accuracy and therefore its reliability.

In this thesis, we want to focus on prediction inference using ensemble methods. Ensemble methods combine multiple models into one final prediction in order to achieve a better predictive performance. Those methods have their roots in the machine learning community. The idea is that an algorithm can improve a prediction technique by iteratively applying it and combining the single results at the end. The final combination tends to be more precise than using the technique just once. The algorithm therefore is learning from the data step by step, as the performance is improving.

Meinshausen's approach was based on combinations of regression trees which he succeeded to adopted to estimate the conditional distribution function, and therefore the quantiles. We will further analyze this magnificent idea and focus also on different approaches to estimate the conditional quantiles.

In a further step we analyze the coverage of the resulting prediction intervals estimated by different ensemble methods. We will focus on component-wise boosting for quantile regression (Fenske et al., 2009; Bühlmann and Hothorn, 2007) and compare it to Meinshausen's approach of quantile regression forests.

This thesis will be structured as follows:

Chapter 2 compares prediction intervals with confidence intervals, and focuses on adequate interpretations. This chapter is essential, as also a seemingly standard procedure as prediction intervals bears some severe pitfalls in its correct interpretation and usage that can be easily overlooked.

Chapter 3 gives an overview on some theoretical aspects of quantile regression and basic estimation schemes.

Chapter 4 focuses on random forests as one tool to estimate the conditional distribution function and, therefore, the quantiles.

Chapter 5 presents component-wise boosting as a powerful algorithm to estimate linear and additive nonlinear quantile regression, including variable selection for high dimensional data.

Chapter 6 uses various simulation studies to analyze the coverage of the proposed intervals in different setups for both estimation techniques.

Chapter 7 presents an interesting application how ensemble methods can be used for prediction inference. The task is to forecast the amount of people going to the movies in the first week after the release of a new film.

Notation

In this thesis we will stick to a rather unusual notation that in our view combines the advantages of more common approaches.

We will denote random variables as capital letters. In regression settings the response variable is Y and the possibly multidimensional predictor variable is X. We use therefore X for a random variable with possible multiple components (e.g. `treatment`, `age`, `sex` as in the example used in the introduction).

If we look at a future or unobserved realization of a random variable, we use a capital letter to emphasize that it is still a random variable. In a regression setting the realizations Y_i of Y are still random variables, and are therefore denoted with capital letters.

Only when we analyze a given sample of observed realizations, they are not random but fixed. Therefore we go over to the vector notation $\boldsymbol{y} = (y_1, ..., y_n)^\top$ or $\boldsymbol{x}_i = (x_1, ..., x_p)^\top$ where n is the sample size and p the dimensionality of X. Therefore, in a regression setting an observed sample of the random variables (Y, X) is denoted as $(y_1, \boldsymbol{x}_1), ..., (y_n, \boldsymbol{x}_n)$.

This distinction between unobserved realizations $(Y_1, ..., Y_n)$ of a random variable Y, and observations $\boldsymbol{y} = (y_1, ..., y_n)^\top$ has the advantage that one can clearly distinguish if we refer to a fixed value or to a random variable. Let us assume a gaussian distributed response with $Y \sim N(\mu, \sigma^2)$. If we look at the variance $\mathbb{V}(Y_i)$ of a realization, as a result of Y_i being identical distributed as the response $\mathbb{V}(Y_i) = \sigma^2$. If the realization is part of an observed sample $\mathbb{V}(y_i) = 0$, as y_i is fixed.

When we want to condition the response on a given observation of X, we denote this with $Y|X = \boldsymbol{x}$. This may look unappropriate at first glance, as it combines a random variable with a vector. But in fact it is just a result of the described definitions above, as X can also be multidimensional. Only in setups with only one predictor variable we will use the much more common notation $Y|X = x$.

As for the conditional mean $\mathbb{E}(Y|X = \boldsymbol{x}) := \mu(\boldsymbol{x})$ we also use a short form for conditional quantiles $Q_\tau(Y|X = \boldsymbol{x}) := q_\tau(\boldsymbol{x})$ which not only reduces the notational effort but also emphasizes more the interpretation of conditional quantiles as functions in $\boldsymbol{x}$.

Unless stated differently, we will therefore stick to following notation:

Basic notations

variables		x, y, ν, α
scalars		p,l,$n \in \mathbb{R}$
vectors		$\boldsymbol{x} = (x_1, ..., x_n)^\top$
matrices		$\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)$

Statistics notations

random variables		Y, X
future or unobserved realizations		$(Y_1, ..., Y_n)$
observations		$\boldsymbol{y} = (y_1, ..., y_n)^\top$
probability density function		$f(y)$
distribution function		$F_Y(y)$
expected mean		$\mathbb{E}(Y)$
variance		$\mathbb{V}(Y)$
τ-quantile		$Q_\tau(Y)$

Conditional notations
(multidimensional X)

expected conditional mean		$\mathbb{E}(Y|X = \boldsymbol{x})$
expected conditional mean	short form	$\mu(\boldsymbol{x})$
conditional quantile		$Q_\tau(Y|X = \boldsymbol{x})$
conditional quantile	short form	$q_\tau(\boldsymbol{x})$
conditional density function		$f(y|\boldsymbol{x})$
conditional distribution function		$F_{Y|X}(y|\boldsymbol{x})$

Software

All analyses in this thesis are carried out using the statistical programming environment R (R Development Core Team, 2009) in its version `2.9.1`. This software and every add-on package used and described in this thesis is available at `http://www.cran.r-project.org/`.

Chapter 2

Prediction Intervals

In this thesis, the main aspect will be the estimation of nonparametric prediction intervals for future or unobserved values of a continuous variable. We therefore will present quantile regression as a powerful statistical tool to model more than the expected mean of a conditional distribution. The target of quantile regression is to model the conditional quantiles.

We will then focus on ensemble methods that can be adopted to estimate quantile regression, and will use them to construct prediction intervals. Before we start to present different estimation algorithms and analyze the coverage of such intervals in simulation studies, we want to repeat some major properties of general prediction intervals that can be easily overlooked.

In Section 2.1. we will shortly repeat the difference between a classical parametric approach to estimate prediction intervals and a nonparametric approach. In Section 2.2 we focus on the difference between prediction intervals and confidence intervals.

Another important point is the correct interpretation of prediction intervals, which is linked to the problem of finding adequate ways to prove their consistency. In standard literature it is simply noted that the probability of a new observation falling in a correctly specified $\mathrm{PI}_{1-\alpha}(\cdot)$ is $1-\alpha$ (Fahrmeir et al., 2007). This sounds reasonable and is certainly correct. Yet we also discovered a severe pitfall in the interpretation of prediction intervals for new observations. In Section 2.3, we therefore will present a conditional as well as a heuristic interpretation of $\mathrm{PI}_{1-\alpha}(x_{\mathrm{new}})$.

2.1 Parametric and nonparametric intervals

We already mentioned that in this thesis we will present nonparametric tools to estimate prediction intervals. The term *nonparametric* in this case refers to the fact that these tools do not assume any underlying distribution for the variables of interest. But the classical way is to model parametric prediction intervals, based on standard regression analysis.

2.1.1 The classical linear model

In regression analysis, if the influence of a predictor variable X on the response Y is modeled as a linear effect, and the error term is assumed to be normally distributed, we call this the classical linear model. We therefore have random variables $(X_1, Y_1), ..., (X_n, Y_n)$ that are related following the model:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

If we now go over to observed values of this random variables with X being possibly multidimensional $(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_n}, y_n)$ we can formulate the model as:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

where $\boldsymbol{X}$ has the form:

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x_1}^\top \\ \vdots \\ \boldsymbol{x_n}^\top \end{pmatrix}$$

Hence, the dimension of $\boldsymbol{X}$ is $(n \times (p+1))$ where n is the number of observations, and p the number of explaining variables.

Then the parameter vector $\boldsymbol{\beta}$ has the form:

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$$

The vector $\boldsymbol{\beta}$ therefore reflects the linear influence of the predictor variables on the conditional mean. As the expected mean of the error term is zero, one can conclude:

$$\mathbb{E}(Y|X = \boldsymbol{x}) = \beta_0 + \beta_1 \cdot x_1 + ... + \beta_p \cdot x_p$$

β_0 is the expected Y for an observation with $x_1 = ... = x_p = 0$. β_j for $j = 1, ..., p$ is the difference in the expected response level for two observations $x_j = x_j^*$ and $x_j = x_j^* + 1$.

Hence, estimating $\boldsymbol{\beta}$ means estimating $\mathbb{E}(Y|X = \boldsymbol{x})$. We will also use the abbreviation $\mathbb{E}(Y|X = \boldsymbol{x}) := \mu(\boldsymbol{x})$ which reflects more the idea that the result of the estimation will be some kind of regression curve, depending on $\boldsymbol{x}$.

Assumptions for the classical linear model:

1. The matrix $\boldsymbol{X}$ has full rank, therefore the columns of $\boldsymbol{X}$ are linearly independent.
2. $\mathbb{E}(\boldsymbol{\varepsilon}) = \mathbf{0}$
3. $\text{Cov}(\boldsymbol{\varepsilon}) = \mathbb{E}(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top) = \sigma^2\boldsymbol{I}$
4. $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2\boldsymbol{I})$

Where $\boldsymbol{I}$ denotes the identity matrix. The third assumption ($\text{Cov}(\boldsymbol{\varepsilon}) = \sigma^2\boldsymbol{I}$) means nothing less than that the observations are independent from one another and that the variance of the error term is the same for all observations. It therefore may not depend on neither the predictor variables nor the value of the response. This is called *homoscedasticity.*

2.1.2 Parametric prediction intervals

The estimation of parametric prediction intervals always follows the same scheme:

- Predict the new observation Y_{new} by using a point-estimator $\hat{\mathbb{E}}(Y_{\text{new}}|X = \boldsymbol{x}_{\text{new}})$
- Construct a prediction interval around the point estimator by using information about the distribution of Y_{new} and $\mathbb{V}(Y_{\text{new}} - \hat{Y}_{\text{new}})$

For a new observation $\boldsymbol{x}_{\text{new}}$ one can use $\hat{\boldsymbol{\beta}}$ that was estimated based on a previous sample, for a prediction:

$$\hat{Y}_{\text{new}} = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_{1\text{new}} + ... + \hat{\beta}_p \cdot x_{p\text{new}}$$

We estimate the unobserved new realization of Y by using the information of a previous observed sample $(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_n}, y_n)$ to estimate $\hat{\boldsymbol{\beta}}$ and therefore $\hat{\mu}(\boldsymbol{x})$. We then plug in the observed predictor variables $\boldsymbol{x}_{\text{new}}$ to estimate $\hat{Y} = \hat{\mu}(\boldsymbol{x}_{\text{new}})$. If the assumptions are fulfilled, the expected bias $\mathbb{E}(Y_{\text{new}} - \hat{Y}_{\text{new}})$ is 0.

To construct a prediction interval around $\hat{Y}_{\text{new}}$ we do also need an estimation for $\mathbb{V}(Y_{\text{new}} - \hat{Y}_{\text{new}})$ and an underlying distribution function. For $\mathbb{V}(Y) = \sigma^2$, the variance of the bias can be estimated depending on $\boldsymbol{x}_{\text{new}}$ and the predictor variables $\boldsymbol{X}$ of the original data (Fahrmeir et al., 1996):

$$\widehat{\mathbb{V}}(Y_{\text{new}} - \hat{Y}_{\text{new}}) = \sigma^2(1 + \boldsymbol{x}_{\text{new}}^\top(\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{x}_{\text{new}})$$

If the error term $\boldsymbol{\varepsilon}$ is normally distributed, it holds true that the standardized bias is distributed as Students's t distribution with $(n - p)$ degrees of freedom:

$$\frac{\hat{Y}_{\text{new}} - Y_{\text{new}}}{\sqrt{\widehat{\mathbb{V}}(Y_{\text{new}} - \hat{Y}_{\text{new}})}} \sim t(n-p)$$

We can now use the information of the underlying distribution function and the variance of the bias to construct a prediction interval for $\hat{Y}_{\text{new}}$.

For a new observation, we can therefore conclude that the prediction interval

$$\text{PI}(\boldsymbol{x}_{\text{new}}) = \left[\hat{Y}_{\text{new}} \pm t_{1-\alpha/2}(n-p) \cdot \sqrt{\widehat{\mathbb{V}}(Y_{\text{new}} - \hat{Y}_{\text{new}})}\right]$$

covers the true Y_{new} in $(1-\alpha)\cdot 100\%$ of the cases. The information of $\boldsymbol{x}_{\text{new}}$ is hidden inside the estimation of $\hat{Y}_{\text{new}}$ and $\widehat{\mathbb{V}}(Y_{\text{new}} - \hat{Y}_{\text{new}})$.

This is the classical form to construct prediction intervals of $\hat{Y}_{\text{new}}|\boldsymbol{x}_{\text{new}}$. It is mainly focusing on the bias of the model, and depends on the assumed distribution function and the correct estimation of the variance.

For a more flexible approach, we will concentrate on conditional quantiles that can be specified without assuming a distribution.

2.1.3 Nonparametric prediction intervals

The basic idea of parametric prediction intervals was to model the expected mean of a new observation, and then use the information of uncertainty of the assumed underlying distribution function to construct intervals around this point estimator. There is nothing wrong with that, as long as the assumed distribution is at least asymptotically correct.

If we cannot assume any distribution function, we will need more flexible methods to estimate prediction intervals. The solution presented in this thesis is to use quantile regression to model the boundaries of the intervals directly. Instead of making a point prediction and account for the uncertainty by adding and subtracting something on both sides, we will directly estimate the points of the response where most of the future observations should lie between.

The difference can be made clear in a small example: Imagine we want to estimate a 95% prediction interval for the weight of a female baby 12 weeks after birth in a certain country. The parametric way to do this is to estimate the expected mean of the babies weight based on a sample. Afterwards we assume that the weight in some form should be at least asymptotically normaly distributed and therefore use this information and an estimation of the prediction error to build an symmetric interval around this point estimator.

But let us imagine that in this country there are groups of the population in which families cannot afford enough food for their children. Most of the babies are well fed, but some suffer from underweight following malnutrition. This sad fact has to be taken into account in the estimation process. The assumed symmetric normal distribution seems to be unlikely. One solution is to chose another distribution. Another one is to use the empirical distribution given by the sample to estimate directly the boundaries. For a sample of 1000 babies we therefore could use the quantiles, hence the weight of the 25th and the 975th baby of the ordered sample as boundaries. Therefore, we do not assume anything about the distribution, and base our interval only on the information given in the sample.

If we want to include predictor variables in the estimation, we can use conditional quantiles instead and estimate the boundaries based on quantile regression. We will introduce quantile regression and conditional quantiles in detail in Chapter 3.

In this thesis, we will present two major tools to estimate conditional quantiles by ensemble methods. Quantile regression forests (Chapter 4) as well as component-wise boosting for quantile regression (Chapter 5) are high performing prediction tools based on the aggregation of single base learners.

Ergo, we do have two algorithms to model the influence of one or more predictor variables X on the conditional quantiles $Q_\tau(Y|X = \boldsymbol{x})$ of the response variable Y. We are able to include predictor variables in the estimation of the intervals, this way we condition and adjust our prediction on variables that have an effect on the response.

We therefore utilize the definition of quantiles: The probability of a random variable Y being smaller than the corresponding τ-quantile is τ:

$$P(Y < Q_\tau(Y|X = \boldsymbol{x})|X = \boldsymbol{x}) = F_{Y|X}(Q_\tau(Y|X = \boldsymbol{x})) = \tau$$

For a new observation $\boldsymbol{x}_{\text{new}}$ we interpret the conditional quantiles as functions of $\boldsymbol{x}$ which results in the following prediction interval of the level $1 - \alpha$:

$$\text{PI}_{1-\alpha}(\boldsymbol{x}_{\text{new}}) = [q_{\alpha/2}(\boldsymbol{x}_{\text{new}}), q_{1-\alpha/2}(\boldsymbol{x}_{\text{new}})]$$

This idea was already presented by Meinshausen (2006), who estimated the boundaries based on quantile regression forests. To our knowledge this was the first approach toward prediction intervals based on quantile regression by ensemble-methods. The main advantage of this approach is that it does not depend on any assumed distribution of Y. It therefore is a nonparametric tool to estimate prediction intervals.

2.2 On the difference between prediction and confidence intervals

Although we are generally interested in constructing nonparametric intervals based on conditional quantiles, the following remarks hold true basically for all kinds of correctly specified prediction intervals. In the following sections we will denote prediction intervals as PI($\cdot$) whereas confidence intervals are CI($\cdot$).

2.2.1 Intervals for Y

Prediction intervals and confidence intervals have one important thing in common: As they depend on a sample, their boundaries should be seen as random variables. Though the main difference is that confidence intervals are constructed to cover a fixed but unknown value or parameter, whereas prediction intervals should cover a new observation of Y that therefore itself is a random variable.

Assume we have a realizations of a continuous random variable Y with $\mathbb{E}(Y) = \mu$ and $\mathbb{V}(Y) = \sigma^2$:

$$(Y_1, ..., Y_n) \quad \text{distributed as} \quad Y$$

In case of a CI we are interested in an unknown but fixed value like μ that could be estimated by the mean.

$$\hat{\mu} = \frac{\sum_{i=1}^{n} Y_i}{n}$$

The size of the corresponding confidence interval will depend on the assumed distribution of Y and on the variance of the estimation:

$$\mathbb{V}(\hat{\mu}) = \mathbb{V}\left(\frac{\sum_{i=1}^{n} Y_i}{n}\right) = \frac{\sigma^2}{n}.$$

If we want to construct a prediction interval on the same sample, the aim is to cover Y_{new} that itself is a realization of Y. A good prediction of $\hat{Y}_{\text{new}}$ certainly is $\hat{\mu}$, but the PI must not only include the uncertainty of estimating $\hat{\mu}$, but has also to account for the variance that Y implies. Hence, the size of the parametric prediction interval depends on the distribution of Y and on

$$\mathbb{V}(Y_{\text{new}} - \hat{\mu}) = \sigma^2 + \frac{\sigma^2}{n}.$$

This result does not only imply a larger size of a PI for $\hat{Y}_{\text{new}}$ compared to a CI for μ, it also gives kind of a threshold for the size of the prediction interval.

If we could increase the size of the sample $n \to \infty$, we could reduce the size of the confidence interval for $\hat{\mu}$ as

$$\mathbb{V}(\mu - \hat{\mu}) = \frac{\sigma^2}{n} \overset{n\to\infty}{\longrightarrow} 0.$$

Yet the variance of the prediction error we make by estimating $\hat{Y}_{\text{new}} = \hat{\mu}$ will not vanish if we increase the sample size n:

$$\mathbb{V}(Y_{\text{new}} - \hat{\mu}) = \sigma^2 + \frac{\sigma^2}{n} \overset{n\to\infty}{\longrightarrow} \sigma^2.$$

This implies one important difference between confidence intervals and prediction intervals: the size of a PI even for big n will always depend heavily on $\mathbb{V}(Y)$. This also holds true for nonparametric PI based on quantiles:

$$[\hat{Q}_{\alpha/2}(Y), \hat{Q}_{1-\alpha/2}(Y)]$$

For large n the estimation of this quantiles will get better, but as the distance between the true quantiles is given by the distribution and the variance of Y the size of the PI, from a certain point on, cannot be further reduced by a bigger sample size.

2.2.2 Conditional intervals for $Y|X = \boldsymbol{x}$

If we now pass on to conditional intervals, we enter the regression framework again, with one or more predictor variables X and a continuous response Y. We want to construct intervals for $Y|X = \boldsymbol{x}$ that not only depend on the samples of Y and X, but also on the observation $\boldsymbol{x}$. We will denote the resulting intervals as functions of $\boldsymbol{x}$: PI($\boldsymbol{x}$) and CI($\boldsymbol{x}$).

If all assumptions of the standard linear model are fulfilled, we can construct a CI($\boldsymbol{x}_{\text{new}}$) for the fixed $\mathbb{E}(Y|X = \boldsymbol{x}_{\text{new}})$ (Fahrmeir et al., 2007):

$$\text{CI}(\boldsymbol{x}_{\text{new}}) = \left[\boldsymbol{x}_{\text{new}}^\top \hat{\boldsymbol{\beta}} \pm t_{1-\alpha/2}(n-p) \cdot \sqrt{\widehat{\mathbb{V}}(\boldsymbol{x}_{\text{new}}^\top \hat{\boldsymbol{\beta}})}\right]$$

For a new observation of the random variable Y the corresponding parametric PI($\boldsymbol{x}_{\text{new}}$) is:

$$\text{PI}(\boldsymbol{x}_{\text{new}}) = \left[\boldsymbol{x}_{\text{new}}^\top \hat{\boldsymbol{\beta}} \pm t_{1-\alpha/2}(n-p) \cdot \sqrt{\widehat{\mathbb{V}}(Y_{\text{new}} - \boldsymbol{x}_{\text{new}}^\top \hat{\boldsymbol{\beta}})}\right]$$

Those two intervals look quite similar, the main difference is the estimated variance. As in the unconditional case, the size of the PI is always bigger than the size of the CI:

$$2 \cdot t_{1-\alpha/2}(n-p) \cdot \sqrt{\widehat{\mathbb{V}}(Y_{\text{new}} - \boldsymbol{x}_{\text{new}}^\top \hat{\boldsymbol{\beta}})} > 2 \cdot t_{1-\alpha/2}(n-p) \cdot \sqrt{\widehat{\mathbb{V}}(\boldsymbol{x}_{\text{new}}^\top \hat{\boldsymbol{\beta}})}$$

For large sample sizes it holds true that

$$2 \cdot t_{1-\alpha/2}(n-p) \cdot \sqrt{\widehat{\mathbb{V}}(\boldsymbol{x}_{\text{new}}^\top \hat{\boldsymbol{\beta}})} \overset{n\to\infty}{\longrightarrow} 0,$$

while

$$2 \cdot t_{1-\alpha/2}(n-p) \cdot \sqrt{\widehat{\mathbb{V}}(Y_{\text{new}} - \boldsymbol{x}_{\text{new}}^\top \hat{\boldsymbol{\beta}})} \overset{n\to\infty}{\longrightarrow} 2 \cdot t_{1-\alpha/2}(n-p) \cdot \sigma^2.$$

That means that even for a perfectly specified model, with $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}$ and therefore a confidence level of 100% in estimating the conditional mean for a new observation $\boldsymbol{x}_{\text{new}}$, we cannot expect the size of the prediction interval to become arbitrarily small. As long as there exists variance of Y that is not explained by X, the prediction interval cannot be reduced to a size of 0.

The same has to be made clear for nonparametric intervals based on quantile regression: Even if the conditional quantiles are perfectly specified, the resulting intervals will have a certain size that mainly depends on the variance of $Y|X$. Of course it is true that for good predictor variables X we will get smaller intervals. It is also true that it will depend heavily on the new observation $\boldsymbol{x}_{\text{new}}$ how big the resulting PI($\boldsymbol{x}_{\text{new}}$) is. Hence, we do agree with Meinshausen who stated that a varying length of prediction intervals indicates that some observations can be predicted more accurately than others (Meinshausen, 2006).

But we should never expect for a response variable Y with high variance that the boundaries of the prediction intervals are anywhere near the expected mean of an observation. If σ is high, even the best method can never return a prediction interval that covers $(1-\alpha)\cdot 100\%$ of new observations for a given $\boldsymbol{x}_{\text{new}}$ and has a comparable size to the corresponding CI($\boldsymbol{x}_{\text{new}}$) for the conditional mean.

To sum up:

- The boundaries of prediction intervals and confidence intervals are random variables.
- Confidence intervals cover unknown but fixed values or parameters.
- Prediction intervals cover a new realization of a random variable.
- The size of a prediction interval therefore always is bigger and does not reduce to 0 - even for infinitely big sample sizes.

2.2.3 Simulated toy example

To further illustrate the difference between prediction intervals and confidence intervals, we simulated data for a toy example:

$$Y_i = 1 + 3 \cdot x_i + \log(1 + x_i) \cdot \varepsilon_i \quad \forall i = 1, ..., n$$

where x_i is an observed realization of $X \sim U(0, 10)$ and $\varepsilon \sim N(0, 2)$.

By $\log(1+x_i) \cdot \varepsilon_i$ we added moderate heteroscedasticity for the error term and therefore on purpose violated a key assumption of the standard linear regression model to point out the difference between parametric and nonparametric intervals.

We then constructed $\text{CI}_{0.95}(x)$ for μ and $\text{PI}_{0.95}(x)$ for Y_{new} (once based on the parametric model assumptions, and once based on quantile regression), for two different sample sizes ($n = 300$ and $n = 3000$) for all observed realizations of X. Results are presented in Figure 2.1.

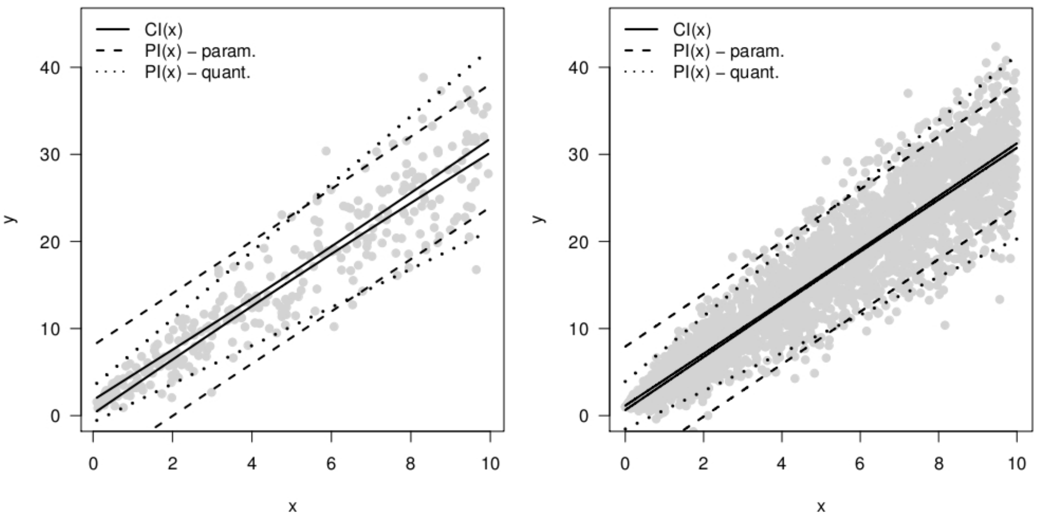

Figure 2.1: Example to compare 95% prediction intervals and confidence intervals for different sample sizes. Solid lines are confidence intervals for μ, the dashed line the corresponding parametric prediction interval for Y_{new}. The dotted line represents a nonparametric prediction interval based on conditional quantiles estimated by boosting. **Left:** $n = 300$, **Right:** $n = 3000$

The theoretically derived results can clearly be confirmed. The confidence interval is of much smaller size, which is further reduced with an increased n. The resulting prediction intervals do not change much when the sample size is increased. Additionally, we can notice the big advantage of the nonparametric approach based on quantiles, as it adopts to the heteroscedasticity of the data and gives a much smaller interval for smaller x.

2.3 Interpretation of prediction intervals for future observations

In the preceding section we illustrated the important differences between confidence intervals CI($\cdot$) and prediction intervals PI($\cdot$). For prediction intervals we distinguished between parametric intervals that use the assumed distribution function of $Y|X = \boldsymbol{x}$ and nonparametric intervals based on quantiles $Q_\tau(Y|X = \boldsymbol{x})$. In this section we will assume a setup with only one predictor variable, but of course the same holds true for intervals based on multiple predictors.

In the following sections we will go into more detail concerning the correct interpretation of a $\text{PI}_{1-\alpha}(x)$. We will assume that based on an observed training sample $\boldsymbol{y^*} = (y_1^*, ..., y_n^*)^\top$ and corresponding $\boldsymbol{x^*} = (x_1^*, ..., x_n^*)^\top$ some kind of parametric or nonparametric prediction interval was computed. Its boundaries are random variables as they depend on the sample. For a new observation x_{new}, $\text{PI}_{1-\alpha}(x_{\text{new}})$ should cover the corresponding Y_{new} of a random variable Y with a probability of $1 - \alpha$.

Yet, what does this mean from a frequentistic point of view? There exist at least two separate ways to interpret the coverage of a prediction interval for future observations:

- **Conditional interpretation:** For any x_{new} and a corresponding sample $\boldsymbol{y} = (y_1, ..., y_n)^\top$, about $(1 - \alpha) \cdot 100\%$ of the observations that all have the same observed value of the predictor variable x_{new} will fall inside the prediction interval PI(x_{new}). The coverage therefore refers to the observations belonging to this x_{new}.
- **Heuristic sample interpretation:** For any new sample $\boldsymbol{y} = (y_1, ..., y_n)^\top$ and corresponding prediction variables $\boldsymbol{x} = (x_1, ..., x_n)^\top$ about $(1 - \alpha) \cdot 100\%$ of the new sample $\boldsymbol{y}$ will fall inside the prediction intervals PI(x_1),...,PI(x_n). The coverage therefore refers to the whole sample.

This is not only a question of interpretation as it also implies different ways how the consistency of such intervals should be investigated by simulations.

In this section, we will justify why the conditional interpretation is the only one adequate when it comes to conditional prediction intervals. Yet we also acknowledge that for testing prediction intervals in real world problems, in most cases only the heuristic sample interpretation is feasible. In Chapter 6 we therefore analyze both interpretations in simulation studies.

2.3.1 The conditional interpretation

Following the conditional interpretation, the proposed coverage of the $\text{PI}_{1-\alpha}(x_{\text{new}})$ refers to every observation with this specific x_{new}. Hence, a new observation with exactly this x_{new} lies inside the prediction interval with a probability of $1-\alpha$:

$$P(Y \in \text{PI}(x_{\text{new}})|X = x_{\text{new}}) = 1-\alpha$$

Therefore, with $f(y|x_{\text{new}})$ being the conditional density of $Y|X$ at the point ($Y = y$ and $X = x_{\text{new}}$) and $I\{\cdot\}$ the indicator function we can conclude:

$$\begin{aligned} 1-\alpha &= \int\limits_{y \in \text{PI}(x_{\text{new}})} f(y|x_{\text{new}})dy \\ &= \int\limits_{Y} I\left\{y \in \text{PI}(x_{\text{new}})\right\} f(y|x_{\text{new}})dy \\ &= \mathbb{E}\left(Y \in \text{PI}(x_{\text{new}})|X = x_{\text{new}}\right) \quad \forall x_{\text{new}} \end{aligned}$$

For observed realizations of (Y, X) with $X = x_{\text{new}}$ we can estimate the conditional probability that Y falls into $\text{PI}(x_{\text{new}})$. Hence, to proof the consistency of the interval we would need many observations for one specific x_{new}.

For $(\boldsymbol{y}, x_{\text{new}})$ with $\boldsymbol{y} = (y_1, ..., y_n)^\top$:

$$\begin{aligned} \hat{P}(Y \in \text{PI}(x_{\text{new}})|X = x_{\text{new}}) &= \hat{\mathbb{E}}\left(Y \in \text{PI}(x_{\text{new}})|X = x_{\text{new}}\right) \\ &= \frac{\#\left\{\boldsymbol{y} \in \text{PI}(x_{\text{new}})\right\}}{n} \end{aligned}$$

Where $\#\{\cdot\}$ counts the cases where the condition inside the curly brackets is true.

2.3.2 The heuristic sample interpretation

If we want to try our intervals in a real world problem, in many cases it will be impossible to get for one specific x_{new} enough observations to show that the coverage of the interval is correct. But as the conditional interpretation holds true for every x_{new}, it seems obvious that we therefore can also directly interpret the prediction interval for a new sample with different x:

As

$$1-\alpha = \int\limits_{y \in \text{PI}(x)} f(y|x)dy \quad \forall x,$$

we can also integrate over x:

$$\begin{aligned}
1-\alpha &= \int\limits_X \int\limits_Y I\left\{y \in \mathrm{PI}(x)\right\} f(y|x) f(x) dy dx \\
&= \int\limits_X \int\limits_Y I\left\{y \in \mathrm{PI}(x)\right\} f(y,x) dy dx \\
&= \int\limits_Y \int\limits_X I\left\{y \in \mathrm{PI}(x)\right\} f(y,x) dx dy \\
&= \int\limits_Y I\left\{y \in \mathrm{PI}(x)\right\} f(y) dy \\
&= \mathbb{E}(Y \in \mathrm{PI}(x))
\end{aligned}$$

To proof the consistency, we can now use a new observed sample of $(\boldsymbol{y}, \boldsymbol{x})$ with $\boldsymbol{y} = (y_1, ..., y_n)^\top$ and $\boldsymbol{x} = (x_1, ..., x_n)^\top$.

$$\begin{aligned}
\hat{P}(Y \in \mathrm{PI}(x)) &= \hat{\mathbb{E}}(Y \in \mathrm{PI}(x)) \\
&= \frac{\sum_{i=1}^n I\left\{y_i \in \mathrm{PI}(x_i)\right\}}{n}
\end{aligned}$$

Meinshausen (2006) focused on the heuristic sample interpretation to show that the proposed prediction intervals based on quantile regression forest work in some typical benchmark data sets. In order to avoid using the same data points for estimating and testing the intervals, cross validation was used. Yet his intention was clearly to show that $(1-\alpha)\cdot 100\%$ of the data lies inside the intervals.

By plugging in

$$\int\limits_{y \in \mathrm{PI}(x)} f(y|x) dy = \mathbb{E}(Y \in \mathrm{PI}(x) | X = x) = 1-\alpha \quad \forall x$$

into the expectation of X

$$\begin{aligned}
\mathbb{E}\left(\mathbb{E}(Y \in \mathrm{PI}(x) | X = x)\right) &= \int\limits_X \int\limits_{y \in \mathrm{PI}(x)} f(y|x) f(x) dy dx \\
&= \mathbb{E}(Y \in \mathrm{PI}(x)) \\
&= 1-\alpha
\end{aligned}$$

we have shown that the conclusion

$$\begin{aligned}
\mathbb{E}\left(Y \in \mathrm{PI}(x_{\mathrm{new}}) | X = x_{\mathrm{new}}\right) &= 1-\alpha \quad \forall x_{\mathrm{new}} \\
\Rightarrow \quad \mathbb{E}(Y \in \mathrm{PI}(x)) &= 1-\alpha
\end{aligned}$$

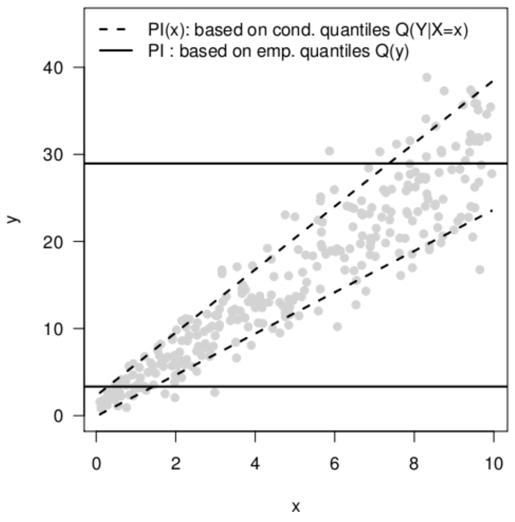

Figure 2.2: Example to compare the conditional interpretation from the heuristic sample interpretation. The dashed line represents a 80% conditional prediction interval PI(x) that gives for every x an adequate coverage. The solid line represents a 'naive' 80% prediction interval. Although it is no adequate interval for every x, it holds the 80% coverage over the whole sample.

in fact is correct. An interval that works for the conditional interpretation also should maintain the coverage for the whole sample. So, why do we argue that only the conditional interpretation is adequate?

The answer is that the backward conclusion is not possible. From $\mathbb{E}(Y \in \mathrm{PI}(x)) = 1 - \alpha$ we cannot conclude that $\mathbb{E}\left(Y \in \mathrm{PI}(x_{\text{new}})|X = x_{\text{new}}\right) = 1 - \alpha \quad \forall x_{\text{new}}$.

We are interested in prediction intervals that for a new observation x_{new} yield an interval which covers a corresponding new realization of $Y|X = x_{\text{new}}$.

$\mathbb{E}(Y \in \mathrm{PI}) = 1 - \alpha$ can be easily fulfilled by 'naive' intervals that have nothing to do with x. An example could be the empirical quantiles of the training sample $\boldsymbol{y}^*$ (Figure 5.2. illustrates this for the simulated example of Section 2.2.3.):

$$\mathrm{PI} = [\hat{Q}_{\alpha/2}(\boldsymbol{y}^*), \hat{Q}_{1-\alpha/2}(\boldsymbol{y}^*)]$$

This 'naive' interval will cover $(1 - \alpha) \cdot 100\%$ of the data points of a new sample. But it completely omits the variance of Y explained by X. Therefore it will never fulfill the stronger definition of $\mathbb{E}\left(Y \in \mathrm{PI}(x_{\text{new}})|X = x_{\text{new}}\right) = 1 - \alpha \quad \forall x_{\text{new}}$. Hence, this interval is the example why the backward conclusion is not correct. We have found an interval that holds the sample coverage, but does not fulfill the conditional intepretation for every x: In the middle of the distribution of x we will cover 100% of future observation while at the outer limits of x the coverage will be lower than $(1 - \alpha) \cdot 100\%$.

We therefore conclude that using the heuristic sample interpretation of course can give an indication if the coverage of the intervals is correct. In many practical situations it may be the only solution feasible. Correctly specified conditional prediction intervals should also give the right coverage over the whole sample. Yet, one should always have the correct interpretation in mind as also intervals which do not depend on x could cover much of the sample.

Chapter 3

Quantile Regression

We already presented the basic idea of *nonparametric* prediction intervals based on quantiles. They do not depend on any assumption about the distribution of the variable of interest, but do rely only on the information in the sample. For conditional prediction intervals for new observations we estimate the boundaries using conditional quantiles. These intervals therefore do not only depend on the sample, but also on the information of the predictor variables $\boldsymbol{x}_{\text{new}}$ of the new observation. The key to this approach is quantile regression.

One of the most important aspects of this thesis is to introduce and compare different ensemble methods to estimate conditional quantiles. But before we get to ensemble methods, it is necessary to have an idea of how quantile regression works, and how the parameters are estimated classically.

This chapter presents a short introduction to quantile regression. We first concentrate on theoretical aspects and the path from standard regression analysis to quantile regression before examples and basic estimation methods are presented. In the last section we will focus on some interesting properties of conditional quantiles.

3.1 From the conditional mean to conditional quantiles

What the regression curve does is give a grand summary for the averages of the distributions corresponding to the set of x's. We could go further and compute several different regression curves corresponding to the various percentage points of the distributions and thus get a more complete picture of the set. Ordinarily this is not done, and so regression often gives a rather incomplete picture. Just as the mean gives an incomplete picture of a single distribution, so the regression curve gives a corresponding incomplete picture for a set of distributions.

(Mosteller and Tukey (1977, p. 266), cited in Koenker and Hallock (2001))

In standard regression analysis, the focus lies on the conditional mean of a response variable Y given one or more predictors X. The influence of the predictor variables can be modeled as additive linear parametric effects $\hat{\mu}(\boldsymbol{x}) = \mathbb{E}(Y|X = \boldsymbol{x}) = \boldsymbol{x}^\top\boldsymbol{\beta}$, or also in an additive nonparametric form $\hat{\mu}(\boldsymbol{x}) = \sum_{j=1}^{p} f_j(\boldsymbol{x}_j)$.

3.1.1 The classical linear model

We already presented the classical linear model and its main assumptions in Section 2.1.1. We used it to model the conditional mean of the response variable, in order to get a prediction based on $\hat{\mathbb{E}}(Y_{\text{new}}|X = \boldsymbol{x}_{\text{new}})$.

$$Y_i = \boldsymbol{x}_i^\top\boldsymbol{\beta} + \varepsilon_i$$

Based on a observed sample $(y_1, \boldsymbol{x}_1), ..., (y_n, \boldsymbol{x}_n)$ of realizations of the random variables Y and X we can compute the parameter vector $\boldsymbol{\beta}$. In standard regression analysis estimation of the parameters is performed by minimizing the sum of squared residuals which yields in the ordinary least squares (OLS) estimator.

$$\begin{aligned}
\hat{\boldsymbol{\beta}}_{\text{OLS}} &= \underset{\beta}{\text{argmin}} \quad \text{OLS}(\boldsymbol{\beta}) \\
&= \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta})^2 \right\} \\
&= \underset{\beta}{\text{argmin}} \quad (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \\
&= \underset{\beta}{\text{argmin}} \quad \left\{ \boldsymbol{y}^\top\boldsymbol{y} - 2\boldsymbol{y}^\top\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\beta}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta} \right\}
\end{aligned}$$

The idea of this method goes back to Carl Friedrich Gauß (1777-1855) who used it in his pioneering work to predict the path of asteroid Ceres in 1801 (Forbes, 1996). The resulting $\hat{\boldsymbol{\beta}}_{\text{OLS}}$ that minimizes this sum returns the best model fit.

$$\frac{\partial \text{OLS}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -2\boldsymbol{X}^\top\boldsymbol{y} + 2\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\beta} \overset{!}{=} 0$$

As the matrix $\boldsymbol{X}$ has full rank, and $\boldsymbol{X}^\top\boldsymbol{X}$ therefore is invertible, one can conclude:

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}.$$

3.1.2 Quantiles

The best known quantile is the 0.5-quantile, the median. By definition, the median splits an ordered sample into two parts of equal size. Therefore 50% of the observations have higher values than the median, and 50% lower, respectively. The main advantage of the median over the arithmetic mean is its robustness towards very extreme values or outliers.

Let us assume that we have a company where 99 workers earn 2000 EUR a month. At the head of the company the chief executive officer (CEO) has a monthly salary of 22000 EUR. The arithmetic mean yields a mean income of 2200 EUR whereas the median is 2000 EUR which stronger reflects the situation of 99% of the company's employees.

Yet the median is just a special term for the 0.5-quantile. As the median splits an ordered sample into two parts of equal size, similarly the quartiles divide the sample into four parts with equal proportion of the observations in each segment. Quintiles divide the sample in five parts whereas deciles in ten parts.

The τ-quantile refers to the general case. In a population the proportion τ achieves smaller values than the τ-quantile and $(1-\tau)$ higher ones. By using different τ, quantiles are an descriptive approach to report more than just the mean of a variable, but also some aspects about its distribution. Plotting boxplots of different measurements includes more information than plotting means, as they can also show if the variance or the shape (symmetric or not) of the distribution changes.

The distribution function of a real valued random variable Y can be described by its quantiles $Q_\tau(Y)$, which for $0 < \tau < 1$ are defined as those y for which the distribution function $F_Y(y)$ yields τ:

$$P(Y \leq Q_\tau(Y)) = F_Y(Q_\tau(Y)) = \tau$$

If the distribution function in invertible, we can use the quantile function as the inverse of the distribution function. Then we can also define the τ-quantile more directly as a function of τ:

$$Q_\tau(Y) = F_Y^{-1}(\tau)$$

3.1.3 Conditional quantiles

As we have seen, standard regression focuses on the conditional mean of the response variable. The underlying question is the influence of predictor variables X on the expected mean of the response variable Y.

$$Y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + \varepsilon_i$$

For $\boldsymbol{\varepsilon} \sim N(0, \sigma^2 \boldsymbol{I})$ we conclude that

$$(Y|X = \boldsymbol{x}) \sim N(\boldsymbol{x}^\top \boldsymbol{\beta}, \sigma^2 \boldsymbol{I}).$$

For every data point $X = \boldsymbol{x}$ we therefore assume that the response follows a gaussian distribution. In fact by modeling the expected mean we always model a whole conditional distribution function, only in expecting that $\boldsymbol{\varepsilon} \sim N(0, \sigma^2 \boldsymbol{I})$. We expect that in every point $X = \boldsymbol{x}$ the response follows the same distribution, with the same variance only shifted by the expected mean in this point.

By classical quantile regression we want to use a much more flexible approach that also will be more accurate as it in fact does not estimate the distribution parameter for a normal distribution, but estimates directly the conditional quantiles without assuming a distribution.

The conditional distribution function $F_{Y|X}(y|\boldsymbol{x})$ is defined by the probability that Y is smaller than $y \in \mathbb{R}$ conditional on $X = \boldsymbol{x}$.

$$F_{Y|X}(y|\boldsymbol{x}) = P(Y \leq y|X = \boldsymbol{x})$$

The basic idea of quantile regression is, in contrast to standard regression, not to model the expected mean but the $\tau-$quantile $Q_\tau(Y|X = \boldsymbol{x})$ of the conditional distribution function.

Therefore, for every τ with $0 < \tau < 1$ the τ-quantile $Q_\tau(Y|X = \boldsymbol{x})$ is defined as:

$$Q_\tau(Y|X = \boldsymbol{x}) = F^{-1}_{Y|X}(\tau) = \inf\{y : F_{Y|X}(y|\boldsymbol{x}) \geq \tau\}$$

Assume that we have a real valued response variable Y and the one-dimensional predictor X with a linear influence on Y following the standard gaussian model:

$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad \varepsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$$

If this model is assumed to be correct, it can be easily linked to the conditional distribution function.

$$\begin{aligned}
F_{Y|X}(y|x_i) &= P(Y_i \leq y|X = x_i) \\
&= P\left(\frac{Y_i - \beta_0 - \beta_1 x_i}{\sigma} \leq \frac{y - \beta_0 - \beta_1 x_i}{\sigma}\right) \\
&= \Phi\left(\frac{y - \beta_0 - \beta_1 x_i}{\sigma}\right) \\
&= \tau \\
\Longleftrightarrow \frac{Q_\tau(Y|X = x_i) - \beta_0 - \beta_1 x_i}{\sigma} &= \Phi^{-1}(\tau)
\end{aligned}$$

Now we can model the τ-quantile of the response variable as a linear function of x.

$$\begin{aligned} q_\tau(x) = Q_\tau(Y|X=x) &= \beta_0 + \sigma \cdot \Phi^{-1}(\tau) + \beta_1 x \\ &= \beta_{0\tau} + \beta_1 x \end{aligned}$$

where

$$\beta_{0\tau} = \beta_0 + \sigma \cdot \Phi^{-1}(\tau).$$

We can conclude that the parameter β_1 in this case remains the same, whether we use standard (conditional mean) or quantile regression (conditional quantiles). Therefore, β_1 does not depend on τ. For the median ($\tau = 0.5$) not even the intercept changes as $\Phi^{-1}(0.5) = 0$:

$$q_{0.5}(x) = Q_{0.5}(Y|X=x) = \beta_0 + \beta_1 x$$

Of course, this all holds true due to the assumption of symmetric *i.i.d.* gaussian ε. In a heteroscedastic setup, the results for β_1 would heavily depend on τ. We will demonstrate this in a simple simulated example in Section 3.3.

3.2 Basic estimation

As we have already seen, in standard regression analysis parameters are calculated by minimizing the sum of squared residuals, which leads to following equation:

$$\hat{\boldsymbol{\beta}}_{\text{OLS}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

From a decision theoretic point of view, this estimator minimizes the quadratic loss.

It is obvious that for estimating the τ quantiles a symmetric loss function is only appropriate for the median. For every $\tau \neq 0.5$, as we do not want to have on both sides of the regression curve the same proportion of data points, we have to penalize the distance from the expected quantile unsymmetrically. The weight of each residual therefore has to depend on τ.

3.2.1 Minimizing the *check function*

In practice the estimation of the quantiles from a sample $y_1, ..., y_n$ is calculated based on the ordered list of variables $y_{[1]}, ..., y_{[n]}$.

The median can now be estimated by $y_{[(n+1)/2]}$ given n is odd or $\frac{y_{[n/2]}+y_{[n/2+1]}}{2}$ for even n.

Regarding this process of ordering and sorting the sample, it therefore is at least surprising that quantiles can be linked to an optimization problem. But in the same way as the mean is the solution to the problem of minimizing a sum of squared residuals of an observed sample,

$$\min\left\{\sum_{i=1}^{n}(y_i - \hat{Y})^2\right\} \quad \Rightarrow \hat{Y} = \bar{y}$$

the median is the solution to the problem of minimizing a sum of absolute residuals:

$$\min\left\{\sum_{i=1}^{n}|y_i - \hat{Y}|\right\} \quad \Rightarrow \hat{Y} = \hat{Q}_{0.5}(\boldsymbol{y})$$

For the other quantiles, we now have to weight the absolute residuals asymmetrically (depending on τ). It can be shown that the coefficients for a linear quantile regression can be estimated by minimizing the *check function* (Koenker, 2005):

$$\underset{\beta_\tau}{\operatorname{argmin}} \sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}_\tau) \quad \text{where} \quad \rho_\tau(u) = \begin{cases} u \cdot \tau & u \geq 0 \\ u \cdot (\tau - 1) & u < 0. \end{cases}$$

As $\boldsymbol{x}^\top \boldsymbol{\beta}_\tau = \hat{Y}$ the expected loss is therefore

$$\mathbb{E}[\rho_\tau(y - \hat{Y})] = (\tau - 1) \cdot \int_{-\infty}^{\hat{Y}} (y - \hat{Y}) f(y) dy + \tau \cdot \int_{\hat{Y}}^{\infty} (y - \hat{Y}) f(y) dy.$$

Minimizing the expected loss with respect to $\hat{Y}$ (compare to Fenske (2008))

$$\begin{aligned} \frac{\partial}{\partial \hat{Y}} \mathbb{E}_{F_Y}[\rho_\tau(y - \hat{Y})] &= -(\tau - 1) \cdot \int_{-\infty}^{\hat{Y}} f(y) dy - \tau \cdot \int_{\hat{Y}}^{\infty} f(y) dy \\ &= -\tau + \int_{-\infty}^{\hat{Y}} f(y) dy \\ &= F_Y(\hat{Y}) - \tau \quad \overset{!}{=} \quad 0 \end{aligned}$$

reveals the τ-quantile:

$$\hat{Y} = Q_\tau(Y) = F_Y^{-1}(\tau)$$

Quantile regression parameters can therefore be estimated similarly to the standard regression parameters by minimizing the expected loss. The difference in the two loss functions and the dependency of the *check function* on τ is demonstrated in Figure 3.1.

In the standard regression model, the quadratic loss is symmetric and penalizes especially high distances to the regression line or $\hat{Y}$. The loss function for quantile regression for the median ($\tau = 0.5$) is also symmetric but less sensitive to extreme observations as outliers as it refers to the absolute residuals. For small τ, the check

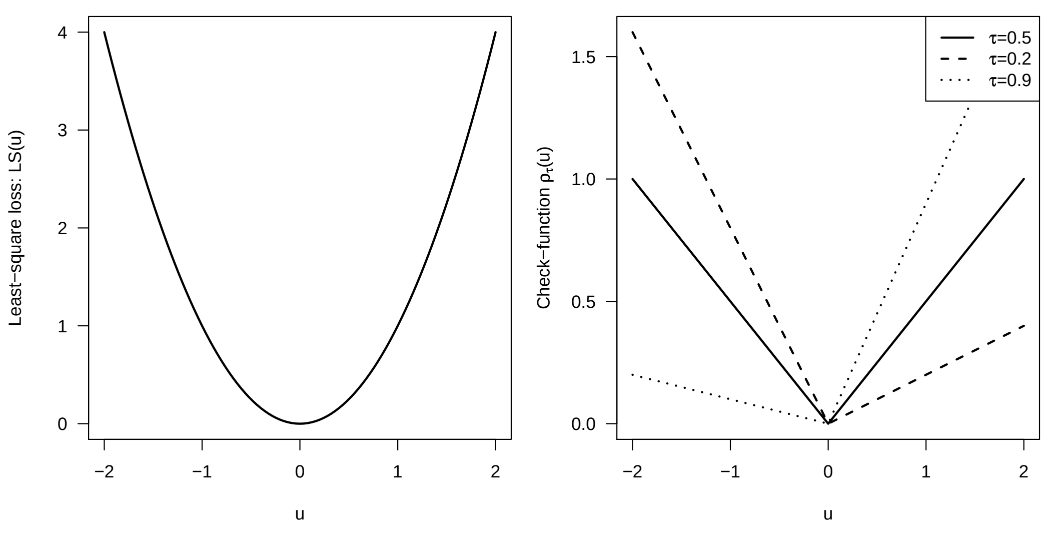

Figure 3.1: Loss function of the standard regression model (left) and the check function for quantile regression (right) for different values of τ.

function gives little weight for observations where $(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}_\tau) \geq 0$ or $u \geq 0$ which means the expected quantile is smaller than the observed response. On the other hand, if $u < 0$ and therefore the expected quantile is bigger than the observed outcome, the loss function yields much higher values. Thus, by minimizing the loss, $\tau \cdot 100\%$ of the observations will be smaller than $\hat{Y}$ and $(1 - \tau) \cdot 100\%$ bigger.

Yet as the *check-function* is not differentiable in 0, the solution to this minimization problem can not be written down by a single formula, as in the ordinary least squares (OLS) case. The standard solution is the usage of mathematical optimization techniques provided by linear programming. An implementation using this techniques is available through the function `rq()` in the `quantreg` package (Koenker, 2009b,a).

In this thesis we will focus on the estimation of conditional quantiles by ensemble methods, rather than linear programming. Fenske et al. (2009) showed that minimization of the *check-function* by boosting measures is more than competitive and implies some favorable properties. We will focus on this in detail in Chapter 5. A completely different way to estimate conditional quantiles will be presented in Chapter 4, where the approach of Meinshausen (2006) to combine random forests with quantile regression is laid out.

But before we get to ensemble methods, in order to give a more complete picture, we will shortly touch two rather unusual approaches based on quasi-likelihood and bayesian inference respectively.

3.2.2 Quasi-Likelihood approach

As an alternative estimation method we present another approach based on maximum likelihood inference. As in classical quantile regression we do not have assumptions about the distribution of the error term, in this case one also uses the term quasi-maximum likelihood inference. Geraci and Bottai (2007) employ a likelihood function that is based on the asymmetric Laplace distribution (ALD)

$$Y \sim ALD(\mu, \sigma, \tau)$$

where $-\infty < y < \infty$, $\mu \in \mathbb{R}$ and $\tau \in (0,1)$. The density is

$$f(y) = \frac{\tau(1-\tau)}{\sigma} \cdot \exp\left\{-\rho_\tau\left(\frac{y-\mu}{\sigma}\right)\right\},$$

where $\rho_\tau(\cdot)$ denotes the *check-function.* The shape of this density for different τ is demonstrated in Figure 3.2.

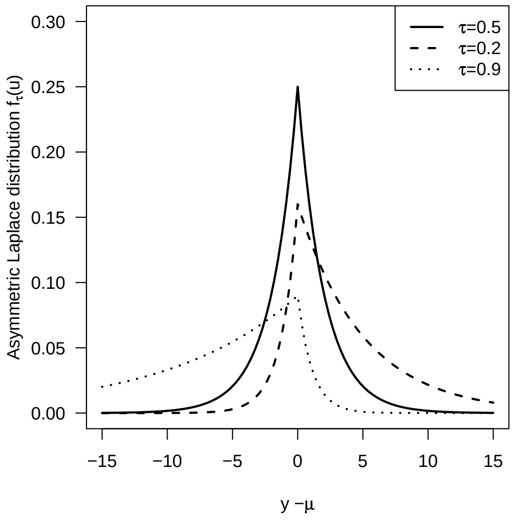

Figure 3.2: The density of the asymmetric Laplace distribution for different τ.

If we (contradictory to the classical approach for quantile regression) assume the ALD as the distribution of the error terms ε_τ, this yields the following Quasi-Likelihood to maximize

$$\frac{1}{\sigma}\exp\left\{-\sum_{i=1}^{n}\rho_\tau\left(\frac{y_i - \boldsymbol{x}_{\boldsymbol{i}}^\top\boldsymbol{\beta}_\tau}{\sigma}\right)\right\} \longrightarrow \max$$

which equals the minimization of the *check-function.*

$$\Longleftrightarrow \sum_{i=1}^{n}\rho_\tau(y_i - \boldsymbol{x}_{\boldsymbol{i}}^\top\boldsymbol{\beta}_\tau) \longrightarrow \min$$

3.2.3 Bayesian approach

Bayesian inference in generalized linear or also additive models has become more and more feasible through Markov chain Monte Carlo (MCMC) methods which can be used to obtain samples of the posterior distribution even in very complex situations. The obvious advantage of the bayesian framework is the benefit of having the entire posterior distribution of the parameter of interest. This distribution includes information about the estimation uncertainty which can be taken into account when making predictions.

For quantile regression, Yue and Moyeed (2001) proposed to use, in absence of any realistic information, an improper uniform distribution $p(\boldsymbol{\beta}_\tau) \propto 1$ as a prior for $\boldsymbol{\beta}_\tau$. They show that the resulting joint posterior distribution is proper. The posterior distributions of the parameters are obtained using MCMC methods. Credible intervals around the parameters are computed using the MCMC samples of $\boldsymbol{\beta}_\tau(p)$.

The posteriori distribution of $\boldsymbol{\beta}_\tau$, $\pi(\boldsymbol{\beta}_\tau|\boldsymbol{y})$ for given observations $\boldsymbol{y} = (y_1, ..., y_n)$ is given by

$$\pi(\boldsymbol{\beta}_\tau|\boldsymbol{y}) \propto L(\boldsymbol{y}|\boldsymbol{\beta}_\tau)p(\boldsymbol{\beta}_\tau),$$

where $p(\boldsymbol{\beta}_\tau)$ denotes the prior distribution of $\boldsymbol{\beta}_\tau$ and $L(\boldsymbol{y}|\boldsymbol{\beta}_\tau)$ the quasi-likelihood function, based on the ALD with $\sigma = 1$ which must be seen as a rather strong assumption:

$$L(\boldsymbol{y}|\boldsymbol{\beta}_\tau) = \tau^n(1-\tau)^n \exp\left\{-\sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}_\tau)\right\}$$

If one takes the maximum of the posterior distribution in this setup as a point estimator for $\boldsymbol{\beta}_\tau$, it maximizes the ALD-Likelihood if the improper prior $p(\boldsymbol{\beta}_\tau) \propto 1$ was used.

$$\begin{aligned} \pi(\boldsymbol{\beta}_\tau|\boldsymbol{y}) &\propto L(\boldsymbol{y}|\boldsymbol{\beta}_\tau)p(\boldsymbol{\beta}_\tau) \\ &\propto L(\boldsymbol{y}|\boldsymbol{\beta}_\tau) \\ &\propto \exp\left\{-\sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}_\tau)\right\} \longrightarrow \max \end{aligned}$$

This yields in the same value as minimizing the *check-function*:

$$\Longleftrightarrow \sum_{i=1}^{n} \rho_\tau(y_i - \boldsymbol{x}_{\boldsymbol{i}}^\top \boldsymbol{\beta}_\tau) \longrightarrow \min$$

This approach was analyzed in a diploma-thesis by Cieczynski (2009).

To sum up, the Quasi-Likelihood approach in this case equals the point estimator in the bayesian framework and also results in the same parameters as the decision theory based way of minimizing the check function as the expected loss. All three different basic approaches may have their advantages, often linked to the framework they are based on. But in the end, when it comes to a point estimator for $\boldsymbol{\beta}_\tau$ they do all result in minimizing the *check-function*.

3.3 Simulated toy examples

To show some of the basic differences but also similarities of quantile regression and standard linear models, we will present some toy examples (compare to Fenske et al. (2009)). Those examples will illustrate some properties of quantile regression which will be summarized in the following section.

We simulated random variables $x_i \overset{\text{iid}}{\sim} U(0,1)$ and $\varepsilon_i \overset{\text{iid}}{\sim} N(0,1)$ for $i = 1, ..., 1000$ for two different linear setups:

- Homoscedastic setup: $Y_i = 2 + 3x_i + \varepsilon_i$
- Heteroscedastic setup: $Y_i = 2 + 3x_i + x_i\varepsilon_i$

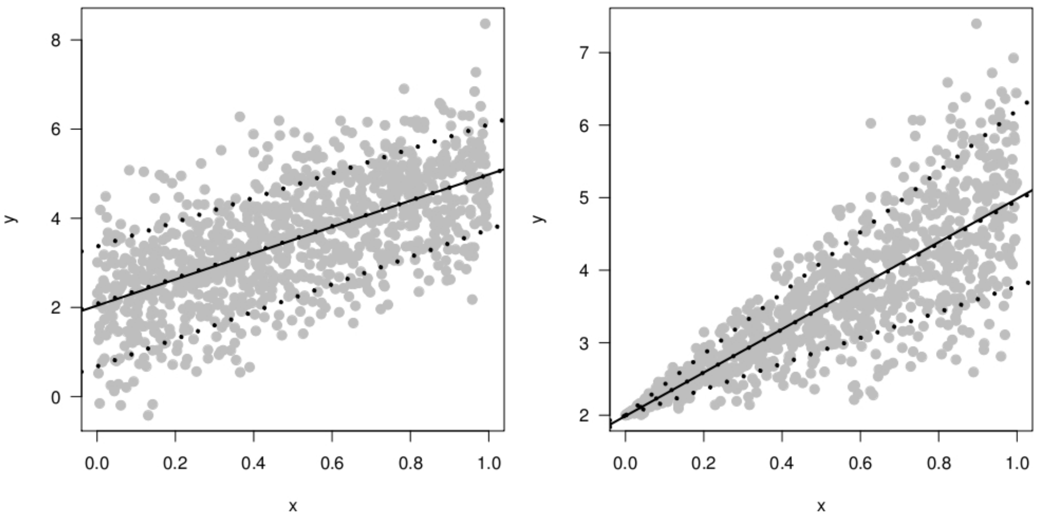

Figure 3.3: Example for quantile regression in simulated data ($\varepsilon_i \sim N(0,1)$). Homoscedastic setup on the left and heteroscedastic setup on the right side. The solid lines represent the results of a standard regression model, the dotted lines represent the results of quantile regression with $\boldsymbol{\tau} = (0.1, 0.5, 0.9)^\top$.

For both setups, both a standard regression and three quantile regression models for $\boldsymbol{\tau} = (0.1, 0.5, 0.9)^\top$ were fitted. The results are illustrated in Figure 3.3. In the homoscedastic setup, both quantile lines are parallel shifted to the standard regression line. As the ε_i are *i.i.d.*, β_1 (the slope of the regression line) remains the same.

In the heteroscedastic setup the slopes of the regression lines differ clearly. It can even be noticed that for a small section of the x-grid, quantile crossing appears. This means that the regression line of a lower quantile crosses the line of a higher quantile and therefore its model yields higher values for a given $\boldsymbol{x}$:

$$\text{Quantile-crossing:} \quad \boldsymbol{x}^\top \hat{\boldsymbol{\beta}}_\tau < \boldsymbol{x}^\top \hat{\boldsymbol{\beta}}_{\tau'} \quad \text{for} \quad \tau > \tau'$$

As the distribution of the residuals remains symmetric ($\varepsilon \sim N(0,1)$), the standard regression toward the conditional mean and the quantile regression with $\tau = 0.5$ focusing on the median result in nearly the same parameter estimations and, therefore, the same regression line.

In the next example, the same setup as in the first example is used. Only the distribution of the error terms is changed into a non symmetric Gamma distribution G(2,2). The results are illustrated in Figure 3.4.

For the homoscedastic setup, the quantile regression lines once again are parallel shifted to the standard regression line. Therefore, β_1 remains the same. But this time, due to the unsymmetric error term distribution ($\varepsilon_i \sim G(2,2)$), the intercept of the standard regression and the $\tau = 0.5$ (median) quantile differ for both setups.

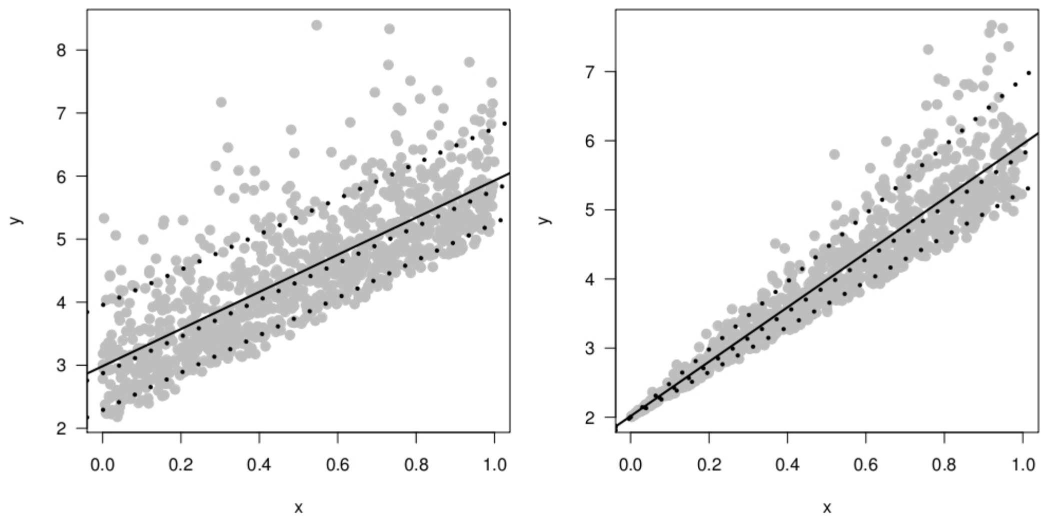

Figure 3.4: Example for quantile regression in simulated data ($\varepsilon_i \sim G(2,2)$). Homoscedastic setup on the left and heteroscedastic setup on the right side. The solid lines represent the results of a standard regression model, dotted lines represent the results of quantile regression with $\boldsymbol{\tau} = (0.1, 0.5, 0.9)^\top$.

3.4 Conclusions on some properties of quantile regression

These examples underline some basic aspects about the relation between standard regression and quantile regression:

- For symmetric error terms, modeling the median or the mean yields the same intercept.
- If the setup is even homoscedastic, also the slope of the regression line is the same for all quantiles.
- If the distribution of the error term is assumed correctly, standard regression in fact does also estimate the quantiles, based on this assumption and less flexible than the classical *nonparametric* approach (Fenske, 2008).

Using quantile regression is also a way to detect misspecification and violations of assumptions of the standard linear model. This is not surprising as we already pointed out that by quantile regression more information can be gained about the conditional distribution function as with standard regression. If one goes on modeling a grid of quantiles, it is obviously possible to describe more facets of the underlying distribution function and, therefore, structural aspects as symmetry and homoscedasticity can be identified.

In comparison to standard regression quantile regression is based only on low-level assumptions.

$$Y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta}_\tau + \varepsilon_{\tau i}$$

- The $\varepsilon_{\tau i}$ must be independent, but do not have to be homoscedastic.
- There is no distribution assumption for $\varepsilon_{\tau i}$, they only have to fulfill $F^{-1}_{\varepsilon_{\tau i}}(\tau) = 0$.

As a conclusion, this means for nonparametric prediction intervals which borders are conditional quantiles that they are in much more situations applyable than standard methods. The main advantage is the absence of an assumed distribution of $\varepsilon_{\tau i}$. Furthermore, they also do not depend on homoscedastic error terms. In fact, as can be seen in the toy examples, the conditional quantiles adopt very precisely to the heteroscedastic setup, and yield different slopes for different τ. The resulting intervals therefore will get broader for big x_{new}, if heteroscedasticity was detected.

Quantile regression is more robust to outliers or high-leverage points. As the median is a more robust measure than the arithmetic mean, this property is also passed on to the other quantiles.

Before estimating the coefficients $\boldsymbol{\beta}_\tau$ by minimizing the *check-function*, τ has to be fixed. The estimation process for different τ is carried out separately even though they are correlated. This independent estimation is the reason why quantile crossing can occur.

This is one of the most important handicaps of classical methods to estimate quantile-regression by minimizing the *check-function*. Yet as we are interested in conditional quantiles to use as borders for prediction intervals, in our case we will model only two quantiles $\hat{q}_\tau(x_{\text{new}})$ and $\hat{q}_{\tau'}(x_{\text{new}})$ with $|\tau' - \tau| = 1 - \alpha$. For a small α the quantiles should therefore differ enough to prevent quantile crossing.

The standard form to minimize the *check-function* by linear programming is implemented in the function `rq()` in the pache `quantreg` (Koenker, 2009b,a).

In the following chapter, a *nonparametric* approach based on ensembles of classification and regression trees is presented which in fact holds a solution to the problem of separate estimation.

Chapter 4

Quantile Regression Forests

In Chapter 3 we introduced quantile regression as a powerful and flexible tool to investigate more aspects about the conditional distribution than with standard regression, focusing on the conditional mean. Some basic concepts for estimating the parameters $\boldsymbol{\beta}_\tau$ were presented, in the end always aiming to minimize the *check-function.*

In this chapter we want to focus on quantile regression forests (Meinshausen, 2006) as a nonparametric approach for estimating conditional quantiles. This further development of random forests is designed to estimate conditional quantiles without using the *check-function*. This approach is situated in the field of ensemble-methods where roughly speaking a better predictive performance is achieved by combining different models. Therefore subsamples of the given data sets are analyzed, often by using bootstrap methods. We will focus on this later in detail.

To understand the concept of quantile regression forests, it is essential to understand in principle the idea of classification and regression trees (CART) which are the basis for random forests. We will especially focus on the aspects of trees and forests that are essential for the usage of quantile regression forests.

This chapter will be structured as follows: In Section 4.1 we will focus on the basic idea of CARTs also by presenting some standard examples. In Section 4.2 we will present random forests as a ensemble-method tool with high predictive validity that is based on CARTs. In Section 4.3 we will finally introduce quantile regression forests, which will later be extensively used as one of the main algorithms to estimate prediction intervals by ensemble methods.

4.1 Classification and regression trees

Classification and regression trees (CART) refer to a nonparametric statistical tool based on recursive partitioning.

The term *nonparametric* is often used as a general description for methods that do not rely on assumptions about the probability distribution, where the data was drawn from. We already used this term in the context of quantile regression where we did not assume any distribution for the model's error term. Hence also prediction intervals based on quantile regressions are *nonparametric* as no assumptions about underlying distributions are needed.

In fact, CART fits in the scope of this definition. Trees can be used for all kinds of data without assuming any distribution. But in this case the term *nonparametric* in its literally meaning can also point to an important difference to other approaches: We do not estimate $\boldsymbol{\beta}$ as in classical regression or $\boldsymbol{\beta}_\tau$ for quantile regression. We give predictions for regression and classification problems without using a parametric model of the form $\boldsymbol{x}^\top\boldsymbol{\beta}$.

4.1.1 Basic idea

We are still in the regression framework as we try to explain and predict the influence of one or more predictor variables X on the response Y. If Y is continuous, we use CART to grow trees that are called regression trees. If Y is binary or categorical, we call them classification trees. Trees normally only make sense for multidimensional X, and favor setups with interactions. In the following X is a set of predictor variables $X_1, ..., X_p$.

The idea of growing trees goes back to the Automatic Interaction Detection (AID) program developed at the Institute for Social Research in Michigan by Morgan and Sonquist (1963). In the statistical community it later attracted attention as a tool for nonparametric classification (Friedman, 1977) before it finally got its name as *Classification and Regression Trees* by Breiman et al. (1984). The basic structure of a tree and the terms used to describe its components is presented in Figure 4.1.

There exist a whole bunch of different approaches of how to construct a classification or regression tree. But most of them can be linked to a general rule (compare to Everitt and Hothorn (2006)):

1. Partition the observations by binary splits in a recursive way.

2. Predict by a constant term in each cell.

The first part can be divided into two separate steps. For a response Y and a given set of predictor variables $X_1, ..., X_p$, one variable X_j $(1 \leq j \leq p)$ has to be selected which splits the response into two optimal groups. If this variable X_j is categorical, the set of categories A has to be chosen that performs the split: $X_j \in A$ or $X_j \notin A$. If X_j is continuous, a cut point ξ has to be selected that also performs a binary split

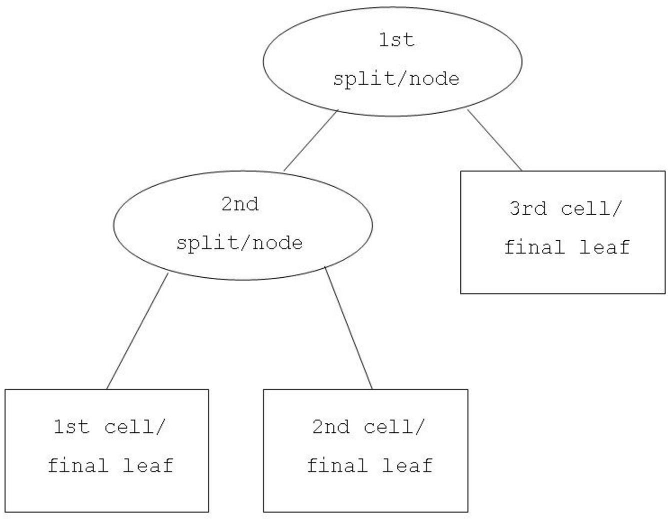

Figure 4.1: Basic structure of a classification or regression tree.

by $X_j > \xi$ or $X_j \leq \xi$.

Difficulties in understanding this first point still often do not arise by the selection of the variable or the cut point. Those questions are familiar to data analysts. But the recursive way these steps are carried out often sounds odd at first sight. Though the idea can be easily explained by an example:

Let us assume a doctor arrives as a first responder to a car accident with several victims. As he cannot help everyone at the same time, the doctor will intuitively decide whom to help first by a few questions which can be answered by yes or no: Which victims are still alive? Have the victims alive stable pulse and breathing? Are the victims in a life threatening situation? He therefore classifies the victims with every question into two separate groups before asking the next one. We can see this classification procedure as a simple classification tree with binary splits in each node. In the final cell there will be the decision if the victim falls in the group which should be attended immediately. The doctor has learned those questions by his daily routine and experience as a professional.

When statisticians want to build a tree, they first need training data. In this training data a tree is grown that best classifies the given observations in its known categories. For a new observation one will now go down the whole tree node by node until the final cell is reached, and classify the new observation with the category the majority of the observations in the training data has in just this cell.

If Y is continuous, the prediction is based on the mean or median value of the observations in the final cell where the new observation has dropped.

As we have already mentioned different algorithm exist to construct regression and classification trees. Those algorithm mainly differ in three points (compare to Everitt and Hothorn (2006)):

- How is the predictor variable X_j to split on selected from $X_1, ..., X_p$?
- How is the cut point ξ or the splitting set of categories A chosen in X_j?
- Which stopping criteria is applied?

The last point is essential as too early stopping would mean to omit important covariates and splits, while no early stopping would lead directly into overfitting by constructing too many nodes with too small cells (or leaves) as endpoints.

4.1.2 Compute trees by `rpart`

One of the best known algorithms for growing trees is based on the description in the book 'Classification and Regression Trees' by Breiman et al. (1984). It is implemented in the R package `rpart` by Therneau et al. (2009).

The selection of the predicting variables and the cut point for the splits are chosen in an integrated form: The algorithm first checks all possible splits in all given variables $X_1, ..., X_p$ before selecting the split that leads to the most homogeneous groups with respect to the response Y. The homogeneity or 'purity' is measured by default with the Gini criterion, but alternatives can be chosen.

The problem of stopping the recursion is solved in retrospect by the so called *pruning*-method. First, a very large tree is grown and afterward one begins to cut back some branches until the tree has reached its optimal size. The decision which was the optimal size for the given tree can be based on cross validation.

As an example, we use the well known iris data which gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica (Fisher, 1936). Figure 4.2 presents the plot of an classification tree computed by `rpart` with default settings.

The tree consists of two nodes and therefore three final cells or leafs. Every iris with `Petal.Length` > 2.45 will be classified as the species setosa. For the rest of the flowers the variable `Petal.Length` is deciding with a splitpoint of 1.75 if the iris belongs to the species versicolor or virginica.

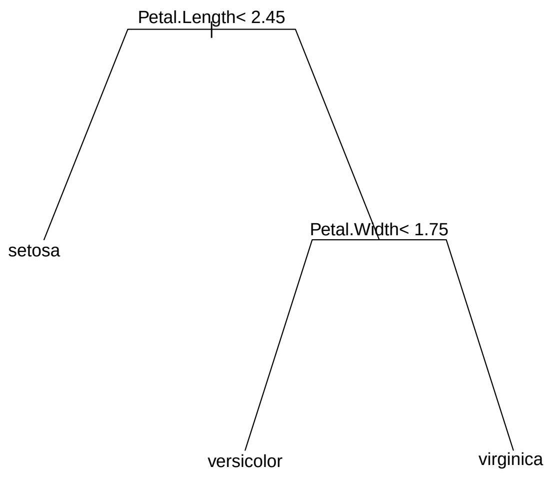

Figure 4.2: Example for a simple classification tree for the iris data computed by `rpart` with default settings. The prediction for a species can be seen in the final leaves.

4.1.3 Compute trees by `party`

Another approach for growing trees is implemented in the package `party` (Hothorn et al., 2009, 2006). This algorithm combines the recursive partitioning idea with a concept of conditional inference.

The aim of the algorithm is to deal with two mayor problems of the classical approaches:

- Overfitting
- Variable selection biased towards covariates with many possible splits

Overfitting can be handled through pruning in the `rpart` package. But the selection bias still stays a problem and can be easily explained:

The classical algorithm checks every possible split in all given predictor variables $X_1, ..., X_p$ before selecting the split that leads to the most homogeneous groups. This implies that a binary variable that has only one possible split is for example less probably selected compared to a variable with two or more possible splits.

The proposed solution is to include permutation tests in the search for the variable to split on. By using resampling methods it is possible to construct general independence tests that work for different scale levels without assuming any distribution. They are based on the conditional distribution of test statistics measuring the association of the given covariates $X_1, ..., X_p$ with the response Y. The theory for this approach of conditional inference is based on the work of Strasser and Weber (1999).

In every step, multiple testing is applied to decide if another split is justified. Therefore in the given subset of observations, independence tests are applied, testing to the level of α if there are any significant effects for any predictor variable with the response. If the test reveals significant effects, the variable X_j with the strongest effect is chosen for the next split. If the H_0 of independence is not rejected, the partitioning procedure is stopped. Therefore all variables are treated equally, no matter what scale level or how many categories they have.

Afterward also the selection of the split point ξ for X_j is based on the conditional distribution generated by resampling methods. As an example, we again use the iris data (Fisher, 1936) for classification. The resulting tree is presented in Figure 4.3.

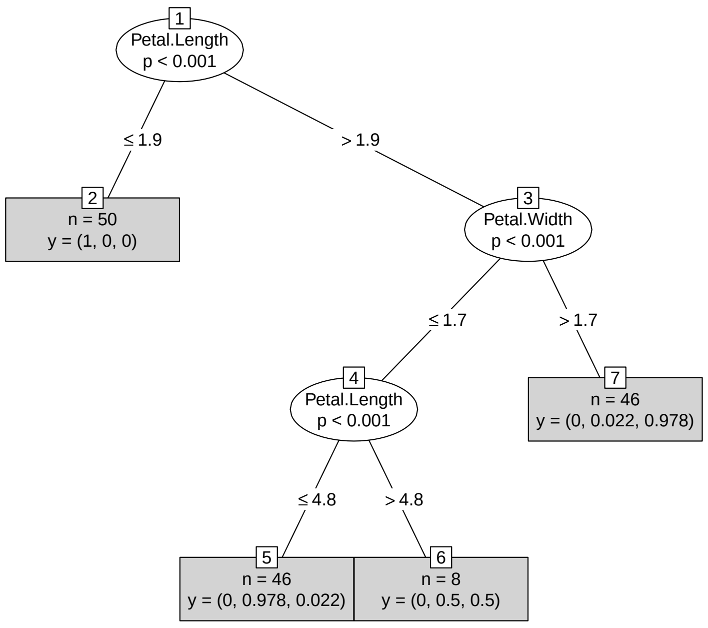

Figure 4.3: Example for a simple classification tree for the iris data computed by `party` with default settings. The proportion for each species in the final leaves is noted in brackets.

If we compare the resulting trees of both algorithms (Figure 4.2 and Figure 4.3) we can note that both algorithms chose the the same variables for the first two splits, although the cut points differ. The tree computed by `party` also includes a third split using the variable `Petal.Length` that was already chosen for the first split.

Additionally to the last split and cut points there is another obvious difference in the two resulting figures: In Figure 4.3 there is also information about the p-values of each split included. In the classical algorithm significance in the statistical sense was never taken into account. But as the stopping criteria now are linked to general independence tests the underlying p-values can also be reported to the user.

Hence, the parameter α that is used for the early stopping criteria in the form of a tuning parameter (a high α leads to large trees with many splits while a small α will lead to relatively small trees, as the H_0 of independence is more often rejected), can also be interpreted as a global significance level. For a given $\alpha = 0.05$ no variables will be included in a tree that do not at least show an effect with a p-value < 0.05 for the actual subset of variables.

Although this algorithm also includes statistical tests and significance levels, the resulting trees still fit in the definition of a *nonparametric* statistical method. For the selection of the variables, permutation tests are used. The resulting p-values do not depend on any asymptotic probability distribution of a test statistic but on its conditional distribution computed by resampling methods.

4.2 Random forests

As we now got to know the idea of classification and regression trees, we can go on to what we are really interested in: Ensemble methods that combine different trees to random forests and the resulting estimation of the conditional distribution function. Therefore, we shortly explain what is meant by the term ensemble methods before we go on over bagging to random forests.

4.2.1 Ensemble methods

Ensemble methods were first introduced in the machine-learning community for so called supervised-learning problems (‘classification‘ in statistical language). A good summary is given in Dietterich (2000). Ensemble methods follow the idea that even with simple prediction techniques a powerful tool for prediction can be computed, if they are aggregated in a useful manner. This may sound odd in the first place, but can be explained in a short example:

Let us assume that a prisoner in his cell has no window and therefore no information about the weather outside. One indication of the weather outside could be the boots of the prison guard when he enters the cell in the morning for control. If they seem wet or the guard even leaves a wet trail, this can be a clear indication of a rainy day outside. Of course it is possible that he just crossed a recently cleaned floor, although this seems unlikely. But on the other hand, if the boots are dry, one can not that easily conclude that there must be a sunny day outside. It is also possible that the officer arrived by car, or is already on duty for hours and the boots are dry again. All in all, the boot-procedure does not seem to give a very accurate prediction about the unknown status of the weather outside. But if we could combine many such perhaps weak prediction techniques, the performance could rise. Let us imagine the prisoner meets 100 other prisoners for lunch. By sharing the information about their guard's boots, they could develop a much more accurate and better performing prediction.

When we now go back from this constructed example to real world statistical problems, we can use this approach to combine different prediction methods (base learners) into one more accurate predictive tool. Therefore, we use one base learner many times for a slightly different data set. For a new observation, we use an aggregated predictor of all base learners by averaging the results.

The term ensemble-methods reflects this general idea, in practice approaches differ in three points:

- Base learners
- Ways to aggregate (Majority voting vs. weighted voting)
- Ways to generate subsamples or re-weighted data in each iteration

4.2.2 Bagging

Bagging ($\equiv$ bootstrap aggregation) was introduced by Breiman (1996) as a technique for generating multiple versions of a base learner and combine them to get an aggregated predictor. Classification and regression trees are used as base learners. If the outcome is continuous, aggregation is done by averaging. For a class prediction, majority voting (in the publication called *plurality voting*) is used. The multiple versions of the predictor are constructed by generating bootstrap samples of the data (drawing n out of n observations with replacement).

The gain of accuracy by this method comes from the instability of the base learners. As trees always include binary splits, they are by its construction quite dependent on the given data set. Lets go back to the iris data set and the tree in Figure 4.3. The first split is performed by the variable `Petal.Length` and divides perfectly the 50 irises of the species iris setosa from the rest. The cut point is 1.9 as the iris from this species with the biggest `Petal.Length` has the exact value of 1.9. But if we now leave this observation out, the cut point shifts to the next lower observed value of 1.7. Hence small modification of the data set can have a strong impact on the tree and therefore on future predictions. When we now modify the data set various times and save all the trees, our future prediction can be based on all the trees and will be less dependent on the original data set.

We can present the bagging procedure as following algorithm (compare to Everitt and Hothorn (2006)):

1. Draw B bootstrap samples from the original training data
2. For each of this bootstrap samples grow a very large tree
3. For prediction of a new observation pass it through all B trees and average their predictions.

Concerning accuracy, aggregated trees by the bagging method nearly always perform better than single trees. As explained ,the prediction rules get more stable as they less depend on the original data set. But also the chosen set of variables will get bigger, as also less important variables that do not enter the original tree may contribute to the aggregation.

The main tuning parameter for bagging is the number of bootstrap samples B that are used for the aggregation. Breiman (1996) in the original publication used 50 iterations but also showed in a simulated classification setup that for a grid of $\{1, 10, 25, 50, 100\}$ bootstrap replicates the main improvement concerning the miss-classification rate was already achieved with 10 iterations. *"More than 25 bootstrap replicates is love's labor lost"* (Breiman, 1996). Well, with today's computational power we can imagine much higher numbers of iterations without loosing too much time.

It makes sense to choose a reasonable high B as bagging should tend to increase stability and predictive accuracy by the number of replicates used.

4.2.3 Random forests

Random forest is a further developed ensemble method based on trees, but still has many similarities to bagging. In fact bagging can be seen as a special case of random forests.

Random forests was introduced as an ensemble method five years after bagging by the same author (Breiman, 2001). Like in bagging, classification and regression trees are used as base learners. For the modification of the data sets in each iteration bootstrap samples are drawn. The single predictors of the bootstrap samples are aggregated by averaging for regression trees and by majority voting for classification trees.

Yet, there is one further important step which helps to increase accuracy: In contrast to bagging, not every variable of $X_1, ..., X_p$ can enter in each base learner. For random forests randomness is not only included in selecting the observations for each iteration by bootstrap, but also for selecting the variables to grow the trees. In every iteration only a certain subset of covariates is randomly included. As a result, the single base learner trees differ more than with the bagging method, as there is more variation in the set of selected variables. Variables that have a relatively low effect on the response have a bigger chance to enter in the final prediction scheme in random forests than with the usual bagging method.

The amount of candidate variables that are selected for each iteration step is an additional tuning parameter. It is often called `mtry`, and usually it makes sense to choose `mtry` $< p$, where p is the number of predictor variables. To choose `mtry`$= p$ yields in bagging, which therefore can be seen as a special case of random forests.

One side-effect of choosing `mtry`$< p$ is also a less time consuming algorithm, as the computing time for trees with less variables to split on is reduced compared to trees with the full set of covariates.

Until now we tried to spare the reader too much detailed notation as the point was to get an idea about the usage of trees and ensemble methods. But as the estimation of conditional quantiles based on random forest is linked to one important step that can only be explained by a formula, we have to take a more mathematical path in the following pages.

For a better understanding how the prediction for a continuous response Y is calculated, some notations are introduced following the publications of Breiman (2001) and Meinshausen (2006):

- $\boldsymbol{\theta}$ is the random parameter vector that determines how a tree is grown
- The resulting tree is denoted as $T(\boldsymbol{\theta})$
- $\mathcal{B}$ is the space of X, so $X : \Omega \rightarrow \mathcal{B} \subseteq \mathbb{R}^p$
- $p \in \mathbb{N}_+$ is the dimensionality of the predictor variable $\boldsymbol{x}$
- Every leaf $l = 1, ..., L$ of a tree corresponds to a rectangular subspace of $\mathcal{B}$: $\mathcal{R}_l \subseteq \mathcal{B}$
- Every $\boldsymbol{x} \in \mathcal{B}$ falls in only on leaf l, and so $\boldsymbol{x} \in \mathcal{R}_l$
- This leaf is denoted $l(\boldsymbol{x}, \boldsymbol{\theta})$ for tree $T(\boldsymbol{\theta})$

One has now a sample of n observations of the predictor variables X and the continuous response Y

$$(y_i, \boldsymbol{x}_i), \quad i = 1, ..., n$$

and fits with them a regression tree $T(\boldsymbol{\theta})$. For a new data point $X = \boldsymbol{x}_{\text{new}}$, the prediction of this single tree $\hat{\mu}(\boldsymbol{x}_{\text{new}})$ is obtained by weighted averaging over the vector of observed values $\boldsymbol{y}$ in the corresponding leaf $l(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta})$.

As one only wants to include the observations that are lying in this leaf, the weight vector $w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta})$ will be defined as 0 if the corresponding $\boldsymbol{x}_i$ does not fall in the same leaf as the new data point $\boldsymbol{x}_{\text{new}}$:

$$w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}) = 0 \quad \forall i : \boldsymbol{x}_i \notin \mathcal{R}_{l(x_{\text{new}}, \boldsymbol{\theta})}$$

Furthermore, the weights should sum up to 1 over all observations:

$$\sum_{i=1}^{n} w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}) = 1.$$

Hence, we define the weights as

$$w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}) = \frac{I\{\boldsymbol{x}_i \in \mathcal{R}_{l(x_{\text{new}}, \boldsymbol{\theta})}\}}{\# \{j : \boldsymbol{x}_j \in \mathcal{R}_{l(x_{\text{new}}, \boldsymbol{\theta}}\}},$$

where $I\{\cdot\}$ denotes the indicator function and $\#\{\cdot\}$ counts the observations where the condition is fulfilled.

Hence, for a $\boldsymbol{x}^*_{\text{new}}$ that falls in a leaf where three observations of the original training data are included, these three observations will all get a weight of $\frac{1}{3}$ whereas all other observations have the weight 0.

Now these weights are used for the prediction of $E(Y|X = \boldsymbol{x}_{\text{new}})$ based on the tree $T(\boldsymbol{\theta})$:

$$\hat{\mu}(\boldsymbol{x}_{\text{new}}) = \sum_{i=1}^{n} w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}) y_i$$

For the case of the $\boldsymbol{x}^*_{\text{new}}$ that falls in a leaf with three original observations this means that $\hat{\mu}(\boldsymbol{x}^*_{\text{new}})$ is just the arithmetic mean over the y_i of these three observations. This sounds reasonable, and also for random forests the averaging process is straight-forward.

For random forests one has an ensemble of k single trees $T(\boldsymbol{\theta}_t)$, with an *i.i.d.* vector $\boldsymbol{\theta}_t$, $t = 1, ..., k$. For every single tree we can compute the weights $w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}_t)$ for all observations. Hence for $k = 1000$ trees and $n = 100$ observations one has 1000×100 weights for every new observation $\boldsymbol{x}_{\text{new}}$.

As in the end one wants to average over the y_i, it makes sense to define the mean weight $\tilde{w}_i(\boldsymbol{x}_{\text{new}})$ over the k trees:

$$\tilde{w}_i(\boldsymbol{x}_{\text{new}}) = \frac{1}{k} \sum_{t=1}^{k} w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}_t)$$

With those, one can now compute the prediction $\hat{\mu}(\boldsymbol{x}_{\text{new}})$ as the weighted mean over the observed response variables, similar to the averaging in a single tree:

$$\hat{\mu}(\boldsymbol{x}_{\text{new}}) = \sum_{i=1}^{n} \tilde{w}_i(\boldsymbol{x}_{\text{new}}) y_i$$

Like in the standard regression model, random forests do estimate the conditional mean $\mu(\boldsymbol{x}) = E(Y|X = \boldsymbol{x})$ of the response variable Y. Unlike in standard regression, no assumptions about any underlying probability distributions are needed. Additionally no additive linear effect of the form $\boldsymbol{x}^\top\boldsymbol{\beta}$ has to be expected. One can conclude that this approach is a very flexible and powerful nonparametric solution to a regression or classification problem if the main goal of the analysis is a good prediction.

But if one wants to interpret the effect of single variables or even to say something about the significance of an effect, this approach bears some problems. Single trees are easy to interpret and therefore also easy to communicate to subject-matter scientists. But the accuracy one gains by using random forests is payed by the lack of interpretability. Random forest do work more like a black box method: We put some training data in, and the box yields a powerful prediction for new observations. Yet the explanation of the form how X affects Y is more or less impossible.

Although we can not say anything about significance or appearance on an effect, there is a form to estimate the importance of single variables for the prediction called permutation importance (Strobl et al., 2007). This procedure is based on computing the difference in accuracy of the prediction if the variable of interest is permuted or not. If the accuracy is reduced dramatically, one can conclude that this variable had an important effect on the prediction. If the accuracy does stay unchanged, this indicates that there is no solid effect of this variable on the response.

4.3 Quantile regression forests

In the precedent section random forests were presented as a powerful ensemble method to predict the conditional mean $\hat{\mathbb{E}}(Y|X = \boldsymbol{x}_{\text{new}})$. The final estimation was carried out using the mean weights $\tilde{w}_i(\boldsymbol{x}_{\text{new}})$ to average over all y_i of the original data set:

$$\hat{\mathbb{E}}(Y|X = \boldsymbol{x}_{\text{new}}) = \sum_{i=1}^{n} \tilde{w}_i(\boldsymbol{x}_{\text{new}}) y_i$$

Meinshausen (2006) showed that these $\tilde{w}_i(\boldsymbol{x}_{\text{new}})$ can also be used to estimate the conditional distribution function $\hat{F}_{Y|X}(y|\boldsymbol{x}_{\text{new}})$.

This is a very meaningful discovery as only this step makes it possible to use random forest for the estimation of conditional quantiles, and therefore prediction intervals. Once we have an estimation for $\hat{F}_{Y|X}(y|\boldsymbol{x}_{\text{new}})$ it can be plugged-in to estimate $\hat{Q}_\tau(Y|X = \boldsymbol{x}_{\text{new}})$:

$$\hat{q}_\tau(\boldsymbol{x}_{\text{new}}) = \inf \left\{ y : \hat{F}_{Y|X}(y|\boldsymbol{x}_{\text{new}}) \geq \tau \right\}$$

The algorithm that leads to this final result is called *Quantile Regression Forests* (Meinshausen, 2006). Based on quantile regression forests we can construct prediction intervals of the form

$$\text{PI}(\boldsymbol{x}) = [\hat{q}_{0.025}(\boldsymbol{x}), \hat{q}_{0.975}(\boldsymbol{x})],$$

without the usual assumptions and limitations of classical prediction intervals.

With an ensemble of trees like random forest we can therefore not only estimate the conditional mean, but also the conditional quantiles. In the following section we present similar to the path from standard to quantile regression in Chapter 3 the path from random forest to quantile regression forests.

4.3.1 From random forests to quantile regression forests

To understand the concept of the usage of $\tilde{w}_i(\boldsymbol{x}_{\text{new}})$ to estimate $F_{Y|X}(y|\boldsymbol{x}_{\text{new}})$ we first repeat shortly where these weights come from:

- For every single tree we computed the weight $w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}_t)$ for $t = 1, ..., k$
- Those $w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}_t)$ are 0 for every observation i that does not lie in the same leaf as $\boldsymbol{x}_{\text{new}}$, for the rest it sums up to 1
- $\tilde{w}_i(\boldsymbol{x}_{\text{new}})$ are the mean $w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}_t)$ over the k trees
- They therefore still sum up to 1:

$$\begin{aligned}\sum_{i=1}^{n} w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}) = 1 \implies & \sum_{i=1}^{n} \frac{1}{k} \sum_{t=1}^{k} w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}_t) = 1 \\ \implies & \sum_{i=1}^{n} \tilde{w}_i(\boldsymbol{x}_{\text{new}}) = 1\end{aligned}$$

For every new observation $\boldsymbol{x}_{\text{new}}$, we can compute n $\tilde{w}_i(\boldsymbol{x}_{\text{new}})$ by dropping $\boldsymbol{x}_{\text{new}}$ down the k trees. Depending on $\boldsymbol{x}_{\text{new}}$, the weights will be big for an observation i which was often included in the same leaf as $\boldsymbol{x}_{\text{new}}$, and 0 for an observation that never got in contact with $\boldsymbol{x}_{\text{new}}$ in any leaf.

We therefore can call the $\tilde{w}_i(\boldsymbol{x}_{\text{new}})$ roughly an indicator for the similarity of $\boldsymbol{x}_{\text{new}}$ and $\boldsymbol{x}_i$. It is big for $\boldsymbol{x}_i$ that are similar to $\boldsymbol{x}_{\text{new}}$ as both will often fall in the same leaf. For observations $\boldsymbol{x}_i$ that are very different from $\boldsymbol{x}_{\text{new}}$, the $\tilde{w}_i(\boldsymbol{x}_{\text{new}})$ will be small as they are 0 in every tree where both observations did not follow the same splits.

For general random forests this information about the similarity is used to compute the weighted mean over the y_i for the prediction of $\mathbb{E}(Y|X = \boldsymbol{x}_{\text{new}})$. The observations y_i that belong to a set of predictor variables $\boldsymbol{x}_i$ similar to $\boldsymbol{x}_{\text{new}}$ will play a higher role as y_i that have a very different $\boldsymbol{x}_i$.

To use this measure of similarity also as a tool to estimate the conditional distribution function, we denote $F_{Y|X}(y|\boldsymbol{x}_{\text{new}})$ slightly different from the original definition:

$$F_{Y|X}(y|\boldsymbol{x}_{\text{new}}) = P(Y \leq y|X = \boldsymbol{x}_{\text{new}}) = \mathbb{E}(I\left\{Y \leq y\right\}|X = \boldsymbol{x}_{\text{new}})$$

Here one can already see analogies towards the expected conditional mean $\mathbb{E}(Y|X = \boldsymbol{x}_{\text{new}})$ that will be used in the estimation.

As random forests approximate the conditional mean by summing up the weighted observations

$$\hat{\mathbb{E}}(Y|X = \boldsymbol{x}_{\text{new}}) = \sum_{i=1}^{n} \tilde{w}_i(\boldsymbol{x}_{\text{new}}) y_i,$$

quantile regression forests sum up the weights itself, for every observation with $y_i \leq y$:

$$\begin{aligned}\hat{\mathbb{E}}(I\left\{Y \leq y\right\}|X = \boldsymbol{x}_{\text{new}}) &= \sum_{i=1}^{n} \tilde{w}_i(\boldsymbol{x}_{\text{new}}) I\left\{y_i \leq y\right\} \\ &= \hat{F}_{Y|X}(y|\boldsymbol{x}_{\text{new}})\end{aligned}$$

This step is the core of quantile regression forests (Meinshausen, 2006). For a simple estimation of an empirical unconditional distribution function, one just takes the proportion of $y_i \leq y$:

$$\hat{F}_Y(y) = \hat{P}(Y \leq y) = \frac{1}{n}\sum_{i=1}^{n} I\left\{y_i \leq y\right\}$$

By adding the weights for each observation, the estimation is now conditioned on $\boldsymbol{x}_{\text{new}}$, as the $\tilde{w}_i(\boldsymbol{x}_{\text{new}})$ will give more importance to observations with $\boldsymbol{x}_i$ similar to $\boldsymbol{x}_{\text{new}}$.

As a simple example let us imagine an dichotomous univariate covariate $X =$`sex` and the continuous outcome $Y =$ `body height`. If one now computes $k = 100$ random forests with `mtry`$= 1$, it is quite probable that for nearly all forests results one split by `sex`. For a new observation with $x =$`female`, the $\tilde{w}_i(x_{\text{new}})$ will therefore give weight 0 to every male observation. Hence, the resulting estimated conditional distribution $\hat{F}_{Y|X}(y|x = \texttt{female})$ will only depend on female observations.

4.3.2 The algorithm for quantile regression forests

As already presented in the precedent sections, the quantile regression forest algorithm falls in the class of ensemble methods and is based on random forests. For a given sample $y_1, ..., y_n$ of the continuous response Y and potential more-dimensional predictor variables $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ we can summarize:

1. Draw k bootstrap samples of the original data set.
2. Grow trees $T(\boldsymbol{\theta}_t)$ for every bootstrap sample $t = 1, ..., k$ as in random forest.
3. For a given $X = \boldsymbol{x}_{\text{new}}$ drop $\boldsymbol{x}_{\text{new}}$ down all trees.
4. Compute the weights for every observation in every tree:
$$w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}_t) = \frac{I\big\{\boldsymbol{x}_i \in \mathcal{R}_{l(x_{\text{new}},\boldsymbol{\theta}_t)}\big\}}{\#\big\{j : \boldsymbol{x}_j \in \mathcal{R}_{l(x_{\text{new}},\boldsymbol{\theta}_t)}\big\}}$$
5. Compute the the averaged weights for every observation:
$$\tilde{w}_i(x_{\text{new}}) = \frac{1}{k}\sum_{t=1}^{k} w_i(\boldsymbol{x}_{\text{new}}, \boldsymbol{\theta}_t)$$
6. Compute the estimation of the distribution function:
$$\hat{F}(y|X = \boldsymbol{x}_{\text{new}}) = \sum_{i=1}^{n} \tilde{w}_i(\boldsymbol{x}_{\text{new}}) I\left\{y_i \leq y\right\}$$

7. Compute the conditional quantiles as:

$$\hat{Q}_\tau(Y|X = \boldsymbol{x}_{\text{new}}) = \inf\left\{y : \hat{F}_{Y|X}(y|\boldsymbol{x}_{\text{new}}) \geq \tau\right\}$$

The presented algorithm is a powerful extension to random forests as it not only uses the averaged observations to approximate the conditional mean but provides measures to approximate the full conditional distribution function.

This aspect is used for a prediction of conditional quantiles $q_\tau(\boldsymbol{x})$. The method does not depend on the assumption of any underlying probability distribution and therefore is nonparametric. It further does not expect a linear additive effect of the form $\boldsymbol{x}^\top\boldsymbol{\beta}$.

The algorithm is made available for users with the package `quantregForest` (Meinshausen, 2007) that depends on the `randomForest` (Liaw and Wiener, 2002) package. The random forest depend therefore on regression trees as proposed by Breiman et al. (1984).

The main tuning parameters in this approach are:

- the number of trees to be grown (k or `ntree` as in `quantregForest`)
- the number of variables considered in each iteration (`mtry`).

The default settings are `ntree` $= 1000$ and `mtry` $= \frac{p}{3}$, where p denotes the number of predictor variables $X_1, ..., X_p$. Node-sizes are restricted to have more than 10 observations in each node. Meinshausen (2006) points out that the latter parameter seems not to change much, but also refers to Lin and Jeon (2002) who have shown that growing each tree until the nodes are pure leads to overfitting.

The most influential tuning parameters are `ntree` and `mtry`. It is possible to optimize `mtry` by using the out-of-bag predictions. This is a special feature for ensemble methods that use bagging to draw different random subsamples:

In every iteration, the bootstrap procedure draws n out of n observations randomly with replacement. The replacement obviously leads to observations that are more than once in the bootstrap sample and others that do not enter at all. It can be shown that asymptotically about 63.2% of the original observations are not included in the bootstrap sample. That is an important fact, as only by this variation the ensembles differ enough to give a more stable prediction in the aggregated form. But one can now use the observations not included in this iteration (out-of-bag) to test the prediction accuracy.

It is very important that the accuracy is not tested with the same data points as the method was fitted on. The best way to do this is an independent test data set, what nevertheless in practice often is not available. Another solution are cross validation methods, where parts of the data are randomly chosen not to be included in the fit, and then used to evaluate the method on. This is repeated until every data point once was excluded from fitting, the results are averaged.

In ensemble methods which use bagging, this is already possible with the out-of-bag data points and no additional cross-validation is necessary. In every iteration the base learner can be evaluated on the out-of-bag data. In our case we can now compute the trees for a grid of different `mtry` $= (p, \sqrt{p}, \frac{p}{2}, \frac{p}{3}, \frac{p}{4}, \frac{p}{5})$ and evaluate the accuracy on the data points not included in this iteration step. At the end, one can choose the value of the `mtry`-grid that achieved the best results.

4.3.3 Consistency

Meinshausen (2006) proofs the consistency of the proposed method under various assumptions. In this thesis, we will focus mainly on the coverage of prediction intervals computed by quantile regression forests in simulation studies, which can also be seen as a way to proof the consistency without using too many mathematical tricks. But nevertheless we will report the results here shortly, for details we refer to the original publication.

Assumptions:

A1: $\mathcal{B} = [0,1]^p$ and X uniform on $[0,1]^p$

A2: The proportion of observations in a node relative to all observations is vanishing for large n, $\max_{l,\theta} k_\theta(l) = o(n)$, for $n \to \infty$. The minimal number of observations in a node is growing for large n that is $1/\min_{l,\theta} k_\theta(l) = o(n)$, for $n \to \infty$.

A3: When finding a variable for a splitpoint, the probability that variable $m = 1, ..., p$ is chosen for the splitpoint is bounded from below for every node by a positive constant. If a node is split, the split is chosen so that each of the resulting sub-nodes contains at least a proportion γ of the observations in the original node, for some $0 < \gamma \leq 0.5$.

A4: There exists a constant L so that $F_{Y|X}(y|x)$ is Lipschitz continuous with parameter L that is for all x, $x' \in \mathcal{B}$,

$$\sup_x |F_{Y|X}(y|x) - F_{Y|X}(y|x')| \leq L||x - x'||_1$$

A5: The conditional distribution function $F_{Y|X}(y|x)$ is, for every $x \in \mathcal{B}$, strictly monotonously increasing in y.

If the assumptions A1-A5 are fulfilled, Meinshausen (2006) proofs that the error made by estimation of the conditional distribution converges uniformly in probability to zero:

$$\sup_{y\in\mathbb{R}} |\hat{F}_{Y|X}(y|x) - F_{Y|X}(y|x)| \xrightarrow{\text{p}} 0 \quad \forall x \in \mathcal{B}, n \to \infty$$

This result shows that quantile regression forests are a consistent way of estimating the conditional distributions and quantiles.

4.3.4 Advantages of quantile regression forests

It was already noted that quantile regression forests are a powerful tool to estimate the conditional quantiles $\hat{q}_\tau(\boldsymbol{x}) = \hat{Q}_\tau(Y|X = \boldsymbol{x})$. They do not depend on assumptions about underlying distributions and are not restricted to linear effects of the form $\boldsymbol{x}^\top\boldsymbol{\beta}$.

The algorithm made available with the package `quantregForest` is relatively fast and the usage is easy to understand also for users not familiar with ensemble methods or other computer-intensive tools.

Meinshausen (2006) proved the consistency and showed in numerical examples that the algorithm is competitive in terms of predictive power. But he did not mention at all one important advantage of his procedure:

The estimation of $q_\tau(\boldsymbol{x})$ for different τ are not independently computed but can be made simultaneously. The independently estimated parameters $\boldsymbol{\beta}_\tau$, although they are of course highly depending, is one important handicap of every method that is based on minimizing the *check-function*.

In quantile regression forests, first the conditional distribution function is estimated and afterwards plugged in to get the conditional quantiles.

$$\hat{q}_\tau(\boldsymbol{x}) = \inf\left\{y : \hat{F}_{Y|X}(y|\boldsymbol{x}) \geq \tau\right\}$$

For different τ, one uses the same $\hat{F}_{Y|X}(y|\boldsymbol{x}_{\text{new}})$ and therefore the estimation is computed simultaneously.

As the distribution function is monotonously increasing in y, quantile crossing can not occur. It is possible that for different τ, $\inf\left\{y : \hat{F}_{Y|X}(y|\boldsymbol{x}) \geq \tau\right\}$ will lead to the same value, but it can never happen that $\hat{q}_\tau(\boldsymbol{x})$ is bigger than $\hat{q}_{\tau'}(\boldsymbol{x})$ for $\tau < \tau'$.

Chapter 5

Boosting for Quantile Regression

In the preceding chapter random forests were used to combine ensembles of trees to a prediction with better performance. It was shown how this technique can be used to estimate the conditional distribution function and therefore the conditional quantiles $Q_\tau(Y|X = \boldsymbol{x})$. This is our first major tool to estimate prediction intervals by ensemble methods.

Another major tool will be component-wise boosting. For the estimation of prediction intervals, we will use it to minimize the *check-function*, which again leads to conditional quantiles.

Boosting in many aspects is a similar but broader approach than random forests. As random forests it was first introduced in the machine learning community. It got popular as a further improved tool for classification using ensembles.

In section 4.2.1. we already lined out the three main differences in ensemble methods:

- Base learners
- Ways to aggregate
- Ways to generate data in each iteration

In random forests we used trees as base learners, majority voting for aggregation and bootstrap methods to draw data in each iteration.

For boosting we do not assume any specific base learner. Besides trees also stumps (trees with only one split), regression models or splines are possible. For the aggregation information about the accuracy of the base learners is used. Very strong predictors get higher weights, whereas weak predictors will contribute less to the final method.

The main feature though is the data generation: Instead of using bootstrap samples with equal probability for all observations to enter in the ensemble, boosting selects especially the data points where the preceding predictors failed to show a good performance. Hence, the main idea is to concentrate from iteration to iteration on observations which have shown to be problematic in the past, in order to improve the accuracy of the resulting prediction.

Our focus will be on estimating conditional quantiles by minimizing the *check-function*. In the following Section 5.1. we first introduce the original AdaBoost algorithm which is mainly used as a classification tool. The next step (Section 5.2.) is the characterization of boosting as a functional gradient descent algorithm, before we finally present in Section 5.3. the algorithm for additive quantile regression which will be extensively used in the rest of the thesis.

5.1 AdaBoost

One of the first boosting algorithms was AdaBoost for classification, introduced by Freund and Schapire (1996a). It is situated in the field of ensemble methods and combines an ensemble of simple classifiers to a more powerful classification scheme. But there is an important difference to the methods described until now:

In Bagging and in Random Forest, bootstrap resamples of the observed original data were used to generate data sets for each iteration step. Therefore, each observation $(y_i, \boldsymbol{x}_i)$ for $i = 1, ..., n$ has the same chance to enter the ensemble in every iteration step.

In the boosting context we do not use bootstrap samples to generate data, but re-weight the original data points in every iteration.

5.1.1 Ensembles with re-weighted data

In the following sections, in order to achieve constistency, we will use the notation of Bühlmann and Hothorn (2007), which focused on a statistical perspective of boosting and not the one of the original publication by Freund and Schapire (1996a), which was more directed to supervised learning (classification) problems:

The base procedure (or base learner) will be denoted as $\hat{g}(.)$:

$$(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n) \overset{\text{base procedure}}{\longrightarrow} \hat{g}(.)$$

With $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)$ being the original data set with observed values of the random variables X and Y. For every iteration different data sets are generated by re-weighted data points. The prediction is computed by the base procedure on the weighted sample:

$$\begin{aligned}
\text{re} - \text{weighted data 1} &\overset{\text{base procedure}}{\longrightarrow} \hat{g}^{[1]}(.)\\
\text{re} - \text{weighted data 2} &\overset{\text{base procedure}}{\longrightarrow} \hat{g}^{[2]}(.)\\
\dots & \qquad\qquad\quad \dots\\
\dots & \qquad\qquad\quad \dots\\
\text{re} - \text{weighted data m}_{\text{stop}} &\overset{\text{base procedure}}{\longrightarrow} \hat{g}^{[m_{\text{stop}}]}(.)
\end{aligned}$$

For random forests and bagging, the weights for each data point depend on random drawing with replacement. Data points, which are not included in the iteration steps get the weight 0, the ones which are once or more often in the data set get a weight ≥ 0. They do therefore not depend on the base procedure.

For boosting the weights of the observations in every step depend on the preceding base procedures ($\equiv$ *weak learners*), in order to *boost* the relevance of problematic data points.

The intuitive idea is to alter the distribution over the domain X in a way that increases the probability of the 'harder' parts of the space, thus forcing the weak learner to generate new hypotheses that make less mistakes on these parts.
(Freund and Schapire, 1996a)

Before the invention of AdaBoost, many boosting algorithms depended on prior information about the base procedure in order to get the weights. The new concept of AdaBoost is to adjust adaptively to the errors, the base procedure made in the preceding iterations. It therefore can be called a *sequential* ensemble scheme. In every step the individual weights are updated, giving higher weights for data points for which the base procedure showed poor results in the previous iteration.

In the end, one aggregates the m_{stop} predictors with a linear combination to a single method:

$$\hat{f}_A(.) = \sum_{m=1}^{m_{\text{stop}}} \alpha_m \hat{g}^{[m]}(\cdot)$$

It is of course crucial how the α_m are computed:

A gambler, frustrated by persistent horse-racing losses and envious of his friends' winnings, decides to allow a group of his fellow gamblers to make bets on his behalf. He decides he will wager a fixed sum of money in every race, but that he will apportion his money among his friends based on how well they are doing. Certainly, if he knew psychically ahead of time which of his friends would win the most, he would naturally have

that friend handle all his wagers. Lacking such clairvoyance, however, he attempts to allocate each race's wager in such a way that his total winnings for the season will be reasonably close to what he would have won had he bet everything with the luckiest of his friends.
(Freund and Schapire, 1996a)

In the gambler example, the protagonist tries to maximize his winnings and therefore is looking for a weight vector that gives more importance to successful friends, than to ones out of luck. For a good prediction, the data analyst will try to do the same with his different base learners: We try to give the base learner of iteration m that was successful a higher α_m, than the one which delivered poor results.

5.1.2 The AdaBoost algorithm

As already mentioned the AdaBoost algorithm came to fame in the machine-learning community as a procedure to aggregate weak classifiers. Those base learners at least have to perform a little better than random guessing so that they can be combined into a powerful predictor. One of the reasons for the success of AdaBoost is that it is forcing the learning algorithm to concentrate on observations hard to classify (Freund and Schapire, 1996b).

In the original publication also an application for regression problems is discussed (algorithm AdaBoost.R), but we will focus on the more intuitive AdaBoost algorithm for binary classification (algorithm AdaBoost.M1):

1. Initialization: For the first iteration set the individual weights w_i of the observations $i = 1, ..., n$ to $w_i^{[0]} = \frac{1}{n}$, and $m = 0$.

2. Base Learner: $m = m + 1$. Compute the base learner for the weighted data set:

$$w_i^{[m]} y_i \overset{\text{base procedure}}{\longrightarrow} \hat{g}^{[m]}(\boldsymbol{x}_i) \quad \forall i$$

3. Update: Compute the weighted error-rate for the data points in the sample

$$\text{err}^{[m]} = \frac{\sum_{i=1}^{n} w_i^{[m-1]} I\left\{y_i \neq \hat{g}^{[m]}(\boldsymbol{x}_i)\right\}}{\sum_{i=1}^{n} w_i^{[m-1]}},$$

which is the proportion of observations misclassified with regarding their individual weight from the previous iteration. Misclassified data points with high w_i will contribute more to the error rate, than observations with low w_i.

The iteration m now gets its weight $\alpha^{[m]}$ for the final aggregation, depending on its success in the current iteration:

$$\alpha^{[m]} = \log\left(\frac{1-\mathrm{err}^{[m]}}{\mathrm{err}^{[m]}}\right)$$

Now the individual weights w_i can be updated:

$$\tilde{w}_i = w_i^{[m-1]} \exp\left(\alpha^{[m]} I\left\{y_i \neq \hat{g}^{[m]}(\boldsymbol{x}_i)\right\}\right)$$

If the observation i was misclassified, its weight w_i is multiplied by the factor $\left(\frac{1-\mathrm{err}^{[m]}}{\mathrm{err}^{[m]}}\right)$. Depending on the performance of the iteration, its weight increases to be of higher relevance in the next iteration.

As the updated weights should sum up to 1, they are normed:

$$w_i^{[m]} = \frac{\tilde{w}_i}{\sum_{j=1}^{n} \tilde{w}_j}$$

4. Iteration: Iterate steps 2 and 3 until $m = m_{\text{stop}}$ and compute the final classifier for a new observation $\boldsymbol{x}_{\text{new}}$ by weighted majority voting, based on the performance of each iteration in the original data:

$$\hat{f}_{\text{AdaBoost}}(\boldsymbol{x}_{\text{new}}) = \underset{y\in\{0,1\}}{\operatorname{argmax}} \sum_{m=1}^{m_{\text{stop}}} \alpha^{[m]} I\left\{y = \hat{g}^{[m]}(\boldsymbol{x}_{\text{new}})\right\}$$

As

$$\alpha^{[m]} = \log\left(\frac{1-\mathrm{err}^{[m]}}{\mathrm{err}^{[m]}}\right) \begin{cases} >0 & \text{for} \quad \mathrm{err}^{[\mathrm{m}]} < 0.5 \\ =0 & \text{for} \quad \mathrm{err}^{[\mathrm{m}]} = 0.5 \\ <0 & \text{for} \quad \mathrm{err}^{[\mathrm{m}]} > 0.5 \end{cases},$$

the algorithm will choose the $y \in \{0,1\}$ that was most often predicted in iterations where the error rate was small, and not selected when the error rate was high.

Let us repeat two main aspects to avoid confusion:

- Observations, which are often misclassified in previous iterations, have higher weights in the actual one $\rightarrow$ high $w_i^{[m]}$.
- Iterations, in which many observations are correctly specified that normally make trouble get a high weight in the final aggregation $\rightarrow$ high $\alpha^{[m]}$.

Hence, the data points that are hard to classify will get high weights to force the learning algorithm to lay focus on them. Nevertheless, for a new observation $\boldsymbol{x}_{\text{new}}$, the final classification will depend mainly on the base procedures that performed well.

This algorithm has become the most popular boosting algorithm (Bühlmann and Hothorn, 2007), often used with classification trees as base learners. The results in terms of the error rate are much better than for single trees and led Breiman (NIPS Workshop, 1996) to the statement:

AdaBoost with trees is the best off-the-shelf classifier in the world.
(Citet in Hastie et al. (2001))

5.2 Boosting as functional gradient descent

The success of AdaBoost and its high performance (not only for classification problems) produced much attention in the machine-learning world. However, it took some years until statisticians cached up and showed that this algorithm that in fact has the power to fulfill main statistical tasks, can also be fitted into classic statistical estimation schemes.

Breiman (1999, 1998) first showed that AdaBoost can be understood as an algorithm of steepest descent in function space. Later Friedman et al. (2000); Friedman (2001) included boosting in a framework of functional gradient descent for function estimation. The direct linking of boosting to (forward stagewise) additive modeling by Friedman et al. (2000) not only explained much of how and why boosting works for statistical tasks, but can also be seen as the breakthrough for boosting in the world of statistics (compare to Hofner (2008)).

The application of boosting to minimize a certain loss function, based on models, will be of major interest in this thesis, as we are interested in forms to minimize the *check-function*.

In the following section we will first present the generic functional gradient descent (FGD) algorithm, before we will show how this can be used for regression problems.

5.2.1 Generic FGD algorithm

In Chapter 2 we already have seen that the parameters in a standard linear regression model are estimated by minimizing the quadratic loss.

$$\hat{\boldsymbol{\beta}} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 \right\}$$

If we now do not want to be restricted to the squared error loss and to additive linear effects of the form $\boldsymbol{x}^\top\boldsymbol{\beta}$, we can re-formulate the problem also to

$$\hat{f}(\cdot) = \underset{f(\cdot)}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \rho(y_i, f(\boldsymbol{x}_i)) \right\},$$

where $\rho(\cdot,\cdot)$ denotes a differentiable and convex loss function and $f(\boldsymbol{x}_i)$ is some kind of predicting function that the different components of $\boldsymbol{x}_i$ often by assuming an additve structure:

$$f(\boldsymbol{x}_i) = f(x_{i1}, ..., x_{ip}) = \sum_{j=1}^{p} f_j(x_{ij})$$

Minimizing this loss function can now be carried out through boosting procedures, by always using the steepest descent in every iteration (compare to Friedman (2001); Bühlmann and Hothorn (2007); Hofner (2008)):

1. Initialization: $m = 0$. Initialize the function estimate $\hat{f}^{[0]}(\cdot)$ with an offset value. This can be a constant value c that minimizes the loss-function

$$\hat{f}^{[0]}(\cdot) = \underset{c}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \rho(y_i, c),$$

which in case of a quadratic loss yields in $c = \overline{y}$.

Another possible offset (especially for centered data) is

$$\hat{f}^{[0]}(\cdot) = 0.$$

2. Negative gradient: $m = m + 1$. Compute the negative gradient of the loss function $\rho(y_i, f)$, for every $i = 1, ..., n$, evaluated at the function values of the previous iteration $\hat{f}^{[m-1]}(\boldsymbol{x}_i)$:

$$u_i^{[m]} = - \left. \frac{\partial \rho(y_i, f)}{\partial f} \right|_{f = \hat{f}^{[m-1]}(\boldsymbol{x}_i)}$$

3. Estimation: Use the base procedure to fit the negative gradient vector $\boldsymbol{u}^{[m]} = (u_1^{[m]}, ..., u_n^{[m]})$ to $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$:

$$u_i^{[m]} \overset{\text{base procedure}}{\longrightarrow} \hat{g}^{[m]}(\boldsymbol{x}_i) \quad \forall i$$

$\hat{g}^{[m]}(\boldsymbol{x}_i)$ therefore does not estimate Y_i but the negative gradient for the data point i. $\hat{g}^{[m]}(\cdot)$ denotes the function that approximates the negative gradient vector $\boldsymbol{u}^{[m]}$.

4. Update: Update our estimation function with a step length factor $0 < \nu \leq 1$.

$$\hat{f}^{[m]}(\cdot) = \hat{f}^{[m-1]}(\cdot) + \nu \cdot \hat{g}^{[m]}(\cdot)$$

5. Iteration: Iterate steps 2 to 4 until $m = m_{\text{stop}}$.

Although the layout of the algorithm seems to be similar to AdaBoost, on first sight it is not obvious why functional gradient descent is a boosting algorithm. But let us repeat the main features of AdaBoost:

- In every iteration, the weight of each observation is adapted. Data points that are misclassified get higher weights.
- In the end, base learners which did show good results will play a higher role in the final prediction

In fact, both of these points are also fulfilled in the FGD algorithm:

In every step not only the observations get individual weights, depending on the previous iteration: The y_i are replaced by the gradient $u_i^{[m]}$ which is the negative derivate of the loss function evaluated with the y_i and the estimated function $\hat{f}^{[m-1]}(\boldsymbol{x}_i)$ of the previous iteration. Hence, data points that do make trouble in the previous step will get more important. The learning algorithm is forced to concentrate on those observations which have high effect on the loss function.

In every iteration the update step $\hat{f}^{[m-1]}(\cdot) + \nu \cdot \hat{g}^{[m]}(\cdot)$ makes sure that good base learners will play a higher role in the final prediction. Only a $\hat{g}^{[m]}()$ that is good in approximating the effect of $\boldsymbol{X}$ on $\boldsymbol{U}^{[m]}$ will yield higher values, and therefore lead the way down the steepest descent of the loss function.

Furthermore, one can also imagine that AdaBoost can be linked to additive modeling, as the final prediction is mainly just a linear additive combination of the base learners:

$$\hat{f}(\cdot) = \hat{f}^{[0]}(\cdot) + \sum_{m=1}^{m_{\text{stop}}} \nu \cdot \hat{g}^{[m]}(\cdot).$$

As this may all sound a little bit abstract, we will now focus on a more applied boosting version, which also fits in the FGD scheme and can be used for standard regression methods.

5.2.2 L_2 boosting

L_2 boosting is the application of FGD to the quadratic loss $\rho(y, f) = (y - f)^2$. It therefore yields the estimation for the conditional mean $\mathbb{E}(Y|X = \boldsymbol{x})$.

1. Initialization: $m = 0$. Initialize the function estimate $\hat{f}^{[0]}(\boldsymbol{x})$ with an offset value. For the L_2 loss, a typical offset is $\hat{f}^{[0]}(\boldsymbol{x}) = \overline{y}$

2. Negative gradient: $m = m + 1$. Compute the negative gradient of the loss function $\rho(y_i, f)$ for every $i = 1, ..., n$ evaluated at the values of the regression line of the previous iteration $\hat{f}^{[m-1]}(\boldsymbol{x}_i)$. For the L_2 loss, this yields the residuals:

$$u_i^{[m]} = -\left.\frac{\partial (y_i - f)^2}{\partial f}\right|_{f = \hat{f}^{[m-1]}(\boldsymbol{x}_i)} = 2|y_i - \hat{f}^{[m-1]}(\boldsymbol{x}_i)| \propto \hat{\varepsilon}_i^{[m-1]}$$

3. Estimation: Use the base procedure to fit the negative gradient vector $\boldsymbol{u}^{[m]} = \hat{\boldsymbol{\varepsilon}}^{[m-1]}$ to $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$:

$$\hat{\varepsilon}_i^{[m-1]} \overset{\text{base procedure}}{\longrightarrow} \hat{g}^{[m]}(\boldsymbol{x}_i) \quad \forall i$$

$\hat{g}^{[m]}(\boldsymbol{x}_i)$ therefore does not fit y_i, but the residual of the data point.

4. Update: Update the estimation function with a step length factor $0 < \nu \leq 1$.

$$\hat{f}^{[m]}(\boldsymbol{x}_i) = \hat{f}^{[m-1]}(\boldsymbol{x}_i) + \nu \cdot \hat{g}^{[m]}(\boldsymbol{x}_i)$$

5. Iteration: Iterate steps 2 to 4 until $m = m_{\text{stop}}$.

We do not have to specify which base learner we use, though it will influence the outcome. The natural choice are standard regression models $\hat{g}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\beta}$ which lead to the residuals $\varepsilon_i = (y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})$.

This application now paints a much clearer picture of the similarity to AdaBoost:

The negative gradient vector becomes the residual vector. In every iteration step we now re-weight our data, based on the previous results by simply using the residuals as new observations. They do the job of giving more weight to problematic observations, as this data points obviously do have higher residuals.

Boosting in this case means that we force the learning algorithm to take care of data points that do have high distance to the current estimation. This is just the same thing as AdaBoost forced the base learners to focus on the objects misclassified by the previous classification.

This way, we will always focus on the most problematic points concerning the loss function and leave it to the base learner to fit them, while $\hat{f}(\boldsymbol{x})$ learns in every iteration better how to handle the data. We therefore minimize the loss function $\sum_{i=1}^{n} \varepsilon_i^2$ by steepest descent, as we are handing over the ε_i to a base procedure that by itself is designed to minimize the distance from $\hat{g}(\boldsymbol{x})$ to the corresponding residuals.

In a more visual language, the algorithm walks around in the space of the loss function and searches the points where the biggest payoff for minimizing the loss could be. It finally takes the residuals and fits with them the base procedure which is the tool to go down the steepest path. The iteration of this proceeding leads to a final estimation that minimizes the corresponding loss function.

For the final prediction of a new data point we use the linear combination of the base learners fitted on the residuals. It is clear that especially iterations which showed strong effects of $\boldsymbol{X}$ on the residuals in the original data will have the highest influence in predicting the new point.

5.2.3 Example: The `cars` data

As a simple example to show how L_2 boosting works we analyze the well known `cars` data-set, available in the R package `stats` (R Development Core Team, 2009).

The task is to model the influence of the speed of a car on the stopping distance. The data was recorded in the 1920s. The speed was measured in miles per hour (mph), the distance in feet (ft). The resulting regression line from different iteration steps is presented in Figure 5.1.

One can clearly see the different steps of the algorithm. In iteration `[0]` the function is initialized by the mean stopping distance. In the later iterations the regression line is updated in every step, until it finally reaches the OLS-regression which minimizes the quadratic loss. The calculation was carried out by the function `glmboost()` in the package `mboost` (Hothorn et al., 2009).

5.2.4 Why do we need boosting?

This example was only given to illustrate the way L_2-boosting works. Nevertheless, one might ask what sense does it make to use a boosting algorithm that after a few thousand iterations reaches the regression curve of the standard linear model that can be computed in one step by $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$.

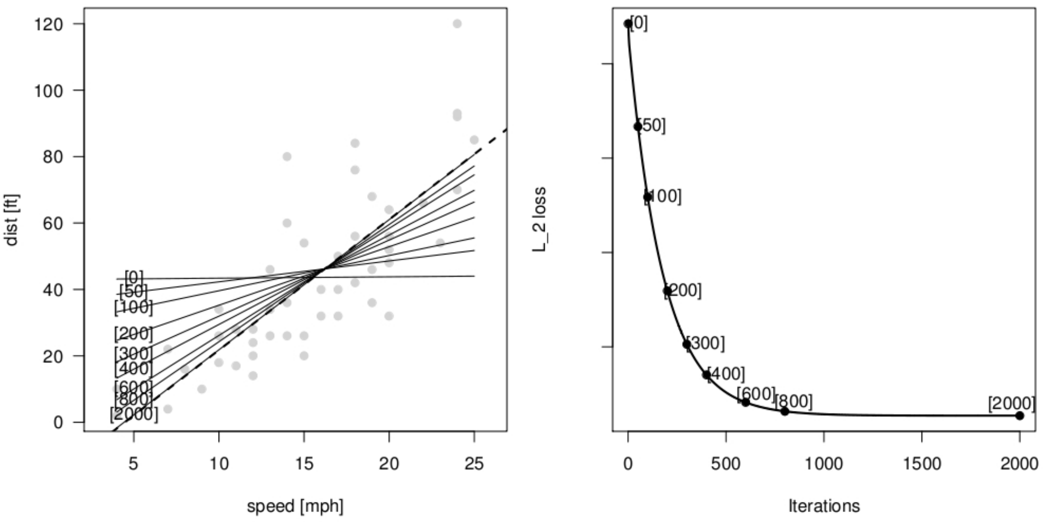

Figure 5.1: Estimation of the regression curve with L_2-boost for the classical `cars` data. **Left:** The dashed line is the result of standard OLS-regression. Solid ines represent the actual estimated function in different iteration steps of the boosting algorithm. **Right:** L_2-loss function path

The answer is: In this case it really is as taking a sledgehammer to crack a nut. But this sledgehammer is highly flexible, and can do a lot more. For motivation purposes, lets take a look at three points:

1. Different loss functions: Only the L_2 loss can be minimized in this generic form ($\boldsymbol{\beta}_{OLS} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$). Boosting through FGD is much more flexible and can work with every loss that at least is convex and differentiable. We will use it to minimize the *check-function* in the next section.

2. Penalizations or Restrictions: L_2 boosting with stopping earlier than the convergence to the standard regression curve is similar to using the lasso (Hastie et al., 2001). The lasso (Tibshirani, 1996) penalizes the sum of the absolute regression parameters with the L_1 norm:

$$\hat{\boldsymbol{\beta}}(\lambda) = \quad \underset{\boldsymbol{\beta}}{\text{argmin}} \sum_{i=1}^{n} \hat{\varepsilon}_i^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

That is basically the same as minimizing the quadratic loss and restricting the parameters to sum up to something smaller than a bound s:

$$\min \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 \quad \text{while} \quad \sum_{\mathrm{j}=1}^{\mathrm{p}} |\beta_{\mathrm{j}}| \leq \mathrm{s}$$

This minimization problem with restriction can be solved by the powerful least angle regression algorithm LARS (Efron et al., 2002). The bound s is the tuning parameter, for large s the result will be the standard OLS-estimator $\hat{\boldsymbol{\beta}}_{OLS}$, for smaller s the parameters are shrinked toward lower values. The lasso tends to leave some of β_j as 0, and therefore already includes a data driven variable selection.

When we stop the boosting algorithm early, we can achieve to get similar solutions by using component-wise boosting algorithms that will be presented in the next section.

3. p > n: For applications where we have more variables that could predict the outcome than observations to estimate the parameters, the OLS-method is not feasible.

Let us imagine a simple situation with one predictor x and two data points. The regression curve has exact one solution: The line is drawn through the two points. If we now omit one observation we have to estimate two parameters (β_0 and β_1) by one data point. There are of course solutions to this problem, but no exclusive one. Every line the passes through this one point will minimize the quadratic loss.

To get back to a distinct solution one has to include restrictions in the model, as in the lasso, or reduce the dimension of X by prior variable selection. Another way out can be component-wise boosting methods.

In the following section we will present how FGD can be used to estimate the $\boldsymbol{\beta}_\tau$ for quantile regression, by introducing a component wise boosting algorithm that minimizes the *check-function*.

5.3 Quantile regression with boosting

In the preceding section we got to know boosting as an approach of functional gradient descent. It can be used to estimate regression lines that minimize any loss function that at least is convex and differentiable.

It has some nice properties and we already stated that by component-wise boosting the solutions can include variable selection and estimations are feasible for $p > n$ problems.

In this section we will finally pass on to use boosting as a tool to estimate conditional quantiles by applying the algorithm to minimize the *check-function.* Fenske et al. (2009) showed that this technique is more than competitive to standard optimization algorithms, based on linear programming. We can therefore combine the advantages of quantile regression for nonparametric prediction intervals with the advantages of this special estimation scheme.

We will first present a component-wise boosting algorithm to estimate linear quantile regression, before it is shown how this can be adjusted to fit also nonlinear effects. In the last section we will focus on tuning parameters.

5.3.1 Linear quantile regression

With L_2-boosting we already presented an algorithm to estimate the conditional mean $\mathbb{E}(Y|X = \boldsymbol{x})$. We did not specify which base learners to use, and therefore did not restrict to linear modeling of the regression curve. We were just estimating $\hat{f}(\boldsymbol{x})$, which implied also a linear model of the form $\boldsymbol{x}^\top\boldsymbol{\beta}$.

To keep it simple, we will now first stick to the linear quantile regression $Q_\tau(Y|X = \boldsymbol{x}) = \boldsymbol{x}^\top\boldsymbol{\beta}_\tau$, as this way the idea of a component wise update is much easier to explain. Later we will adapt the algorithm also to unspecified additive nonlinear effects $Q_\tau(Y|X = \boldsymbol{x}) = \sum_{j=1}^{p} f_j(x_j)$.

To make the notational effort easier, we will omit the index τ for the quantile regression parameters $\boldsymbol{\beta}_\tau$ in the following, and call it just $\boldsymbol{\beta}$. We will also sometimes denote the linear predictor just as η_i.

It was already described in Chapter 3 that the conditional quantiles can be estimated by minimizing the *check-function* $\rho_\tau(\cdot)$:

$$\rho_\tau(y_i - \hat{\eta}_i) = \begin{cases} (y_i - \hat{\eta}_i) \cdot \tau & (y_i - \hat{\eta}_i) > 0 \\ 0 & (y_i - \hat{\eta}_i) = 0 \\ (y_i - \hat{\eta}_i) \cdot (\tau - 1) & (y_i - \hat{\eta}_i) < 0. \end{cases}$$

The path of the *check-function* for different values of τ was presented in Figure 3.1.

For applying the FGD-algorithm to the *check-function*, we will need the negative gradient:

$$-\frac{\partial \rho_\tau(y_i, \eta)}{\partial \eta} = \begin{cases} \tau & (y_i - \hat{\eta}_i) > 0 \\ 0 & (y_i - \hat{\eta}_i) = 0 \\ \tau - 1 & (y_i - \hat{\eta}_i) < 0. \end{cases}$$

Note that the *check-function* bears a severe pitfall, as it is not differentiable in 0. In practice, this can not cause too many problems as the event $y_i = \hat{\eta}_i$ due to Y being continuous will occur with zero probability. We can therefore go on with the definition of $\rho_\tau^\top(0) = 0$ without causing a substantial effect on the algorithm (Fenske et al., 2009).

With this gradient vector we can now formulate a component wise boosting algorithm for linear quantile regression (compare to Fenske et al. (2009)):

1. Initialization: $m = 0$. Initialize the $\hat{\boldsymbol{\beta}}^{[0]}$ vector. A common approach is to set $\beta_j^{[0]} = 0$ for $j = 1, ..., p$, and $\beta_0^{[0]}$ to the median of $\boldsymbol{y}$. If we use an offset value that is subtracted beforehand from $\boldsymbol{y}$ (e.g. the median), one can also set $\beta_0^{[0]} = 0$.

2. Negative gradient: $m = m + 1$. Compute the negative gradient u_i of the *check-function* $\rho_\tau(y_i, \eta_i)$ for every $i = 1, ..., n$ evaluated at the prediction of the previous iteration $\boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}^{[m-1]}$

$$u_i^{[m]} = \begin{cases} \tau & (y_i - \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}^{[m-1]}) > 0 \\ 0 & (y_i - \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}^{[m-1]}) = 0 \\ \tau - 1 & (y_i - \boldsymbol{x}_i^\top \hat{\boldsymbol{\beta}}^{[m-1]}) < 0. \end{cases}$$

3. Component-wise estimation: Use the base procedure (e.g. OLS) to fit the negative gradient vector $\boldsymbol{u}^{[m]}$ to every possible predictor variable $\boldsymbol{x}_1, ..., \boldsymbol{x}_p$ separately:

$$\boldsymbol{u}^{[m]} \overset{\text{base procedure}}{\longrightarrow} \boldsymbol{x}_j \cdot \hat{b_j}^{[m]} \quad \text{for every} \quad \text{j} = 1, ..., \text{p}$$

4. Update one component: Find the best-fitting component that minimizes the L_2-loss:

$$\hat{b_{j*}}^{[m]} = \underset{\hat{b_j}^{[m]}}{\text{argmin}} (\boldsymbol{u} - \boldsymbol{x}_j \hat{b_j}^{[m]})^\top (\boldsymbol{u} - \boldsymbol{x}_j \hat{b_j}^{[m]})$$

Now only this component $j*$ of the previous $\hat{\boldsymbol{\beta}}^{[m-1]}$ vector is updated:

$$\hat{\beta_{j*}}^{[m]} = \hat{\beta_{j*}}^{[m-1]} + \nu \hat{b_{j*}}^{[m]}$$

All the other $\hat{\beta}_j$, $j = 1, ..., p$ with $j \neq j^*$ stay the same: $\hat{\beta_j}^{[m]} = \hat{\beta_j}^{[m-1]}$.

5. Iteration: Iterate steps 2 to 4 until $m = m_{\text{stop}}$.

There are two main differences to the L_2 boosting algorithm presented in section 4.2.2.:

First obviously another loss function is used. This affects the negative gradients u_i: Instead of refitting the residuals as in L_2 boosting, for quantile regression we do refit a vector containing τ and $\tau - 1$. This sounds odd at first glance, but is just a logic conclusion to one of the main properties of quantile regression: It is robust toward outliers, therefore the magnitude of the distance from the estimated quantile in the current iteration is of no further importance. The main thing that matters is the amount of observations bigger or smaller than the quantile, which can be sufficiently expressed by the τ and $\tau - 1$ vector. The negative gradients for L_2 boosting and for quantile regression are compared for the `cars` data in Figure 4.2.

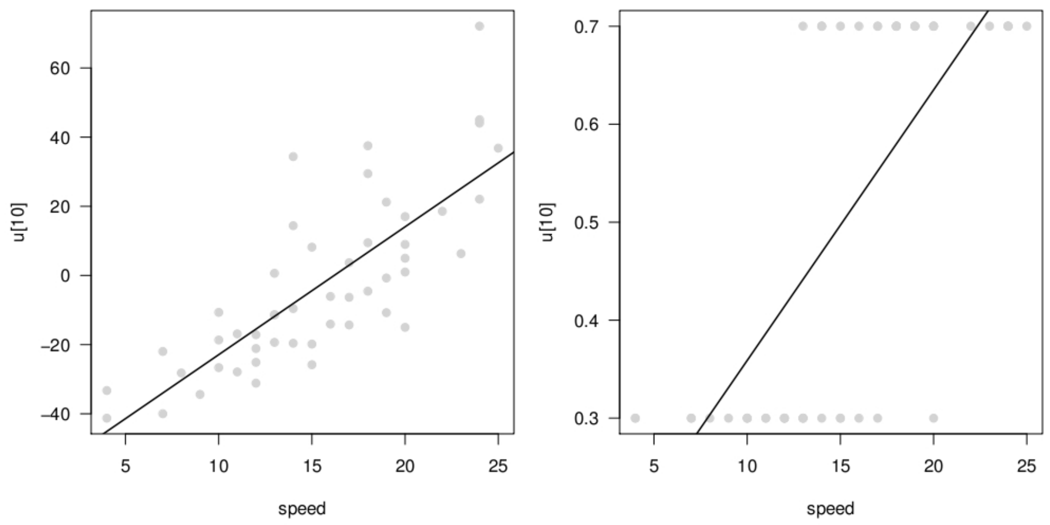

Figure 5.2: Boosting example with the `cars` data. Negative gradients after 10 iterations, fitted on the predictor `speed`. The solid line represents the OLS-fit as the base learner. **Left:** L_2 boosting, **Right:** Boosting for linear quantile regression with $\tau = 0.3$

The second difference has got nothing to do with quantile regression, but can be also implemented for normal L_2 boosting: The algorithm searches in every iteration only one component to update. It selects the predictor variable that can best reduce the L_2-loss for the gradients. This is only relevant for $p > 1$. For the example using the `cars` data set, component-wise boosting or standard boosting lead to the same results, as there is only one variable `speed` to select.

Component-wise boosting leads to some nice properties of this algorithm already mentioned in the previous section. If we stop early enough, the algorithm includes variable selection: automatically some components of $\hat{\boldsymbol{\beta}}$ stay zero, as they never were selected as the best fitting component for the gradients. This feature makes boosting also feasible for $p > n$ problems. As the algorithm always only fits the gradient component-wise to every $j = 1, ..., p$, we do not have any problem with more predictors than observations.

Another problem that can be avoided using component-wise boosting, is multicollinearity. This appears when for a multiple regression model ($p > 1$) two or more predictor variables are highly correlated. The matrix $\boldsymbol{X}^\top \boldsymbol{X}$, which is also necessary if we fit the $\boldsymbol{u}^{[m]}$ instead of the response, will not be invertible if there are linear relationships between predictor variables. If we fit only one component at a time, this problem is avoided. Furthermore both of the highly correlated variables have the chance to enter the final prediction, depending on their performance to minimize the loss function in each iteration step.

The algorithm is implemented through the function `glmboost()` in the `mboost` package (Hothorn et al., 2009). As gradient family `QuantReg()` (Fenske et al., 2009) has to be specified for quantile regression.

5.3.2 Additive quantile regression with nonlinear effects

As a next step we generalize the boosting algorithm for quantile regression also to possible nonlinear setups. As in L_2 boosting, we do not want to focus only on influences of the form $\boldsymbol{x}^\top \hat{\boldsymbol{\beta}}$, but also include a more flexible form $\sum_{j=1}^{p} \hat{f}_j(x_j)$. The only restriction we therefore make is that the effect of different predictors is additive. Boosting methods for additive models are described in Kneib et al. (2009).

The combination of additive models and quantile regression will allow us to model also prediction intervals for a given $\boldsymbol{x}_{\text{new}}$ that has a nonlinear effect on the response and the quantile. This way, we do not only avoid assumptions about underlying distribution functions of the response or the error term, but also reduce the assumptions about the effect of the predictor variables.

To distinguish between linear and nonlinear covariates we will denote in the following section observed values of continuous variables with nonlinear influence on the quantile as $\boldsymbol{z_j}$, with $j = 1, ..., q$. For observed values of common linear predictors (e.g. categorical variables) we stay with the common notation $\boldsymbol{x_l}$, with $l = 1, ..., L$. The corresponding model function therefore is:

$$Q_\tau(Y_i|\boldsymbol{x}_i, \boldsymbol{z}_i) = \eta_{\tau i} = \boldsymbol{x}_i^\top \boldsymbol{\beta}_\tau + \sum_{j=1}^{q} f_{\tau j}(\boldsymbol{z}_{ij})$$

As a result we will have to use different types of base procedures for the different sets of predictor variables. For predictor variables that will be modeled with a linear parametric effect, we will use the OLS base procedure:

$$\hat{b}_l^{[m]} = (\boldsymbol{x}_l^\top \boldsymbol{x}_l)^{-1} \boldsymbol{x}_l^\top \boldsymbol{u}$$

with $\boldsymbol{x}_l = (x_{1l}, ..., x_{nl})^\top$.

For nonlinear effects we will focus on penalized spline (P-splines) with a B-splines basis as base learners that are proposed for additive models with boosting by Schmid and Hothorn (2008).

In principle, the smoothed effect of the predictor variable $\boldsymbol{z_j} = (z_{1j}, ..., z_{nj})^\top$ is therefore modeled as a linear combination, based on B-splines of degree D on a fixed set of equidistant knots K:

$$g_j(\boldsymbol{z_j}) = \sum_{k=1}^{K} \gamma_{jk} B_k(\boldsymbol{z_j})$$

The estimation of $\hat{\boldsymbol{\gamma_j}}$ could be carried out by simple OLS:

$$\hat{\boldsymbol{\gamma_j}} = \quad \underset{\boldsymbol{\gamma_j}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(u_i - \sum_{k=1}^{K} \gamma_{jk} B_k(z_{ij}) \right)^2 \right\}$$

This leads to the best fit of $\hat{g}_j(\boldsymbol{z_j})$ on the negative gradient $\boldsymbol{u}$, but the resulting functions would be too rough. We therefore penalize the roughness and using a penalized least squares criterion (Eilers and Marx, 1996).

$$\hat{\boldsymbol{\gamma_j}} = \quad \underset{\boldsymbol{\gamma_j}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(u_i - \sum_{k=1}^{K} \gamma_{jk} B_k(z_{ij}) \right)^2 + \lambda_j \sum_{k=3}^{K} (\Delta^2 \gamma_{jk})^2 \right\}$$

Where Δ^2 is the difference of second order referring to the knots k:

$$\Delta^2 \gamma_{jk} = \gamma_{jk} - 2\gamma_{jk-1} + \gamma_{jk-2}$$

The parameter λ_j controls the amount of smoothness for the effect j. While big values for λ_j will lead to very smooth results with a relatively high bias and low variance, small values or even $\lambda_j \to 0$ will lead to a very rough line with low bias but high variance.

With the B-spline design matrix $\boldsymbol{Z}_j$, and the penalty matrix $\boldsymbol{K}$ with $\boldsymbol{K} = \boldsymbol{D_2}^\top \boldsymbol{D_2}$

$$\boldsymbol{Z}_j = \begin{pmatrix} B_1(\boldsymbol{z_{1j}}) & \cdots & B_K(\boldsymbol{z_{1j}}) \\ \vdots & & \vdots \\ B_1(\boldsymbol{z_{nj}}) & \cdots & B_K(\boldsymbol{z_{nj}}) \end{pmatrix} \quad \boldsymbol{D}_2 = \begin{pmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{pmatrix}$$

the minimization problem can also be formulated as:

$$\hat{\boldsymbol{\gamma}_j} = \underset{\boldsymbol{\gamma_j}}{\text{argmin}} \left\{ (\boldsymbol{u} - \boldsymbol{Z_j}\boldsymbol{\gamma}_j)^\top (\boldsymbol{u} - \boldsymbol{Z_j}\boldsymbol{\gamma}_j) + \lambda_j \boldsymbol{\gamma}_j^\top \boldsymbol{K} \boldsymbol{\gamma}_j \right\}$$

This can be solved to (Fenske et al., 2009)

$$\hat{g}_j^{[m]} = \boldsymbol{Z}_j(\boldsymbol{Z}_j^\top \boldsymbol{Z}_j + \lambda_j \boldsymbol{K})^{-1} \boldsymbol{Z}_j^\top \boldsymbol{u}^{[m]},$$

where $\boldsymbol{u}^{[m]}$ is the negative gradient vector in iteration step m. This is our base procedure for a nonlinear effect $f_j(\boldsymbol{z}_j)$.

As we have now defined the base-procedures for linear and nonlinear effects, we can formulate the algorithm for estimating additive quantile regression by component-wise boosting:

1. Initialization: $m = 0$. Initialize the $\hat{\boldsymbol{\beta}}^{[0]}$ vector and the estimated function $\hat{f}_j^{[0]}$ for $j = 1, ..., q$.

2. Negative gradient: $m = m + 1$. Compute the negative gradient vector $\boldsymbol{u}^{[m]}$, as in the linear case:

$$u_i^{[m]} = \begin{cases} \tau & (y_i - \eta_{\tau i}^{[m-1]}) > 0 \\ 0 & (y_i - \eta_{\tau i}^{[m-1]}) = 0 \\ \tau - 1 & (y_i - \eta_{\tau i}^{[m-1]}) < 0. \end{cases}$$

3. Component-wise estimation: Use the base procedures to fit the negative gradient vector $\boldsymbol{u}^{[m]}$ to every possible predictor variable $\boldsymbol{x}_1, ..., \boldsymbol{x}_l, \boldsymbol{z}_1, ..., \boldsymbol{z}_q$ separately:

$$\boldsymbol{u}^{[m]} \xrightarrow{\text{base procedure}} \begin{cases} \hat{b}_l^m & \boldsymbol{x}_1, ..., \boldsymbol{x}_L \\ \hat{g}_j^m & \boldsymbol{z}_1, ..., \boldsymbol{z}_q \end{cases}$$

4. Update one component: Find the best-fitting component that minimizes the L_2-loss

$$(\boldsymbol{u}^{[m]} - \hat{\boldsymbol{u}}^{[m]})^\top (\boldsymbol{u}^{[m]} - \hat{\boldsymbol{u}}^{[m]})$$

inserting $\boldsymbol{x}_l \hat{b}_l^m = \hat{\boldsymbol{u}}^{[m]}$ for linear predictor variables and $\hat{g}_j^m(\boldsymbol{z}_j) = \hat{\boldsymbol{u}}^{[m]}$ for nonlinear components.

If the best fitting base learner is a linear effect with index l^*, update the corresponding coefficient vector:

$$\hat{\beta_{l*}}^{[m]} = \hat{\beta_{l*}}^{[m-1]} + \nu \hat{b_{l*}}^{[m]}$$

If the best fitting base learner is a nonlinear effect with index j^*, update the corresponding regression function:

$$\hat{f}_{j*}^{[m]} = \hat{f}_{j*}^{[m-1]} + \nu \hat{g}_{j^*}^{[m]}$$

All the other $\hat{\beta}_l$, $l = 1, ..., L$ with $l \neq l^*$ or $\hat{f}_j$, $j = 1, ..., q$ with $j \neq j^*$ stay constant.

5. Iteration: Iterate steps 2 to 4 until $m = m_{\text{stop}}$.

The algorithm is implemented through the function `gamboost()` in the `mboost` package (Hothorn et al., 2009). To apply it for quantile regression, the gradient family `family=QuantReg()` (Fenske et al., 2009) has to be specified.

As for the linear case, the algorithm fits the negative gradients component-wise and updates only the component that achieves the biggest descent of the *check-function* in every iteration. In the nonlinear case we therefore also profit from the nice properties of a component-wise boosting algorithm as the included variable selection. The algorithm can deal with high dimensional data sets and $p > n$ problems while multicollinearity is avoided.

Additive quantile regression based on smoothing splines normally is computed by linear programming, as implemented in the function `rqss()` of the `quantreg` package (Koenker, 2009b). Fenske et al. (2009) showed in simulation studies that the boosting approach leads to comparable performance results, but noted that it is more flexible in estimating the nonlinear effects. Furthermore, the currently available implementation of the boosting technique can handle a larger number of nonlinear predictor variables.

As an example for applying boosting to estimate quantile regression with a nonlinear effect, we use the already presented `cars` data set (compare to Section 5.2.3). We are now interested in modeling the 0.3-quantile curve for the stopping distance conditioned on the speed of the car. We do not assume a linear effect, and therefore use the boosting algorithm with P-splines as base learners. Results for different iteration steps are presented in Figure 5.3.

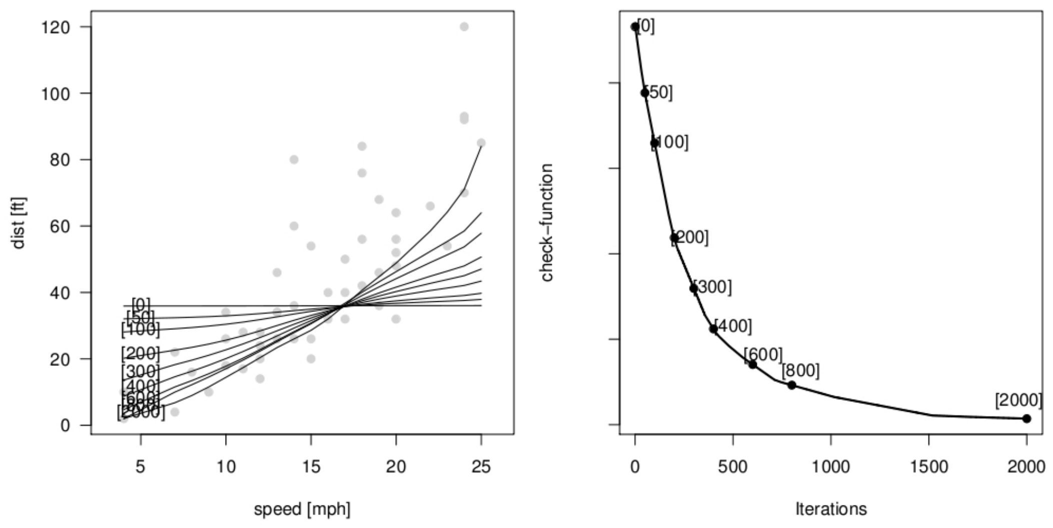

Figure 5.3: Estimation of the conditional 0.3-quantile curve with `gamboost()` for the classical `cars` data. **Left:** Lines represent the actual estimated function in different iteration steps of the boosting algorithm. **Right:** Path of the *ckeck-function*$(\tau = 0.3)$ as the adequate loss

As in Figure 5.1. that presented linear L_2-boosting for the same data, one can clearly see that the algorithm needs a certain number of iterations, until it finally results in a reasonable fit. One can also see that the *check-function* as the loss is reduced with every iteration step.

We can now use the conditional quantile $\hat{Q}_{0.3}(Y|X = x) = \hat{f}(x)$ as a one sided 70% nonparametric prediction interval for a new observation x_{new}. We therefore interpret the modeled quantile conditioned on a observed $X = x$ as a function in x. Hence, the lower boundary for a prediction interval for a new observation can be computed by simply plugging in x_{new}:

$$\text{PI}(x_{\text{new}}) = [\hat{q}_{0.3}(x_{\text{new}}), \infty[$$

The correct conditional interpretation (Chapter 2) is that for a new car with the speed x_{new} the computed $\text{PI}(x_{\text{new}})$ surrounds its stopping distance with a probability of 0.7. Hence, in 70% of the cases a new car with exactly this speed x_{new} will not stop earlier than $\hat{q}_{0.3}(x_{\text{new}})$.

5.3.3 Tuning parameters for boosting with P-spline base learners

With the algorithm for estimating conditional quantiles for boosting with P-spline base learners one has a powerful tool to estimate prediction intervals based on quantiles. It combines the advantages of quantile regression by not assuming any underlying distribution function and the properties of the component-wise boosting approach.

The algorithm already implies data-driven variable selection and model choice and therefore no a priori specification about the predictor variables has to be carried out. Categorical variables will enter as nonlinear effects, continuous variables are candidates for smooth effects. The smoothness of each effect j is controlled by λ_j that is automatically determined in the iteration steps.

Nevertheless, there are tuning and hyper parameters in this approach that have to be set. We will briefly discuss the possible ways to choose them with laying focus on the practical relevance. For a more complete view on the subject including theoretical implications we refer to Schmid and Hothorn (2008).

There a basically two main tuning parameters for boosting which trade off each other: The step length factor ν and the number of boosting iterations m_{stop}.

In early boosting algorithms it was proposed to estimate an adequate step-length $\nu = \nu(m)$ in every iteration step (Friedman, 2001). But in newer works it was shown that this time-consuming procedure is of low importance for the predictive accuracy (Bühlmann and Hothorn, 2007). It was also noted that there is a dependence between m_{stop} and ν. For the case of early stopping applied based on a stopping rule (e.q. AIC or evaluation on a test data set), Schmid and Hothorn (2008) give the theoretical background why the dependence is approximately linear, with m_{stop} inversely proportional to ν. Hence, a two times bigger ν will result in half of the necessary iteration steps m_{stop} to reach the same fit. This was also confirmed in our simulation results. Fenske et al. (2009) therefore argue that one can safely fix $\nu = 0.1$ as it is the default value in `mboost` and focus on finding an optimal choice for m_{stop}. We will follow this approach of a fixed ν, but the question is how this fixed value should be chosen.

As a small ν is contributing to the shrinkage effect in boosting methods, ν should not be chosen too big as this could have negative effects on the prediction performance (Hofner, 2008). Yet, we want to report an important detail for boosting quantile regression, which could also motivate a slightly higher ν:

The negative gradients for minimizing the *check-function* by the gradient family `family=QuantReg()` in `mboost` are defined for every $m < m_{\text{stop}}$ as:

$$u_i^{[m]} = \begin{cases} \tau & (y_i - \eta_{\tau i}^{[m-1]}) > 0 \\ 0 & (y_i - \eta_{\tau i}^{[m-1]}) = 0 \\ \tau - 1 & (y_i - \eta_{\tau i}^{[m-1]}) < 0. \end{cases}$$

Hence, modeling the median leads to a vector with entries -0.5 and 0.5. In comparison when using `family=Laplace()`, which focuses on minimizing the L_1 loss, which should also yield the median, the resulting vector consists of -1 and 1. This eventually leads to the same fit, yet the `Laplace()` version will be approximately twice as fast. This can be resolved by using $\nu = 0.2$ instead of $\nu = 0.1$ when applying `family=QuantReg()`.

One therefore should keep in mind that for quantile regression we already fit relatively small gradients, which also leads to small base learners $\hat{b}^{[m]}$ or $\hat{g}^{[m]}(\cdot)$. A slightly bigger ν than for normal L_2 boosting therefore could decrease the computing time without having a too big effect on the predictive accuracy. This view is also supported by our simulation results (Chapter 6).

As we have fixed ν, the selection of an adequate m_{stop} therefore is of even higher relevance. Early stopping for component-wise boosting implies variable selection and model choice, and can be linked to successful penalized regression techniques as the lasso (Tibshirani, 1996). As the algorithm will first pick the most influential predictor variables in early iterations, stopping at the right time will lead to final predictions that include the shrinked effects of the most important variables without touching the irrelevant ones.

In `mboost` (Hothorn et al., 2009) selecting m_{stop} is implemented by applying AIC measures. Yet this method is not feasible for nonparametric quantile regression. For our simulation study, we will select m_{stop} based on a test data set. We then choose the value which minimizes the empirical risk (regarding the *check-function* as loss) on the the test data (compare to Fenske et al. (2009)). For real world problems where no test data set is available, also cross validation can be used.

Another parameter to be specified is the offset. In `mboost` the default for quantile regression is the median of the response. Fenske et al. (2009) discussed that for quantiles with $\tau < 0.5$ specifying the corresponding τ-quantile itself as offset leads to the same results in the same amount of iteration steps. Yet for quantiles with $\tau > 0.5$ the same approach makes the algorithm much slower. That means that more iteration are needed for the same fit. Therefore, it is reasonable to stick to the median as default offset for every τ.

For linear quantile regression specifying ν, the offset and a procedure to select m_{stop} is already sufficient to start boosting. Yet for additive quantile regression with nonlinear effects using P-splines as base learners additionally two hyper parameters for the splines can be specified:

- The degrees of freedom of the B-spline basis (df): Bühlmann and Yu (2003) argued that smooth base learners should have large bias but low variance. This way, the stagewise adaption of the final prediction function will yield a better prediction. They therefore recommend to use a small fixed value of df for the base learners of every predictor variable. The default setting in `mboost` is df$= 4$. In our analysis we will stick to this setting.

- The number of equidistant knots: As we are using penalized least squares the impact of the amount of knots should be reduced. Schmid and Hothorn (2008) showed that choosing $20 - 50$ knots is enough to obtain a good boosting fit. The default value in `mboost` is 20. In our analysis we will stick to this setting.

To sum up, one can use the default values of `mboost` for the offset and the hyper parameters of the splines. For the step-length ν the default value of 0.1 can be used, but we have noted that especially for linear quantile regression we could reduce the computing time by increasing ν without achieving negative effects on the predictive accuracy. The reason are the small negative gradients $\boldsymbol{u}$ resulting from the *check-function* that by themselves reduce the step length.

The most important tuning parameter is m_{stop}. We will focus on its impact on the predictive accuracy in detail in Chapter 6.

Chapter 6

Simulation Studies

After presenting the idea of constructing prediction intervals by applying quantile regression and two different estimation schemes using ensemble methods, we can now go over to test this tools concerning consistency and their practical usage. We will do this in practice by analyzing the visitors to a new movie in Chapter 7, but the more flexible approach in terms of comparing different data setups and tuning parameters will be testing the coverage on simulated data sets.

There are two main questions that we want to explore:

1. How do prediction intervals estimated by quantile regression forests and by boosting perform in different setups concerning the coverage and the correct estimation of the conditional quantiles:
   - Linear effects of one or multiple predictor variables including high-dimensional situations with more predictors than observations ($p > n$)
   - Additive potential nonlinear effects of one or multiple predictor variables
2. How do the two major tuning parameters of boosting affect the performance of prediction intervals?

We will explore the first question in various simulation studies based on different data setups. As we are interested in nonparametric prediction intervals, we will focus on situations where their parametric counterparts are not available, as fundamental assumptions of standard linear regression as homoscedasticity or gaussian distributed error terms are not fulfilled.

For the second question we will compare for selected setups different settings for the step length ν and the number of boosting iterations m_{stop} regarding the performance of resulting intervals.

But how do we measure the performance of prediction intervals? We already introduced in Chapter 2 two different interpretations for prediction intervals for new observations:

- **The conditional interpretation:** A prediction interval $\mathrm{PI}_{1-\alpha}(\boldsymbol{x}_{\mathrm{new}})$ covers new observations with this specific $\boldsymbol{x}_{\mathrm{new}}$ with a probability of $1-\alpha$. To confirm this coverage we need many observations with the same predictor variables. In real world data sets this can be impossible. As an example imagine a clinical study with genetic variables as predictors for a certain continuous response. The only other observation with exactly the same predictors could be the patient's twin. Yet in simulation studies obtaining many observations of Y for a given $X = \boldsymbol{x}_{\mathrm{new}}$ is no problem. Then the conditional probability of the new observation falling in the interval can be estimated:

  For $(\boldsymbol{y}, \boldsymbol{x}_{\mathrm{new}})$ with $\boldsymbol{y} = (y_1, ..., y_n)^\top$:

$$\begin{aligned}\hat{P}(Y \in \mathrm{PI}(\boldsymbol{x}_{\mathrm{new}})|X = \boldsymbol{x}_{\mathrm{new}}) &= \hat{\mathbb{E}}\left(Y \in \mathrm{PI}(\boldsymbol{x}_{\mathrm{new}})|X = \boldsymbol{x}_{\mathrm{new}}\right) \\ &= \frac{\#\left\{\boldsymbol{y} \in \mathrm{PI}(\boldsymbol{x}_{\mathrm{new}})\right\}}{n}\end{aligned}$$

- **The heuristic sample interpretation:** Prediction intervals $\mathrm{PI}_{1-\alpha}(\boldsymbol{x}_{\mathrm{new}})$ cover new observations with the probability of $1-\alpha$. The size of the interval depends on the predicting variables $\boldsymbol{x}_{\mathrm{new}}$, giving information about the accuracy of the estimation. The $1-\alpha$ level refers to a new sample: $(1-\alpha)\cdot 100\%$ of the new observations in the sample will be covered by the corresponding intervals. This sample coverage can be confirmed by a sample with different realizations of X:

  For a sample $(\boldsymbol{y}, \boldsymbol{x})$ with $\boldsymbol{y} = (y_1, ..., y_n)^\top$ and $\boldsymbol{x} = (x_1, ..., x_n)^\top$:

$$\begin{aligned}\hat{P}(Y \in \mathrm{PI}(\boldsymbol{x})) &= \hat{\mathbb{E}}(Y \in \mathrm{PI}(\boldsymbol{x})) \\ &= \frac{\sum_{i=1}^{n} I\left\{y_i \in \mathrm{PI}(\boldsymbol{x}_i)\right\}}{n}\end{aligned}$$

Those two interpretations imply different ways how simulation studies should be designed. Let us assume we already estimated a function that yields in some form a correct specified prediction interval if we plug in the predictor variables of a new observation $\boldsymbol{x}_{\mathrm{new}}$:

$$\mathrm{PI}_{1-\alpha}(\boldsymbol{x}_{\mathrm{new}}) = [\hat{q}_{\alpha/2}(\boldsymbol{x}_{\mathrm{new}}), \hat{q}_{1-\alpha/2}(\boldsymbol{x}_{\mathrm{new}})]$$

To confirm the sample coverage, we would just take this method and plug in every $\boldsymbol{x}$ of a new sample. The result is one prediction interval for every new data point. The observed response vector $\boldsymbol{y}$ of the sample can now be used to check if in fact the correct amount of observations do fall inside the intervals. We will finally have the proportion of points lying inside the intervals over the whole sample.

For the conditional interpretation, we would select a handful of $\boldsymbol{x}$ and compute the corresponding intervals. Afterwards we simulate many observations for these specific $X = \boldsymbol{x}$ in order to estimate the coverage of the interval at these points. Therefore the result will be a conditional coverage for every single point.

To further illustrate the difference in the two approaches, we simulated another toy example:

$$Y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + (\boldsymbol{x}_i^\top \boldsymbol{\alpha})\varepsilon_i \quad \text{where} \quad \varepsilon_i \overset{\text{iid}}{\sim} N(0,1) \quad \text{for} \quad i = 1, ..., n$$

The components of $\boldsymbol{x}_i$ are observed realizations of the random variable $X \sim U(0,1)$. We chose a heteroscedastic setup with three predictor variables: $\boldsymbol{\beta} = (1,2,3,4)^\top, \quad \boldsymbol{\alpha} = (1,2,1,1)^\top$.

For both approaches we simulated training data consisting of 1500 data points. We modeled the conditional 0.1-quantile and the conditional 0.9-quantile with linear component-wise boosting on the training data.

For the sample coverage we drew a second data set consisting of 1000 data points to evaluate the intervals. As explained above, we pluged in the 1000 $\boldsymbol{x}$ in PI$(\cdot)$ resulting in 1000 intervals. For the conditional coverage we selected five predictor vectors $(\boldsymbol{x}_1, ..., \boldsymbol{x}_5)$ and estimated five prediction intervals. Afterwards we simulated for every one of these predictor vectors 100 new realizations of $Y|X = \boldsymbol{x}$. Results are presented in Figure 6.1.

In the plot of the heuristic sample interpretation the multiple predictor variables and the heteroscedasticity make the borders of the intervals appear not like linear functions but more like clouds. As the two quantiles were estimated separately, the size of the intervals can differ very much, even for data points with very similar $\boldsymbol{x}$.

For the conditional coverage we selected the five $\boldsymbol{x}$ also based on the sample ordered by the size of the intervals. We took the 100th, 300th, 500th, 700th and 900th observation of the ordered test data and simulated the additional realizations of $Y|X = \boldsymbol{x}$ afterwards.

The presented example stands for one simulation step: As the conditional quantiles for $\text{PI}_{1-\alpha}(\boldsymbol{x}_{\text{new}}) = [\hat{q}_{\alpha/2}(\boldsymbol{x}_{\text{new}}), \hat{q}_{1-\alpha/2}(\boldsymbol{x}_{\text{new}})]$ were estimated based on the information of a training data set, it is not sufficient to show that the method works by evaluating prediction intervals based on this one data set. We will repeat the presented design for different training data and different variable setups to proof the consistency of the proposed methods.

We pointed out in Chapter 2 that only the conditional interpretation is adequate when it comes to prediction intervals conditional on predicting variables as we are

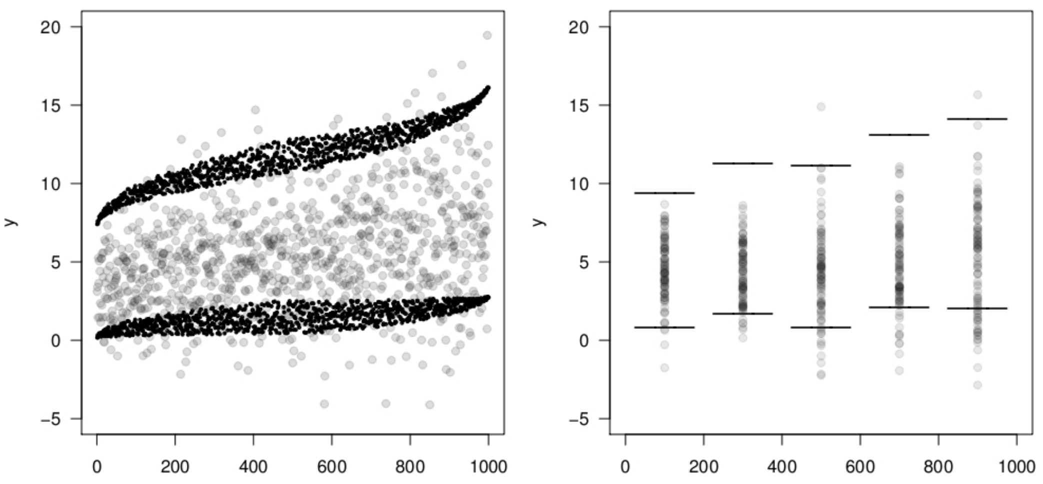

Figure 6.1: Example of a simulation step to compare the two interpretations of prediction intervals. Grey points represent data points in the test data, small dark points the borders of prediction intervals estimated on the training data. The x-axis is the index of the observation in the test data, the sample is ordered by the size of the corresponding prediction interval. **Left:** Heuristic sample interpretation **Right:** Conditional interpretation, horizontal lines represent the borders of the intervals

interested in. Nevertheless, Meinshausen (2006) used the heuristic sample interpretation to confirm the coverage of prediction intervals based on random forests. We call this interpretation *heuristic* as we clearly stated that this view is not incorrect, as of course correct specified intervals should hold the coverage over a whole sample with different x. Yet we also gave an example how looking only at the sample coverage can be misleading, as also intervals can cover $(1-\alpha)\cdot 100\%$ of a new sample that are completely out of range for many specific $\boldsymbol{x}_{\text{new}}$.

For the real world data set used in Chapter 7 we will primarily analyze the sample coverage, as the conditional approach is not feasible. Yet in the simulation studies in this chapter we will first take a look at the adequate conditional coverage for different setups (Section 6.1) before we will show that the proposed methods in fact in most setups also work for the heuristic interpretation (Section 6.2).

6.1 Conditional coverage

We already pointed out several times that the borders of prediction intervals are random variables, as they depend on a sample. It is therefore not enough to estimate a prediction interval on one training data set and proove its consistency on test data. The resulting coverage still depends on the training data. In simulation studies we therefore have to repeat the estimation process several times for different training data sets. Roughly speaking we want to get rid of this dependence on the training sample. As in probability theory we get rid of a nuisance parameter by integrating over it, in simulation practice we iterate over it and middle the result afterwards.

For the conditional interpretation we use following simulation design:

1. **Test data:**
   - Choose five realizations $\boldsymbol{x}_t$ of X to test the conditional coverage on for $t = 1, ..., 5$
   - Simulate 10000 realizations $\boldsymbol{y}_{\text{test}}$ of $Y|X = \boldsymbol{x}_t$ for each $t = 1, ..., 5$
2. **Simulation step:**
   - Simulate a training data set consisting of 1000 observations and estimate the conditional quantiles
   - Use the conditional quantiles to construct prediction intervals for Y_{new}: on $\text{PI}(\boldsymbol{x}_1), ..., \text{PI}(\boldsymbol{x}_5)$
   - Evaluate the prediction intervals on $\boldsymbol{y}_{\text{test}}$ by

$$\text{Coverage}_{\boldsymbol{x_t}} = \frac{\#\{\boldsymbol{y}_{\text{test},t} \in \text{PI}(\boldsymbol{x}_t)\}}{10000} \quad \text{for} \quad t = 1, ...5$$

3. **Iteration:**
   - Repeat the simulation step 100 times

The results of a simulation following this design will be a set of 500 prediction intervals with 100 independent intervals for each of the five test realizations of X we want to confirm the conditional coverage on. For every of this five points we have 10000 new test observations to evaluate the coverage. We therefore get 100 estimations for the conditional coverage of $\text{PI}(\boldsymbol{x}_{\text{new}})$ for five different $\boldsymbol{x}_{\text{new}}$. Every coverage still depends on a training sample, but by middling afterwards we get rid of this dependence.

We will focus on nonparametric prediction intervals based on quantile regression as presented in this thesis. As we want to compare the boosting and the random forest approach for quantile regression, we will test these two methods: Inside each simulation step we will estimate the PI($\boldsymbol{x}$) two times, once with the package `mboost` (Hothorn et al., 2009) with `family=QuantReg()` (Fenske et al., 2009) following the boosting approach and once with the package `quantregForest` (Meinshausen, 2007) following the random forest appoach.

For quantile regression forest we will choose the tuning parameters as proposed by Meinshausen (2006): The main parameters are the number of trees `ntree` and the number of candidate variables considered to split on in every tree `mtry`. We use an ensemble of `ntree` = 1000 trees. In every tree, one third of the available predictor variables are considered. The size of final nodes is restricted to contain more than 10 observations.

In Section 5.3.3 we already focused on the selection of tuning parameters for the boosting approach. We will fix the step length ν and optimize m_{stop}. For this purpose we will use an additional data set to evaluate the fit for every boosting iteration (compare to Fenske et al. (2009)) . In every simulation step another optimization data set is drawn.

Simulation step for boosting:

- Simulate a training data set consisting of 1000 observations and estimate the conditional quantiles by `mboost` with a high fixed m_{stop}.
- Simulate another data set to optimize m_{stop} consisting of 1000 observations and evaluate the empirical risk (concerning the *check-function*) for the model of every boosting iteration of the fit on the training data.
- Select the m_{stop} that minimizes the empirical risk on the optimization data set.

Additionally to the coverage, we will also use two other measures of performance to compare the two algorithms. We will focus on the goodness of fit of the conditional quantiles in the test data, by estimating the bias and the mean squared error (MSE):

$$\text{Bias}(\tau) = \frac{1}{5 \cdot 100} \sum_{t=1}^{5} \sum_{j=1}^{100} \left(\hat{q}_{\tau}^{(j)}(\boldsymbol{x}_t) - q_{\tau}(\boldsymbol{x}_t) \right)$$

$$\text{MSE}(\tau) = \frac{1}{5 \cdot 100} \sum_{t=1}^{5} \sum_{j=1}^{100} \left(\hat{q}_{\tau}^{(j)}(\boldsymbol{x}_t) - q_{\tau}(\boldsymbol{x}_t) \right)^2$$

Where $\hat{q}_{\tau}^{(j)}(\boldsymbol{x}_t)$ denotes the estimated conditional quantile of $Y|X = \boldsymbol{x}_t$ in simulation step j.

These two measures as defined here can be seen as monte carlo estimators of the true bias and MSE of $\hat{q}_\tau(\cdot)$ (Fenske et al., 2009). The monte carlo estimation in this conditional setup is limited, as it refers only to five realizations of X. As it will serve only to compare the two ensemble methods presented here, evaluated at the same five points, this limitation can be accepted.

6.1.1 One predictor variable with linear effect

For the first data setups we focus on a single predictor variable with a linear effect on the response:

$$Y_i = \beta_0 + \beta_1 x_i + (\alpha_0 + \alpha_1 x_i)\varepsilon_i$$

The parameter $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)^\top$ therefore controls the variance and the heteroscedasticity of the error term. As we will focus on nonparametric intervals, we are especially interested in setups that pose problems to parametric prediction intervals and therefore will only use setups with heteroscedastic error terms. In this section we will present results of two setups:

- Gaussian setup: $\boldsymbol{\beta} = (1,2)^\top, \boldsymbol{\alpha} = (2,5)^\top$ and $\varepsilon_i \overset{\text{iid}}{\sim} N(0,1)$
- Gamma setup: $\boldsymbol{\beta} = (1,2)^\top, \boldsymbol{\alpha} = (2,5)^\top$ and $\varepsilon_i \overset{\text{iid}}{\sim} G(2,5)$

x_i are observed realizations of a random variable $X \sim U(0,1)$.

The task is to model one-sided nonparametric 80% prediction intervals for the response Y, we will therefore focus on the estimation of the conditional 0.2-quantile of $Y|X = x_{\text{new}}$:

$$\text{PI}_{0.8}(x_{\text{new}}) = [\hat{q}_{0.2}(x_{\text{new}}), \infty[$$

To test the resulting prediction intervals regarding the conditional interpretation, we will focus on a grid of five realizations of X:

$$x_1 = 0.1, \quad x_2 = 0.25, \quad x_3 = 0.5, \quad x_4 = 0.75, \quad x_5 = 0.9.$$

For the boosting approach we fixed the step length $\nu = 0.5$ and optimized m_{stop} on an optimization data set separately in every simulation step. As the maximum m_{stop} we chose a priori 2000 iterations for this setup. Based on the optimization data the algorithm stopped on average after 935 iterations for the gaussian setup and after 1610 iterations for the gamma-distributed error term. We also looked at the models without early stopping which gave nearly exactly the same results. In this specific setup with only one predictor variable early stopping does not seems to be of very high relevance.

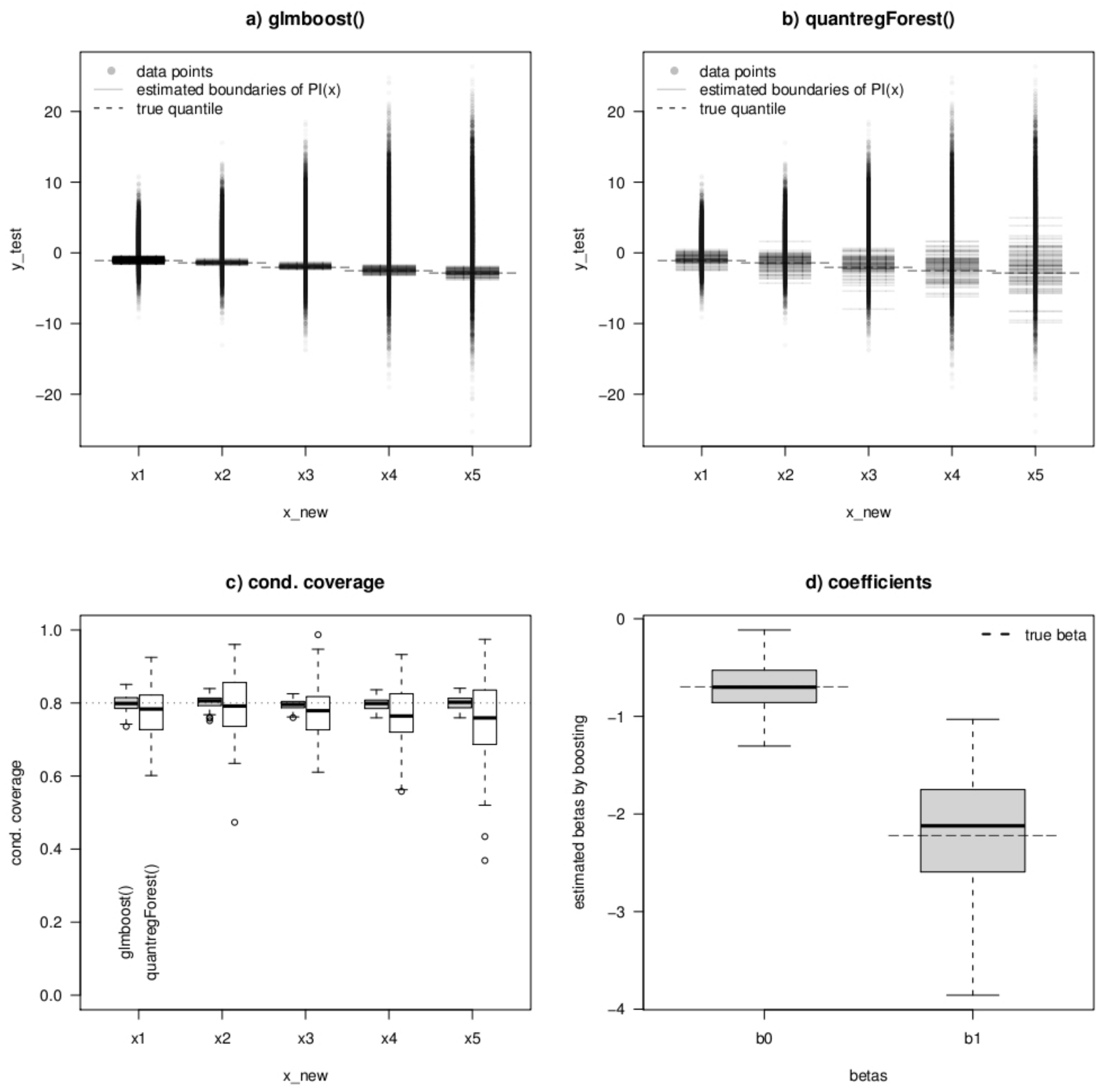

Figure 6.2: Results for the gaussian setup (one predictor variable with linear effect)

Additionally, we also analyzed results on the same data of the gaussian setup with fixed $\nu = 0.1$ and a maximum m_{stop} of 10000 iterations, in order to investigate the relationship between the two main tuning parameters of the boosting approach. Results concerning the conditional coverage are quasi identical, the only difference is that the mean optimized m_{stop} turns out to be 4403. This is nearly exact five times as much as for the $\nu = 0.1$ case, confirming the linear relationship between step length and number of iterations.

Results for the different intervals are presented in Table 6.1. Both approaches give reasonable results concerning the mean conditional coverage in the test grid. Yet the boosting results are clearly closer to 0.8 in every single point of the test grid. Also

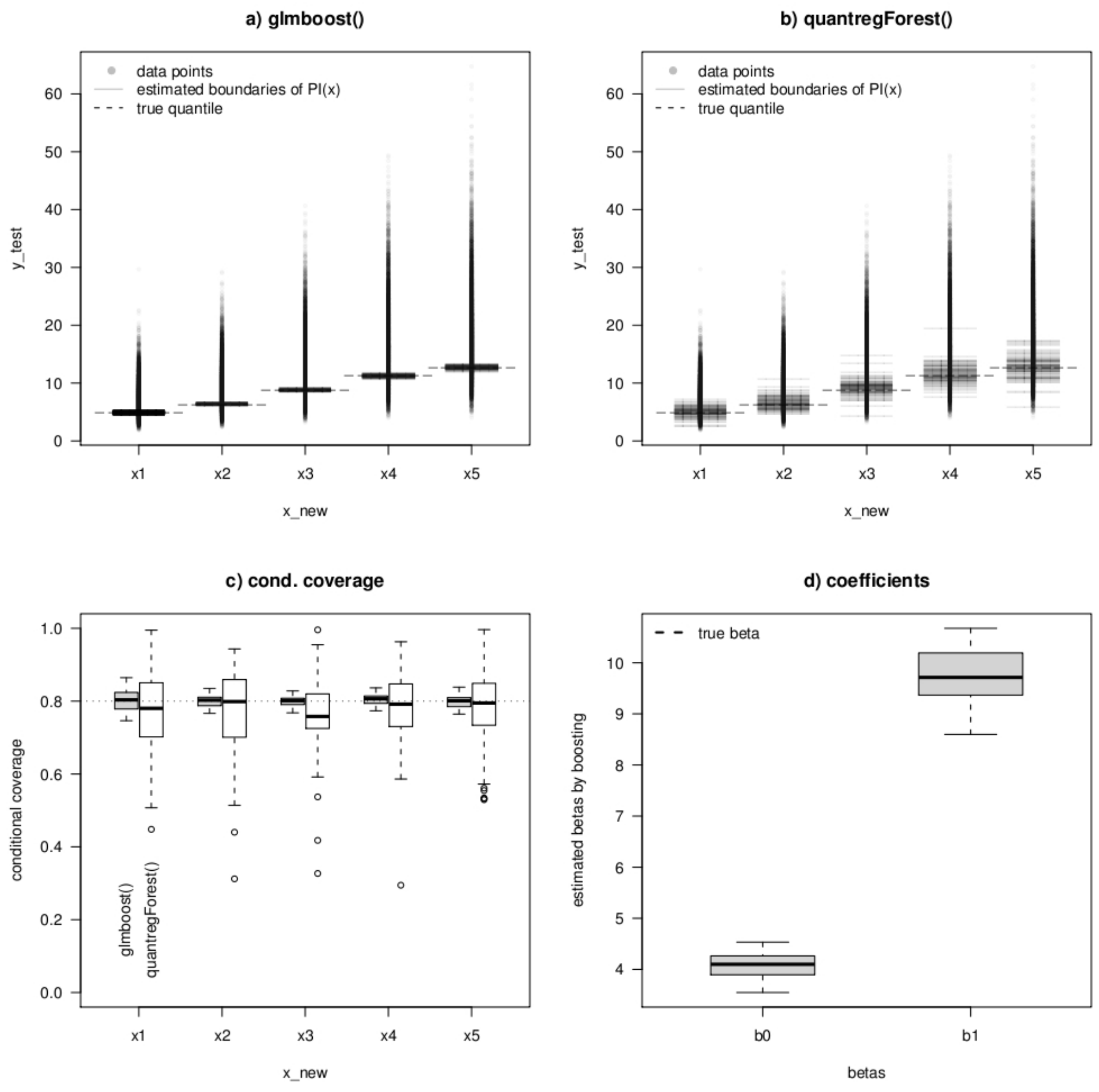

Figure 6.3: Results for the Gamma setup (one predictor variable with linear effect)

concerning the fit of the conditional quantile boosting performed better. MSE as well as the Bias show that in this case linear quantile regression by boosting is superior to quantile regression forests. A reason for that surely can be found in the data setup. Random forests are an ensemble method based on trees. They gain accuracy by forcing the algorithm not to choose in every ensemble the same variables to split on, by always leaving some out. Random forest therefore should perform better for multiple predictor variables and possible interactions. The boosting approach for linear quantile regression focuses on estimating $\hat{q}_\tau(x) = \hat{\beta}_{0\tau} + \hat{\beta}_{1\tau} \cdot x$, which in this setup is the superior estimation scheme.

For graphical illustration we will focus on 4 graphics for each setups (Figure 6.2 and

	Gaussian setup		Gamma setup	
	`glmboost`	`quantregForest`	`glmboost`	`quantregForest`
Coverage$_{x_1}$	**0.7978**	0.7740	**0.8022**	0.7748
Coverage$_{x_2}$	**0.8039**	0.7946	**0.8006**	0.7700
Coverage$_{x_3}$	**0.7956**	0.7770	**0.7994**	0.7591
Coverage$_{x_4}$	**0.7978**	0.7679	**0.8051**	0.7798
Coverage$_{x_5}$	**0.8003**	0.7563	**0.7986**	0.7848
Bias($\tau = 0.2$)	**0.2308**	1.1397	**-0.0208**	0.2167
MSE($\tau = 0.2$)	**0.0919**	2.5739	**0.0484**	1.9866

Table 6.1: Results of the simulation studies for both setups, comparing `glmboost()` and `quantregForest()`. In every row the value of the better performing algorithm for each setup is printed bold.

Figure 6.3.). They compare the conditional coverage of the intervals computed by both algorithm and analyze the goodness of the estimations for $\boldsymbol{\beta}_\tau$ in the linear-boosting case. We will use these graphics for nearly every setup in the this entire Section 6.1:

a) For every of the five test points of the X-grid the 10000 observations of $Y|X = x_t$ are plotted as points. Horizontal lines represent the conditional quantiles estimated by `glmmboost()` in the different simulation steps. The longer dashed line is the true conditional quantile.

b) The same as **a)** but with `quantregForest()` used to compute the conditional quantiles

c) Boxplots of the conditional coverage in the five points of the test grid for 100 simulation steps. The grey-colored boxplots represent the results of the boosting approach whereas the white figures stand for the approach using quantile regression forests. The dotted line is the proposed coverage the intervals should hold.

d) Boxplots of the estimated coefficients by the boosting approach in the 100 simulation steps. Dashed lines represent the true $\boldsymbol{\beta}_\tau$

In this setup, we can also observe by looking at Figure 6.2 and Figure 6.3. why the performance of `glmboost` is better. The estimated conditional quantiles are for both setups much closer to the true quantile than with the random forest approach, which in some simulation steps delivers quite poorly estimated quantiles which also leads to a too low or too high coverage. One can notice that in some simulation steps the conditional coverage of selected points falls beyond 40% and sometimes is close to 100%. For the boosting approach the conditional coverage does not differ that much from the proposed 80%. This is a result of the very precise estimation of $\boldsymbol{\beta}_\tau$ as can be seen in sector **d)** of the figures.

6.1.2 Multiple predictor variables with linear effect

For setups with multiple predictors that have a linear effect on the response variable we generate data following this model, similar to the one in the previous section:

$$Y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + (\boldsymbol{x}_i^\top \boldsymbol{\alpha})\varepsilon_i$$

With $\boldsymbol{x}_i = (1, x_{1i}, x_{2i}, x_{3i})^\top$ and every component independently drawn as a realization of $X \sim U(0,1)$.

We will again focus on situations including heteroscedasticity in two different setups:

- Gaussian setup: $\boldsymbol{\beta} = (1,2,3,4)^\top, \boldsymbol{\alpha} = (1,1,2,1)^\top$ and $\varepsilon_i \overset{\text{iid}}{\sim} N(0,1)$
- Gamma setup: $\boldsymbol{\beta} = (1,2,3,4)^\top, \boldsymbol{\alpha} = (1,1,2,1)^\top$ and $\varepsilon_i \overset{\text{iid}}{\sim} G(2,5)$

To test the conditional coverage we selected five possible realizations of X:

$$\begin{array}{ll} \boldsymbol{x}_1 = (1, 0.1, 0.1, 0.1)^\top & \boldsymbol{x}_2 = (1, 0.25, 0.25, 0.25)^\top \\ \boldsymbol{x}_3 = (1, 0.5, 0.5, 0.5)^\top & \boldsymbol{x}_4 = (1, 0.75, 0.75, 0.75)^\top \\ \boldsymbol{x}_5 = (1, 0.9, 0.9, 0.9)^\top & \end{array}$$

In this setup we will focus on constructing one-sided as well as two-sided 80% prediction intervals:

- One-sided $\text{PI}_{0.8}(\boldsymbol{x}_{\text{new}}) = [\hat{q}_{0.2}(\boldsymbol{x}_{\text{new}}), \infty[$
- Two-sided $\text{PI}_{0.8}(\boldsymbol{x}_{\text{new}}) = [\hat{q}_{0.1}(\boldsymbol{x}_{\text{new}}), \hat{q}_{0.9}(\boldsymbol{x}_{\text{new}})]$

For the boosting approach we fixed $\nu = 0.5$ and raised the maximum m_{stop} to 4000 iterations for the one-sided intervals. For estimating $\hat{q}_{0.2}(\boldsymbol{x}_{\text{new}})$ the optimized m_{stop} resulted to be 2989 iterations for the gaussian approach and 3196 for the gamma-setup. For the two-sided intervals the maximum number of iterations was set to be 10000. For $\hat{q}_{0.1}(\boldsymbol{x}_{\text{new}})$ the optimal m_{stop} was 5211 in the gamma setup (3257 in the gaussian) while for $\hat{q}_{0.9}(\boldsymbol{x}_{\text{new}})$ 8015 iterations were necessary in the gamma case (4062 for the gaussian).

Results for one-sided intervals are presented in Table 6.2. As in the previous section, the boosting approach in this situation leads to much better results. The conditional coverage for both algorithm and both setups is good in the center of the X-grid $(\boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4)$. For $\boldsymbol{x}_1$ and $\boldsymbol{x}_5$ boosting still yields good results, while `quantregForest` seems to overestimate the quantiles for small $\boldsymbol{x}$ and underestimate it for big $\boldsymbol{x}$ in the Gamma setup.

	Gaussian setup		Gamma setup	
	`glmboost`	`quantregForest`	`glmboost`	`quantregForest`
One-sided				
Coverage$_{\boldsymbol{x}_1}$	**0.7864**	0.7317	**0.7919**	0.6310
Coverage$_{\boldsymbol{x}_2}$	**0.8011**	0.7808	**0.7963**	0.7810
Coverage$_{\boldsymbol{x}_3}$	**0.7977**	0.7822	**0.7982**	0.7976
Coverage$_{\boldsymbol{x}_4}$	**0.8022**	0.7834	**0.8055**	0.8152
Coverage$_{\boldsymbol{x}_5}$	0.8051	**0.7990**	**0.8001**	0.8561
Bias($\tau = 0.2$)	**−0.0152**	0.1300	**−0.0063**	-0.0892
MSE($\tau = 0.2$)	**0.0773**	0.4824	**0.0490**	0.7115
Two-sided				
Coverage$_{\boldsymbol{x}_1}$	**0.7828**	0.8207	**0.7890**	0.7637
Coverage$_{\boldsymbol{x}_2}$	0.7930	**0.7945**	**0.7967**	0.8089
Coverage$_{\boldsymbol{x}_3}$	**0.7978**	0.7669	0.8044	**0.7986**
Coverage$_{\boldsymbol{x}_4}$	**0.8002**	0.7408	**0.8021**	0.7839
Coverage$_{\boldsymbol{x}_5}$	**0.8014**	0.7213	**0.7973**	0.7675
Bias($\tau = 0.1$)	**−0.0181**	0.1835	**−0.0070**	-0.1818
MSE($\tau = 0.1$)	**0.1147**	0.6382	**0.0430**	0.5956
Bias($\tau = 0.9$)	**−0.0101**	-0.2028	**0.0164**	-0.0493
MSE($\tau = 0.9$)	**0.1118**	1.3251	**0.2660**	3.6071

Table 6.2: Results of the simulation studies concerning one-sided and two sided 80% prediction intervals for both setups, comparing `glmboost()` and `quantregForest()`. In every row the value of the better performing algorithm for each setup is printed bold.

This view is supported by Figure 6.4, illustrating the results for one sided intervals in the Gamma setup with the same graphics as in the previous section. We omitted plotting also the figures for the gaussian setup as the results are similar. We can clearly see here the problems the random forest approach has to predict observations with more extreme values of X, as $\boldsymbol{x}_1$ and $\boldsymbol{x}_5$. Even though the estimation of $\hat{\boldsymbol{\beta}}_\tau$ seems to be excellent by the boosting approach, we can also observe that the coverage for the outer values of the X-grid has a slightly broader range, and in some simulation steps yielded poorer results than in the one predictor variable setups in the previous section (Figure 6.3).

Results of the two sided intervals are presented in the lower part of Table 6.2. Of course, as two sided intervals imply estimation of two conditional quantiles, the results concerning the coverage get less accurate. Once again the boosting approach is clearly superiour. Interestingly the random forests seem to have more problems with the gaussian setup than with the setup of the heavy skewed gamma error term.

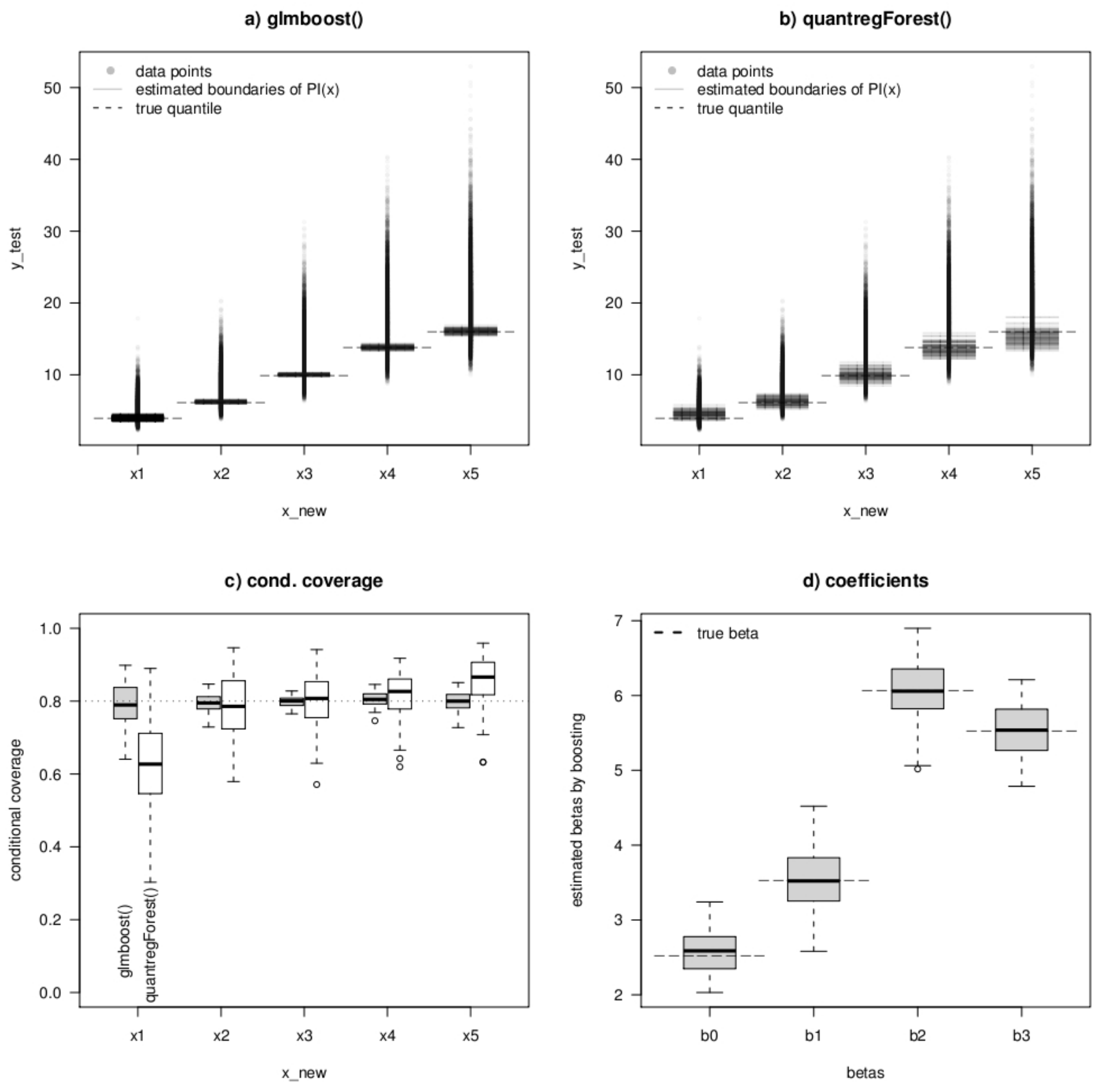

Figure 6.4: Results for one-sided intervals in the Gamma setup (multiple predictor variables with linear effect)

In Figure 6.5 it gets clear that both algorithms do have more problems with the upper boundary of the intervals, i.e. the 0.9-quantile than with the lower boundary. The reason for this is probably the heavy right-skewed Gamma distribution for the error term, which results in many observations lying near the lower boundary than near the upper. As for the one-sided intervals we can note a tendency for the random forest approach to overestimate quantiles on the smaller (left) side of the X grid and underestimate them on the upper (right) side.

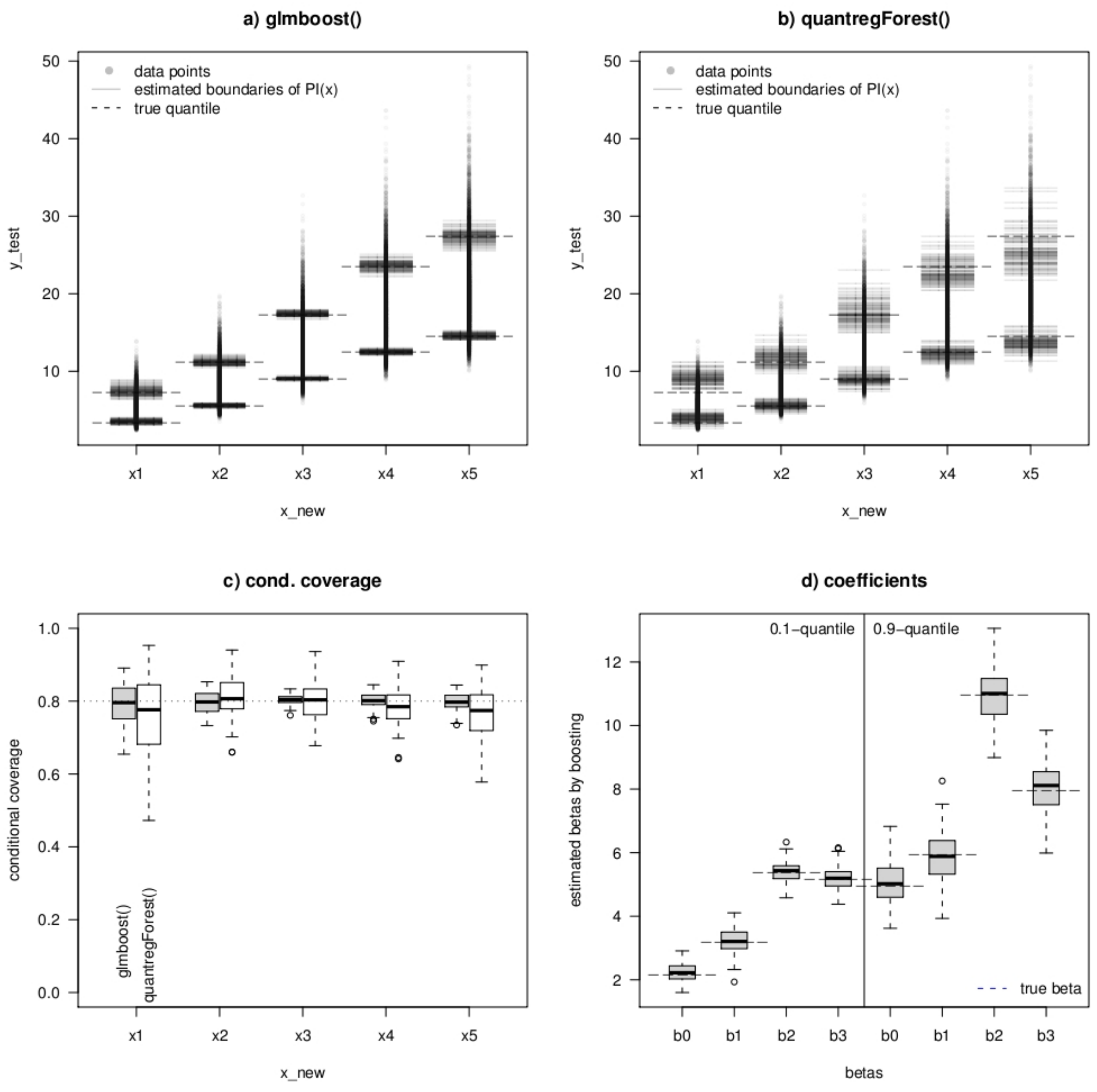

Figure 6.5: Results for two-sided intervals in the Gamma setup (multiple predictor variables with linear effect)

6.1.3 Multiple predictor variables with linear effect, including variable selection

As a next step we will focus on one of the main features of the component-wise boosting approach, which is the included data-driven variable selection. As the algorithm in every iteration only updates the coefficient that achives the steepest decent concerning the loss function, variables with minor or no effect on the response should be left out.

Data is generated following the already presented model including heteroscedasticity:

$$Y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + (\boldsymbol{x}_i^\top \boldsymbol{\alpha})\varepsilon_i$$

With $\boldsymbol{x}_i = (1, x_{1i}, ..., x_{20i})^\top$ and every component independently drawn as a realization of $X \sim U(0,1)$.

To investigate the variable selection effect we will use following setup including 20 predictor variables of which 10 have no effect on the response:

- $\boldsymbol{\beta} = (0, -5, -4, -3, ..., 3, 4, 5, \overbrace{0, ..., 0}^{10})^\top, \quad \boldsymbol{\alpha} = (1, \overbrace{0.5, ..., 0.5}^{10}, \overbrace{0, ...0}^{10})^\top$
  $\varepsilon_i \overset{\text{iid}}{\sim} N(0,1)$ with $n = 1000$ training observations

It is important that the $\boldsymbol{\alpha}$ vector is also 0 for the 10 predictor variables that should have no effect on the response. As for the normal distributed error term

$$\boldsymbol{\beta}_\tau = \boldsymbol{\beta} + \boldsymbol{\alpha}\Phi^{-1}(\tau)$$

the resulting component of $\boldsymbol{\beta}_\tau$ would not be 0 if the corresponding component of $\boldsymbol{\alpha}$ is not.

As in the foregoing setups we consider five possible realizations of X to test the conditional coverage. We try to cover the whole range of X by selecting:

$$\begin{array}{ll} \boldsymbol{x}_1 = (1, \overbrace{0.1, ..., 0.1}^{20})^\top & \boldsymbol{x}_2 = (1, 0.25, ..., 0.25)^\top \\ \boldsymbol{x}_3 = (1, 0.5, ..., 0.5)^\top & \boldsymbol{x}_4 = (1, 0.75, ..., 0.75)^\top \\ \boldsymbol{x}_5 = (1, 0.9, ..., 0.9)^\top & \end{array}$$

	`glmboost`		`quantregForest`
	m_{stop} opt.	$m_{\text{stop}} = 15000$	
Coverage$_{\boldsymbol{x}_1}$	**0.7252**	**0.7252**	0.9759
Coverage$_{\boldsymbol{x}_2}$	**0.7772**	**0.7772**	0.8926
Coverage$_{\boldsymbol{x}_3}$	0.7959	0.7959	**0.7979**
Coverage$_{\boldsymbol{x}_4}$	**0.8092**	**0.8092**	0.7573
Coverage$_{\boldsymbol{x}_5}$	**0.8150**	**0.8150**	0.7663
Bias($\tau = 0.2$)	0.0064	**0.0062**	-0.3438
MSE($\tau = 0.2$)	**0.2320**	0.2321	1.6138

Table 6.3: Results for the variable selection setup. Results of the two boosting approaches (once with early stopping and once without) do differ mostly only after the fourth digit. The values of the best performing algorithm in each row is printed bold.

For the boosting approach we fixed the step length at $\nu = 0.5$ and set the maximum m_{stop} to 15000. In the simulation the mean optimized m_{stop} was 6609. We compared results for models with the maximum m_{stop} with the ones with early stopping and found nearly no difference. Hence, overfitting at least does not seem to kick in very fast in this setup.

Results for one-sided 80% prediction intervals are presented in Table 6.3. As the two boosting approaches mostly only differ after the fourth digit, there is hardly any difference between stopping after 3000 to 10000 iterations and directly taking 15000 itereations as a fixed m_{stop}.

Compared to `quantregForest()` boosting by `glmboost()` also in this setup is clearly better performing. The prediction intervals of both estimation schemes seem to have problems to hold the conditional coverage for outer values of the X grid as $\boldsymbol{x}_1$ and $\boldsymbol{x}_5$. But as boosting seems to overestimate the conditional 0.2-quantile for $\boldsymbol{x}_1$, quantile regression forests underestimates it. The result is a too high coverage for random forests as boosting fails to reach the proposed 0.80 level at this point.

This can also be observed in Figure 6.6 **a)** and Figure 6.6 **b)**. The resulting intervals from `quantregForest` fail to adopt to the different values of $\boldsymbol{x}$. They yield more or less the same conditional quantile for $\boldsymbol{x}_1$ to $\boldsymbol{x}_5$, although the true quantiles differ significantly. If we look only at the conditional coverage at $\boldsymbol{x}_3$, which is better than for the boosting approach, we did not notice that.

Finally we look at the estimated coefficients from the boosting approach (Figure 6.6 **d)**): The variable selection seems to work, as the components $\beta_{11} - \beta_{20}$ are mostly estimated as 0. Therefore the corresponding predictor variables are correctly identified to have no effect on the response. The estimations for the non-zero coefficients seems to be as good as in previous settings.

The only coefficient that apparently makes some trouble is the intercept. As the true β_0 is 0, and α_0 is 1 due to the gaussian error term the correct $\beta_{0\tau}$ is $\Phi^{-1}(\tau)$. For $\tau = 0.2$ this yields -0.84162. Yet the median of resulting estimations is -0.259. This is no dramatic bias, yet compared to the well estimated other coefficients this result makes suspicious. In fact we have seen in different setups with multiple predictor variables and high variance of the response similar problems concerning the intercept.

It is possible that the relatively poor results (0.7252) of the boosting approach concerning the conditional coverage for $\boldsymbol{x}_1$ can be partly explained by this bias. As for $\boldsymbol{x}_1 = (1, 0.1, ..., 0.1)^\top$ the other coefficients are less important on the estimation of the conditional quantile compared to $\boldsymbol{x}$ with bigger components (e.g. $\boldsymbol{x}_3 = (1, 0.5, ..., 0.5)^\top$), a small bias in the estimation of the intercept has a stronger effect for $\boldsymbol{x}_1$ than for $\boldsymbol{x}_3$.

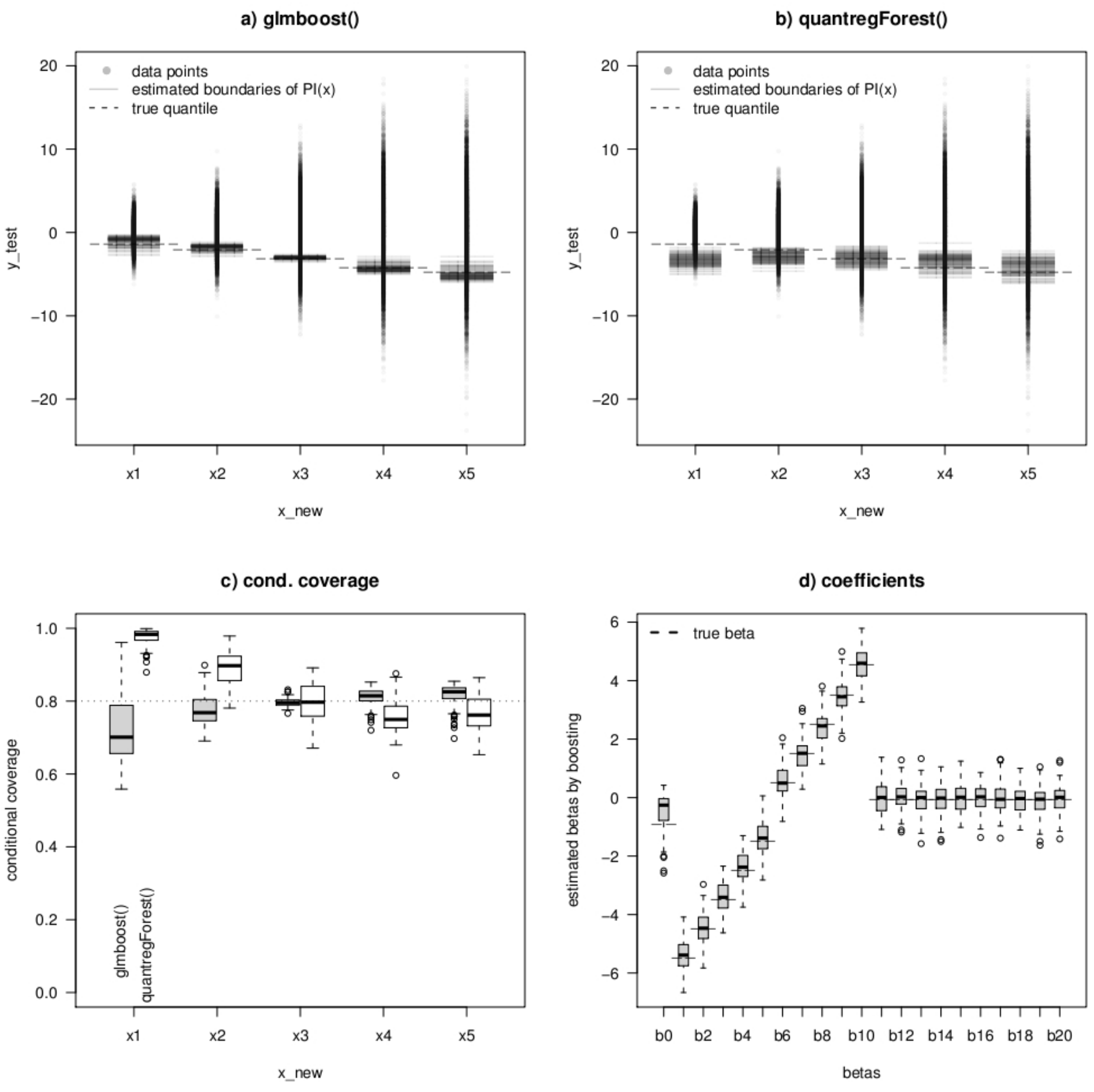

Figure 6.6: Results for the variable-selection setup (multiple predictor variables with linear effect)

Of course we do not want to negate the overall very accurate estimation of coefficients by `glmboost()` for quantile regression, by pointing out these seemingly minor difficulties for β_0. But it is simply the discrepancy between the accuracy of the estimations for common coefficients and the intercept that makes it unlikely to be an artifact of no relevance. Another illustration of this intercept problem can be observed in the high dimensional setups of the following section.

6.1.4 High dimensional data with linear effects, including $p > n$

Before focusing on nonlinear effects we want to illustrate another advantage of non-parametric prediction intervals. Ensemble methods as random forests and boosting can cope with high dimensional data setups, including situations with more predictive variables than observations ($p > n$). We will focus on one data setup with two different sample sizes.

$$Y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + (\boldsymbol{x}_i^\top \boldsymbol{\alpha})\varepsilon_i \quad \varepsilon_i \overset{\text{iid}}{\sim} N(0,1)$$

With $\boldsymbol{x}_i = (1, x_{1i}, ..., x_{200i})^\top$ and every component independently drawn as a realization of $X \sim U(0,1)$.

- $n = 150$, $\boldsymbol{\beta} = (0, -10, -9, -8, ..., 8, 9, 10, \overbrace{0, ..., 0}^{180})^\top$, $\boldsymbol{\alpha} = (1, \overbrace{2, ..., 2}^{20}, \overbrace{0, ..., 0}^{180})^\top$
- $n = 1000$, $\boldsymbol{\beta} = (0, -10, -9, -8, ..., 8, 9, 10, 0, ..., 0)^\top$, $\boldsymbol{\alpha} = (1, 2, ..., 2, 0, ..., 0)^\top$

Hence we use two times the same data setup, just with different sample sizes. Once the sample size is reduced to just 150 observations in the training data to construct a situation with more predictor variables than observations. In a second step we use $n = 1000$ as in the other setups so far to investigate the effect of an increased sample for high dimensional data.

	n=150		$n = 1000$	
	`glmboost`	`quantregForest`	`glmboost`	`quantregForest`
Coverage$_{\boldsymbol{x}_1}$	**0.6488**	0.9991	**0.9149**	0.9999
Coverage$_{\boldsymbol{x}_2}$	**0.7071**	0.9256	**0.8322**	0.9213
Coverage$_{\boldsymbol{x}_3}$	0.7212	**0.7463**	**0.7842**	0.7531
Coverage$_{\boldsymbol{x}_4}$	**0.7294**	0.7117	**0.7689**	0.7125
Coverage$_{\boldsymbol{x}_5}$	**0.7329**	0.7180	**0.7564**	0.7133
Bias($\tau = 0.2$)	5.1670	**-0.1446**	1.0350	**-0.0385**
MSE($\tau = 0.2$)	**39.5668**	109.3197	**12.0492**	97.5515

Table 6.4: Results for the high dimensional data setup with two different sample sizes. The values of the best performing algorithm in each row is printed bold for each sample size separately.

To test the conditional coverage we selected five realizations of X:

$$\begin{aligned} &\boldsymbol{x}_1 = (1, \overbrace{0.1, ..., 0.1}^{200})^\top \quad && \boldsymbol{x}_2 = (1, 0.25, ..., 0.25)^\top \\ &\boldsymbol{x}_3 = (1, 0.5, ..., 0.5)^\top \quad && \boldsymbol{x}_4 = (1, 0.75, ..., 0.75)^\top \\ &\boldsymbol{x}_5 = (1, 0.9, ..., 0.9)^\top \end{aligned}$$

As in previous cases we modeled nonparametric one-sided 80% intervals by estimating the conditional 0.2-quantile $\hat{q}_{0.2}(\boldsymbol{x})$. For the boosting approach we selected following tuning parameters:

- $n = 150$: Maximum $m_{\text{stop}} = 15000$, $\nu = 0.5$. The mean optimum m_{stop} selected was 6140
- $n = 1000$: Maximum $m_{\text{stop}} = 35000$, $\nu = 0.5$. The mean optimum m_{stop} selected was 24630

The size of the optimization data set to select m_{stop} was 1000 in both cases. We also analyzed the models without early stopping, but did not find any relevant differences. We conclude that as in previous linear settings, overfitting apparently does not kick in very fast as it somehow does not downgrade the accuracy of the estimation to set a moderate but fixed m_{stop}.

Looking at the results from the $n = 150$ case in Table 6.4 for the first time in this simulation analysis, we can only report a very unsatisfying accuracy. The conditional coverage for none of the two algorithm lies in a reasonable range around the proposed 80%. The prediction intervals computed by boosting never cover more than around 73% of the test observations. The conditional quantile is always overestimated, as can be seen in Figure 6.7 **a)**.

But also the random forest approach fails to estimate intervals that hold the coverage. As in parts already observed in Section 6.1.3, the quantiles delivered by `quantregForest` do not change much for the different $\boldsymbol{x}$, the algorithm tends to compute intervals that apparently do not depend much on the predictor variables. As a result, for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ the coverage is much to high, as the corresponding conditional quantiles are underestimated. The opposite happens for the other $\boldsymbol{x}$, where the quantiles are overestimated and therefore the intervals cover less than 80% of the test data.

The reason for the unsatisfying results for the boosting approach can be clearly found in the imprecise estimation of the coefficients as can be seen in Figure 6.7 **d)**. For the high dimensional setups we omitted plotting the results for every component of $\hat{\boldsymbol{\beta}}_\tau$ and focused on components $\hat{\beta}_0 - \hat{\beta}_{30}$ in order to assure a clear appearance. The results of the left out $\hat{\beta}_{31} - \hat{\beta}_{200}$ were similar to the ones for $\hat{\beta}_{21} - \hat{\beta}_{30}$. We can clearly see a tendency to estimate the coefficients of predictor variables with no effect as 0, but the results are much less accurate and homogeneous than in setups in Section 6.1.3.

For the larger sample size of 1000 observations, results of the boosting approach do improve considerably (Table 6.4). The conditional coverage is still too high at the lower end of the X grid ($\boldsymbol{x}_1$) and too low for the upper end $\boldsymbol{x}_5$. Yet for the other $\boldsymbol{x}$ the results are reasonable, taking into account the complexity of the data situation.

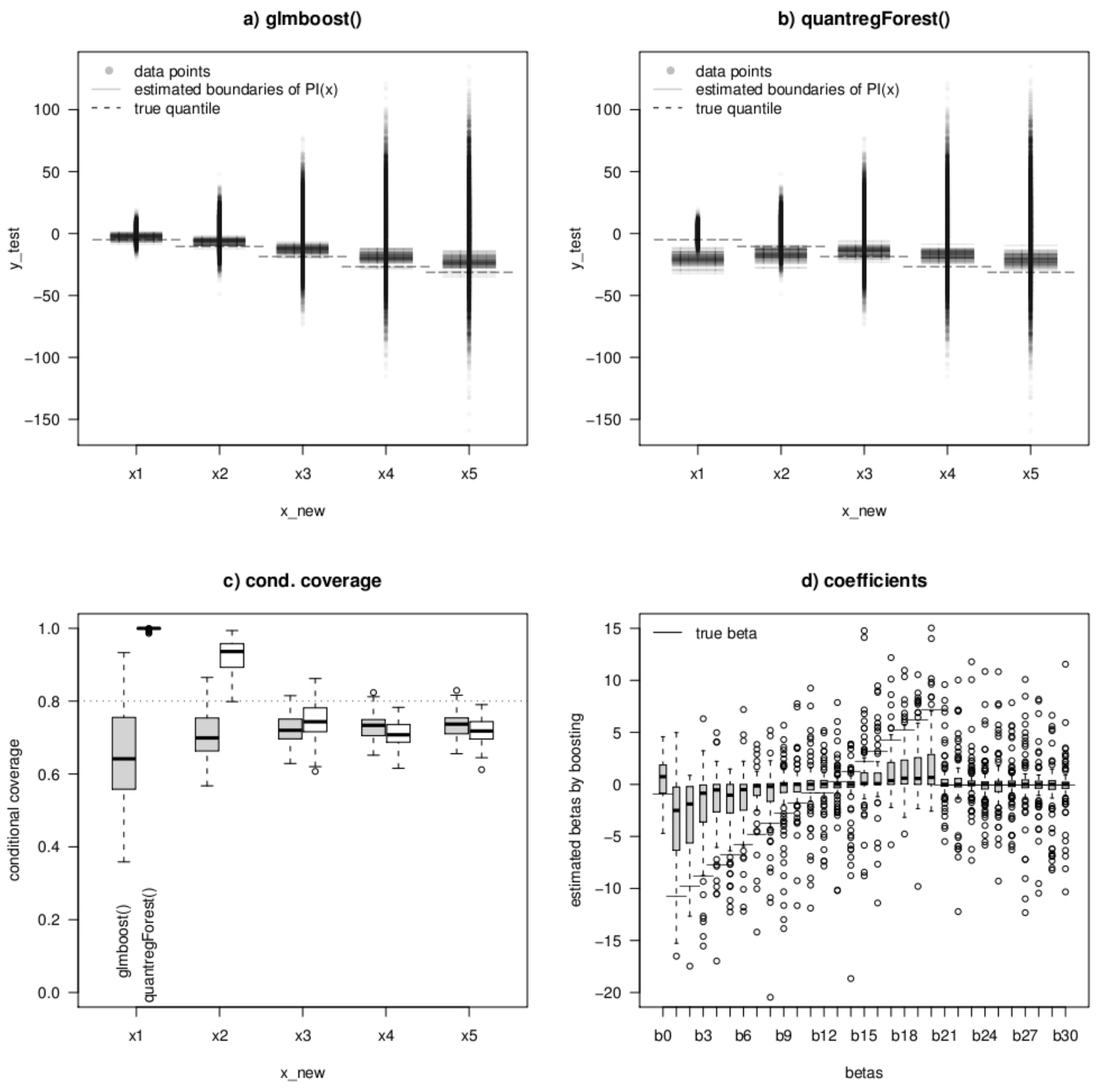

Figure 6.7: Results for the high dimensional data setup with $n = 150$ ($p > n$). In sector **d)** of the figure we omitted to include the boxplots of $\hat{\beta}_{31}$–$\hat{\beta}_{200}$ in order to reduce the graphical complexity

In the results of the random forest approach no relevant improvement can be seen for the higher sample size. The algorithm still fails to deliver a satisfying coverage for any $\boldsymbol{x}$ but $\boldsymbol{x}_3$. As can be seen in Figure 6.8, the estimated conditional quantiles seems not to depend on $\boldsymbol{x}$. For $\boldsymbol{x}_3$, which represents the expected mean of X in every component, this is not that dramatic as for more extreme observations of X.

The conditional quantiles resulting from the boosting approach suffer in some parts from the same problem, as also the values of $\hat{q}_{0.2}(\boldsymbol{x})$ are too small for $\boldsymbol{x}_1$ and too high for $\boldsymbol{x}_5$. But this effect is far less dramatic than for the random forest approach.

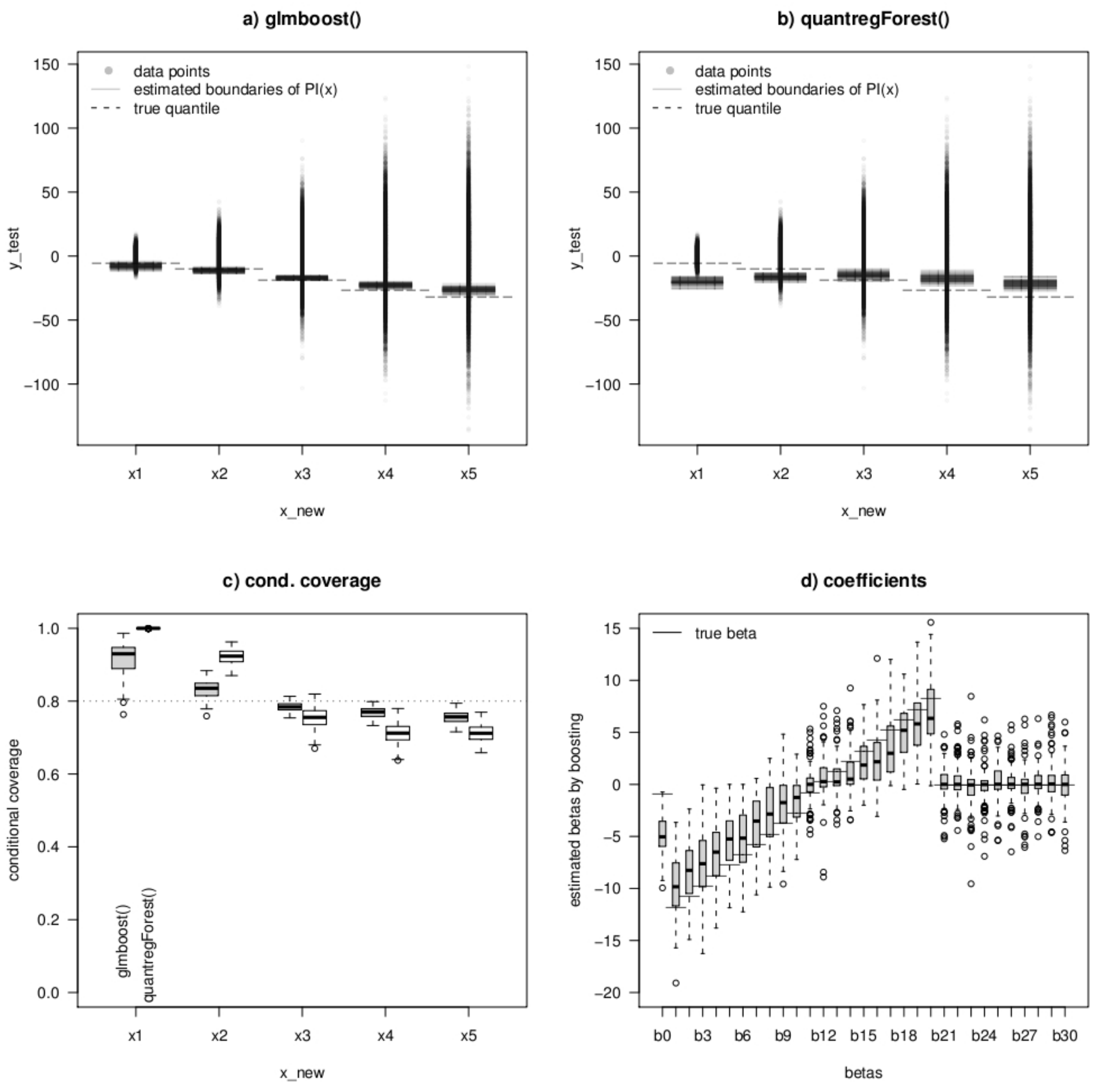

Figure 6.8: Results for the high dimensional data setup with $n = 1000$. In sector **d)** of the figure we omitted to include the boxplots of $\hat{\beta}_{31}$–$\hat{\beta}_{200}$ in order to reduce the graphical complexity.

The estimation of the coefficients (Figure 6.8 **d)**) improved significantly towards the $p > n$ case.

Yet for one component we do observe a remarkable bias: The intercept as already in Section 6.1.3 seems to cause problems for component-wise boosting. Its estimation nearly in every simulation step goes wrong, this happened for no other component. The correct intercept should be $\Phi^{-1}(0.2) = -0.8416$, yet in the mean it is estimated as -4.0530.

This *intercept-problem* cannot be resolved by selecting the feature `center=TRUE` inside the `glmboost()` function which forces the algorithm to mean center the predictor variables before fitting. We have not found a solution to this problem, yet we discovered that it does not appear in the same setup with the components of $\boldsymbol{x}$ independently drawn from $X \sim U(-0.5, 0.5)$ instead of $X \sim U(0, 1)$. At first glance this seems to be the same as mean centering the predictor variables. Yet the realizations of X also enter in $(\boldsymbol{x}_i^\top \boldsymbol{\alpha})\varepsilon_i$ and therefore have an effect on the heteroscedasticity of the error term and the variance of the response. Therefore we suspect those two factors to be of relevance in the challenge to explain when and why this problem appears. This view is supported by the fact that in earlier less complex setups with smaller dimensionality the estimation of the intercept was as accurate as for the other components (Section 6.1.1 and 6.1.2).

6.1.5 One predictor variable with nonlinear effect

Until now, we focused on testing the component-wise boosting algorithm for linear quantile regression (Section 5.3.1) available with the function `glmboost()` from package `mboost` (Hothorn et al., 2009) and compared it to results from the quantile regression forest approach as provided by the package `quantregForest` (Meinshausen, 2007).

We now pass on to the boosting algorithm for additive quantile regression with possible nonlinear effects (Section 5.3.2) available with the `mboost` function `gamboost()`, with the gradient family `QuantReg()` specified. We therefore simulate data with a predictor variable that has a nonlinear effect on the response:

$$Y_i = \beta_0 + f_1(x_i) + (\alpha_0 + g_1(x_i))\,\varepsilon_i \quad \text{where} \quad \varepsilon_i \overset{\text{iid}}{\sim} N(0,1)$$

x_i are observed realizations of random variable $X \sim U(0, 3)$.

For simulating data with a nonlinear effect of one predictor variable on the response including heteroscedasticity we used the same two setups as Fenske et al. (2009):

- 'sin' setup: $\beta_0 = 2$, $\alpha_0 = 0.5$, $f_1(x_i) = 3\sin(\frac{2}{3}x_i)$, $g_1(x_i) = 1.5(x_i - 1.5)^2$
- 'log' setup: $\beta_0 = 2$, $\alpha_0 = 0.7$, $f_1(x_i) = 1.5\log(x_i)$, $g_1(x_i) = \frac{1}{2}x_i$

The sample size for both setups was fixed as $n = 1000$. We constructed one-sided and two-sided 80% prediction intervals. As realizations of X to test the conditional coverage on, we selected $x_1 = 0.3$, $x_2 = 1.1$, $x_3 = 1.5$, $x_4 = 1.9$ and $x_5 = 2.7$.

For the boosting approach we again selected the optimum m_{stop} based on an optimization data set, while the step-length ν was fixed. In order to investigate again the relationship between the two parameters, we carried out the simulations for the one-sided intervals once with a fixed $\nu = 0.5$ and maximum $m_{\text{stop}} = 1000$ and once with a fixed $\nu = 0.5$ and a maximum $m_{\text{stop}} = 5000$ for the 'sin' setup. The resulting quantiles where nearly exactly the same, only the optimal m_{stop} in the mean raised from 162 ($\nu = 0.5$) to 763 ($\nu = 0.1$). This supports the view of the two main tuning parameters trading of each other with m_{stop} being inversely proportional to ν (Schmid and Hothorn, 2008). As the results did not differ concerning the conditional coverage we sticked to the faster version with the step-length raised to 0.5.

	'sin' setup		'log' setup	
	`gamboost`	`quantregForest`	`gamboost`	`quantregForest`
One-sided				
Coverage$_{x_1}$	**0.7808**	0.7780	**0.7882**	0.7561
Coverage$_{x_2}$	**0.8044**	0.7446	**0.8020**	0.7699
Coverage$_{x_3}$	**0.7879**	0.7586	**0.7951**	0.7557
Coverage$_{x_4}$	**0.7938**	0.7784	**0.7993**	0.7839
Coverage$_{x_5}$	**0.7937**	0.7835	**0.8000**	0.7783
Bias($\tau = 0.2$)	0.0480	**0.0332**	**0.0030**	0.0830
MSE($\tau = 0.2$)	**0.0307**	0.5613	**0.0335**	0.2674
Two-sided				
Coverage$_{x_1}$	**0.7838**	0.7173	**0.7871**	0.7098
Coverage$_{x_2}$	**0.7964**	0.7008	**0.7948**	0.7257
Coverage$_{x_3}$	**0.7832**	0.7426	**0.7973**	0.7409
Coverage$_{x_4}$	**0.7921**	0.7663	**0.7973**	0.7409
Coverage$_{x_5}$	**0.7807**	0.7382	**0.7958**	0.7545
Bias($\tau = 0.1$)	**0.0662**	0.0897	$\mathbf{-0.0021}$	0.0966
MSE($\tau = 0.1$)	**0.0450**	0.7180	**0.0421**	0.3852
Bias($\tau = 0.9$)	$\mathbf{-0.0371}$	-0.1181	$\mathbf{-0.0147}$	-0.1209
MSE($\tau = 0.0$)	**0.0396**	0.6663	**0.0375**	0.3020

Table 6.5: Results for the 'sin' as for the 'log' setups.

Results for the conditional coverage of the resulting intervals are presented in Table 6.5. For both setups both algorithm deliver reasonable results for the one-sides intervals, with the boosting intervals clearly being closer to the correct coverage of 80% of the test data. For the two-sided intervals only the boosting intervals succeed in yielding intervals that are at least very close to cover 80% of the test data given the specific values of $X = x$.

The graphical illustrations of the resulting intervals and their coverage in Figure 6.8 and Figure 6.9 was changed in sector **d)** compared to the linear setups, as this times we do not have any coefficients that were estimated. Instead we present for the two setups a simple scatter-plot to show the non-linearity of the effect of X on Y. We only plotted results for the two-sided intervals.

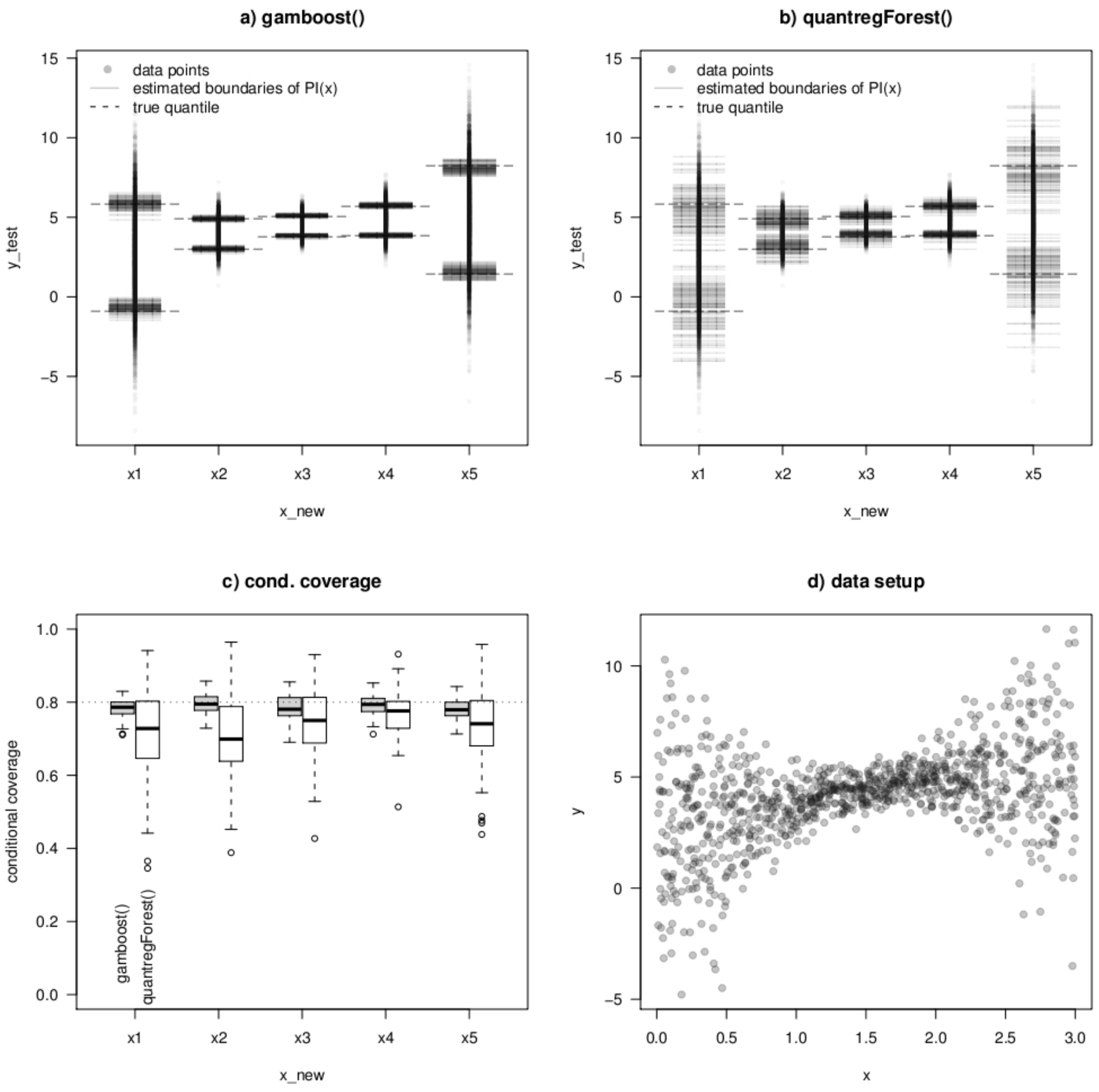

Figure 6.9: Results for the two-sided intervals in the 'sin' setup. As in this nonlinear setup no coefficients are estimated, section d) shows a scatterplot of a training data set to illustrate the shape of the effect.

The results of the 'sin' setup in Figure 6.9 show that the conditional quantiles are estimated quite precisely by `gamboost()` in almost every single simulation step. As a result also the conditional coverage does not differ much from the proposed 80%. The same does not hold true for the intervals provided by quantile regression forest. The estimation of the conditional quantiles especially for outer values of the X-grid is not very exact, results differ a lot in every simulation step.

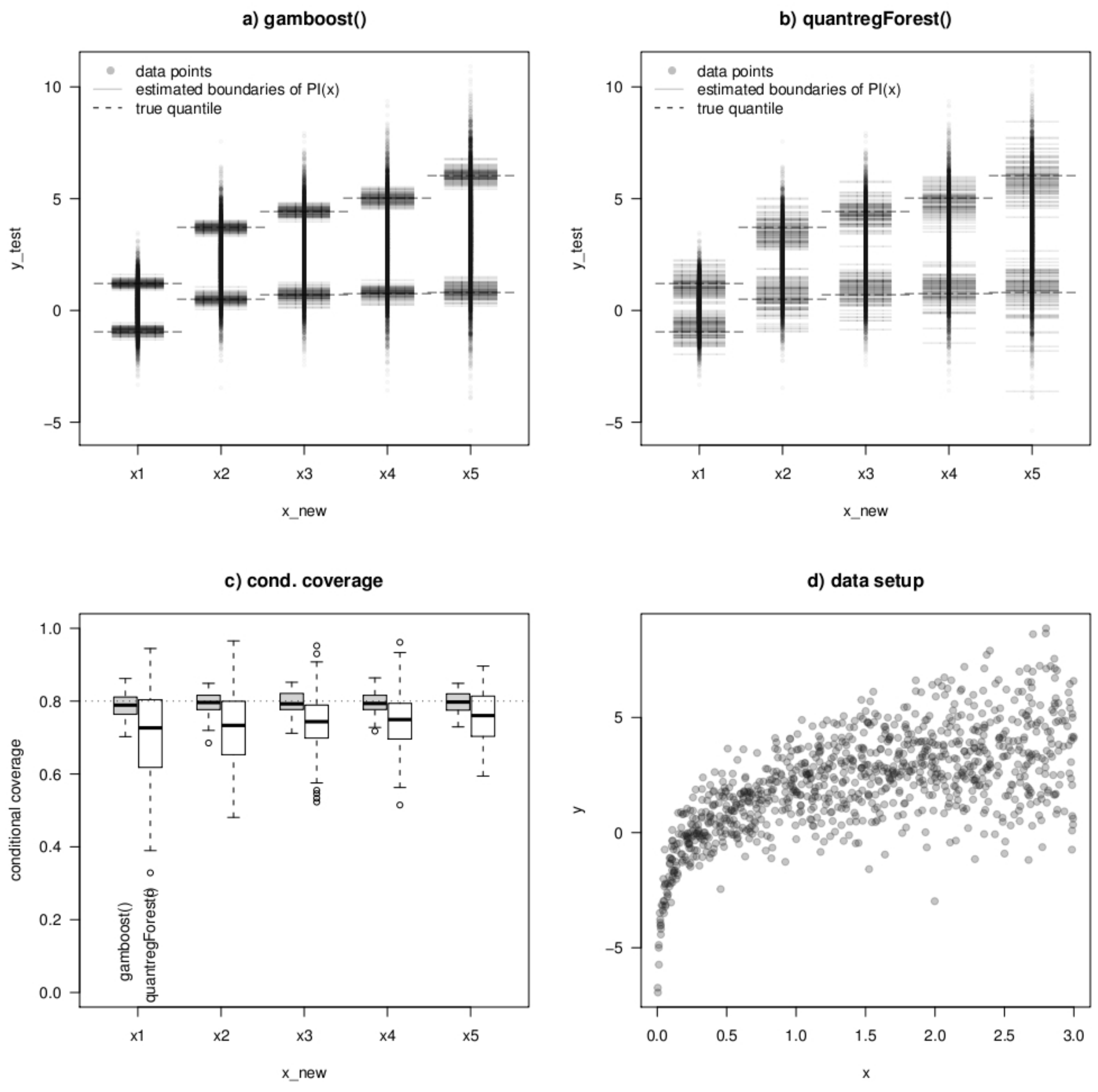

Figure 6.10: Results for the two-sided intervals in the 'log' setup. As for the 'sin' setup, section d) shows a scatterplot of a training data set to illustrate the shape of the effect.

For the 'log' setup results are illustrated in Figure 6.9. As for the 'sin' setup, results of `gamboost()` are highly accurate. The conditional quantiles are estimated correctly and therefore also the conditional coverage in every single point is very close to 80%. The results from `quantregForest()` are much less satisfying. Although the conditional quantiles are not estimated completely out of range, results from single estimation steps differ vastly. As a result the resulting intervals in some steps fail to hold a at least reasonable coverage, as sometimes the intervals fail to cover more than 50% of the test points.

6.1.6 Multiple predictor variables with additive linear and nonlinear effects

As a final setup for conditional coverage we step up further the complexity of the data situation by combining predictive variables with linear and nonlinear effects with variables without any effect (compare to Fenske et al. (2009)):

$$\begin{aligned} Y_i &= 2 + 3\sin(\frac{2}{3}x_{i1}) + \frac{3}{2}\log(x_{i2}) + 2x_{i3} - 2x_{i4} + 0 \cdot (x_{i5} + x_{i6}) \\ &+ \big(0.7 + 1.5(x_{i1} - 1.5)^2 + 0.5(x_{i2} + x_{i3})\big)\,\varepsilon_i \end{aligned}$$

With $\varepsilon_i \overset{\text{iid}}{\sim} N(0,1)$ and every component of $\boldsymbol{x}_i = (x_{i1}, ..., x_{i6})^\top$ being an observed realization of $X \sim U(0,3)$.

To test the conditional coverage of the resulting intervals we selected five data points, covering the X grid:

$$\begin{array}{ll} \boldsymbol{x}_1 = (\overbrace{0.3, ..., 0.3}^{6})^\top & \boldsymbol{x}_2 = (1.1, ..., 1.1)^\top \\ \boldsymbol{x}_3 = (1.5, ..., 1.5)^\top & \boldsymbol{x}_4 = (1.9, ..., 1.9)^\top \\ \boldsymbol{x}_5 = (2.7, ..., 2.7)^\top & \end{array}$$

In this setup we constructed one-sided 80% prediction intervals by modeling the conditional 0.2-quantiles.

For the boosting approach following the results of previous setups we set the step length $\nu = 0.5$ and optimized m_{stop}. In this setup we want to further investigate when finally overfitting will kick in and what effect it has on the conditional coverage of our intervals. We therefore carried out the simulation two times, once with a maximum m_{stop} of 2000 iterations and once with 10000.

For both settings we also optimized the parameter, which as expected yielded the same results. The mean m_{stop} selected were 617 iterations. Hence, a maximum of 2000 iterations was absolutely enough for this setup. Yet the models with $m_{\text{stop}} = 10000$ gives us the opportunity to analyze if and how overfitting affects the results of the prediction intervals.

Overfitting means that the estimation focuses too much on the training data, and therefore the trade-off between bias and variance is shifted towards a very low bias resulting in a high variance. Small changes in the data can affect the estimation strongly which reduces the predictive validity of the resulting models. Following the component-wise boosting approach, stopping the algorithm early enough should avoid overfitting. In our task to fit prediction intervals to prevent overfitting is crucial, as we are especially interested in using the modeled intervals not for the description of the training data, but for predictions of new observations.

Results for both estimation schemes and the different selections of m_{stop} are presented in Table 6.6. Results for the optimized m_{stop} are illustrated in Figure 6.11.

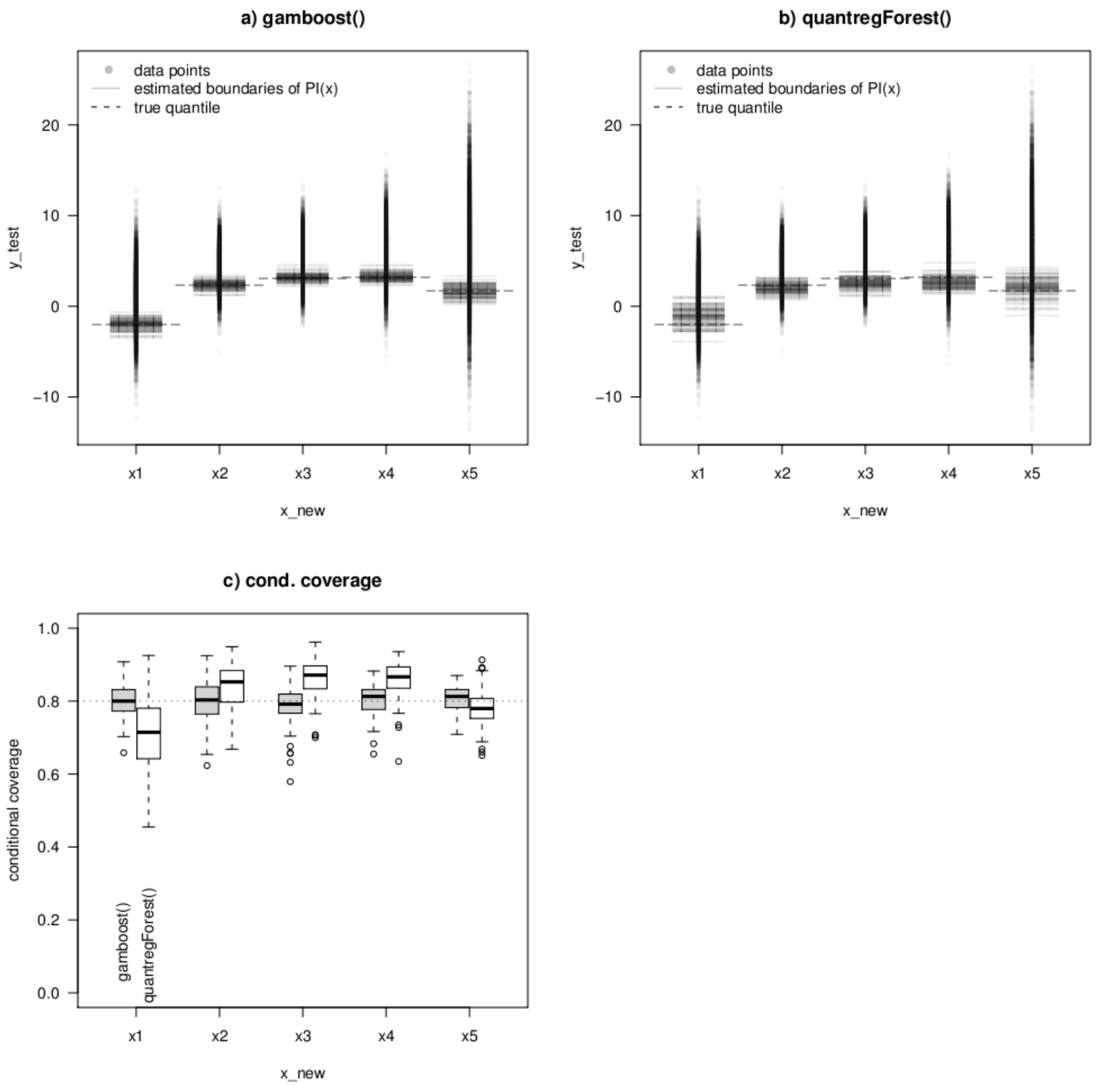

Figure 6.11: Results for one-sided 80% prediction intervals: Multiple predictor variables with additive linear and nonlinear effects. The illustrated results for the boosting algorithm refer to the models with optimized m_{stop}

The boosting approach delivers very accurate results. Those are best for intervals with optimized m_{stop}, for higher iterations the coverage slowly gets lower, yet with the fixed $m_{\text{stop}} = 10000$ still showing reasonable results. We can therefore conclude

	`gamboost`			`quantregForest`
	m_{stop} opt.	$m_{\text{stop}} = 2000$	$m_{\text{stop}} = 10000$	
Coverage$_{\boldsymbol{x}_1}$	**0.8000**	0.8019	0.7907	0.7082
Coverage$_{\boldsymbol{x}_2}$	**0.7978**	0.7941	0.7897	0.8355
Coverage$_{\boldsymbol{x}_3}$	**0.7895**	0.7856	0.7864	0.8619
Coverage$_{\boldsymbol{x}_4}$	**0.8034**	0.7880	0.7848	0.8570
Coverage$_{\boldsymbol{x}_5}$	0.8072	**0.7990**	0.7961	0.7796
Bias($\tau = 0.2$)	-0.0374	$\mathbf{-0.0003}$	0.0236	-0.0877
MSE($\tau = 0.2$)	**0.2471**	0.4983	0.6913	0.9176

Table 6.6: Results comparing 80% one-sided prediction intervals computed by boosting with different m_{stop} and quantile regression forests.

that our results confirm that early stopping improves the accuracy of predictions.

Yet it gets also clear that we deal with a rather slow overfitting behavior (Bühlmann and Hothorn, 2007). Optimizing the tuning parameter on an additional data frame in every simulation step yielded 617 iterations. If we keep on iterating the boosting algorithm more than 15 times as long as proposed, we still have very reasonable results. The intervals estimated with a fixed $m_{\text{stop}} = 10000$ still even leave behind prediction intervals estimated quantile regression forests, concerning their conditional coverage.

6.2 Sample coverage

As we have already extensively analyzed the conditional coverage for nonparametric prediction intervals based on quantile regression estimated by boosting and quantile regression forests, we will now pass on to show some selected setups to investigate the sample coverage by using the heuristic sample interpretation.

This interpretation follows the introduction of quantile regression forests by Meinshausen (2006), who noted that his algorithm could be used to compute prediction intervals and confirmed this idea in some machine-learning benchmark data sets. He constructed intervals $\text{PI}_{(1-\alpha)}(\boldsymbol{x})$ around each observation of the data sets by using cross-validation and then checked if more or less $(1-\alpha) \cdot 100\%$ of the observed response values in fact lie inside the proposed borders given by conditional quantiles.

He therefore used the heuristic sample interpretation, by assuming that the correct specified interval should cover $(1-\alpha) \cdot 100\%$ observations of a new sample. We call this view heuristic, as it in fact is not incorrect and can be very helpful in practical situations. We will also use it in the real world example in Chapter 7.

It is true that we can expect from a correct specified $\text{PI}_{(1-\alpha)}(\boldsymbol{x})$ also to cover $(1-\alpha)\cdot 100\%$ of a new sample of $Y|X=\boldsymbol{x}$.

$$\begin{aligned} \mathbb{E}\left(Y \in \text{PI}(\boldsymbol{x}_{\text{new}})|X=\boldsymbol{x}_{\text{new}}\right) &= 1-\alpha \quad \forall \boldsymbol{x}_{\text{new}} \\ \Rightarrow \quad \mathbb{E}(Y \in \text{PI}(\boldsymbol{x})) &= 1-\alpha \end{aligned}$$

Yet we also showed that this sample coverage is not enough to confirm that these intervals work for any realization of X by giving an example of an interval that in fact holds the sample coverage, but is not correctly specified for nearly every realization of X: The prediction intervals based on unconditional quantiles $Q_\tau(y)$ of a training data will cover in many cases the correct proportion of observations in a test data, while nearly never fulfilling the conditional interpretation. Therefore we have shown that the backward conclusion is not possible:

$$\begin{aligned} \mathbb{E}(Y \in \text{PI}(\boldsymbol{x}_{\text{new}})) &= 1-\alpha \\ \not\Rightarrow \quad \mathbb{E}\left(Y \in \text{PI}(\boldsymbol{x}_{\text{new}})|X=\boldsymbol{x}_{\text{new}}\right) &= 1-\alpha \quad \forall x_{\text{new}} \end{aligned}$$

We will focus on the same data situations as in Section 6.1, yet the simulation design for confirming the sample coverage is quite different from the one to confirm the conditional coverage, as it has been used in the earlier setups:

1. **Training data:**
   - Simulate 1000 realizations of X and $Y|X=\boldsymbol{x}$: $(y_j^*, \boldsymbol{x}_j^*)$ for $j=1,...,1000$
   - Estimate the conditional quantiles $\hat{Q}_\tau(Y|X=\boldsymbol{x}^*)=\hat{q}_\tau(\boldsymbol{x}^*)$ on the training data
2. **Test data:**
   - Simulate a test data set consisting of 1000 observations $(y_i, \boldsymbol{x}_i)$ for $i=1,...,1000$
   - Use the conditional quantiles from the training data to construct prediction intervals for every component of $\boldsymbol{y}$ by plugging in the observations of the predictor variables: $\text{PI}(\boldsymbol{x}_i)$ for $i=1,...,1000$
   - Evaluate the prediction intervals on $\boldsymbol{y}$ by

$$\text{Sample-coverage} = \frac{\sum_{i=1}^{1000} I\{y_i \in \text{PI}(\boldsymbol{x}_i)\}}{1000}$$

3. **Iteration:**
   - Repeat steps **1.** and **2.** 100 times

Unlike in the conditional case, results therefore will be just one number - the sample coverage in every simulation step.

We will analyze different linear and nonlinear setups already introduced in Section 6.1. In this section we will not further concentrate on the impact of tuning parameters as we will always choose the same methods as already presented in the conditional case. For the boosting approach we therefore will simulate in every simulation step a third data set to optimize m_{stop} on.

We will focus on two-sided 80% prediction intervals and will therefore estimate the conditional 0.1-quantile as well as the 0.9-quantile:

$$\text{PI}(\boldsymbol{x}_{\text{test}}) = [\hat{q}_{0.1}(\boldsymbol{x}_{\text{test}}), \hat{q}_{0.9}(\boldsymbol{x}_{\text{test}})]$$

As in the conditional case, in every simulation step we will estimate intervals for the corresponding test data set once with the component-wise boosting approach and once by using quantile regression forests. Additionally we will also estimate 'naive' prediction intervals that do not take into account the information of the prediction variables, to further illustrate the reason why we prefer the conditional interpretation:

$$\text{'naive' PI} = \left[\hat{Q}_{0.1}(\boldsymbol{y}^*), \hat{Q}_{0.9}(\boldsymbol{y}^*)\right]$$

These 'naive' quantile intervals just use the unconditional empirical quantiles of the response in the training data set.

6.2.1 Linear setups

We will present results of some selected linear setups that were already introduced in Section 6.1. The underlying model for every setup includes heteroscedasticity and a gaussian distributed error term:

$$Y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta} + (\boldsymbol{x}_i^\top \boldsymbol{\alpha})\varepsilon_i \quad \text{with} \quad \varepsilon_i \overset{\text{iid}}{\sim} N(0,1)$$

The components of $\boldsymbol{x}_i$ are observed realizations of $X \sim U(0,1)$.

We will analyze results of following three data situations:

- **Setup 1** One predictor variable:
  $\boldsymbol{\beta} = (1,2)^\top, \quad \boldsymbol{\alpha} = (2,5)^\top$
- **Setup 2** Multiple predictor variables:
  $\boldsymbol{\beta} = (1,2,3,4)^\top, \quad \boldsymbol{\alpha} = (1,1,2,1)^\top$
- **Setup 3** High-dimensional data:
  $\boldsymbol{\beta} = (0,-10,-9,-8,...,8,9,10,\overbrace{0,...,0}^{180})^\top$, $\boldsymbol{\alpha} = (1,\overbrace{2,...,2}^{20},\overbrace{0,...,0}^{180})^\top$

Results concerning the sample coverage of prediction intervals constructed by `glmboost()`, `quantregForest()` as well as the unconditional 'naive' intervals are presented in Table 6.7.

	`glmboost`	`quantregForest`	'naive' intv.
Setup 1			
min. sample coverage	**0.7590**	0.6650	0.7550
mean sample coverage	**0.7993**	0.7113	0.7981
max. sample coverage	0.8390	0.7740	**0.8320**
Setup 2			
min. sample coverage	**0.7390**	0.6030	0.7310
mean sample coverage	0.7953	0.6776	**0.8005**
max. sample coverage	0.8420	0.7320	**0.8390**
Setup 3			
min. sample coverage	0.6740	0.7300	**0.7540**
mean sample coverage	0.7191	0.7763	**0.7980**
max. sample coverage	0.7710	**0.8090**	0.8340

Table 6.7: Results for selected linear data setups concerning sample coverage. The values of the best performing algorithm in each row is printed bold.

For the low dimensional setups intervals estimated by `glmboost()` can hold the proposed coverage of 80% over the whole sample. The random forest approach has some problems for linear effects with few predictors. This was already observed in simulations for the conditional coverage. The best performing algorithm concerning sample coverage seems to be the 'naive' intervals.

Of course, we do not propose to use the 'naive' intervals as prediction intervals in practice. We just want to show how misleading focusing only on sample coverage can be.

In Figure 6.12 we further illustrate results from Setup 2. Figures in all four sectors are structured as follows:

- The response vector $\boldsymbol{y}$ of the test data is plotted against the sample index (1-1000) ordered by the size of the response (bold gray points).
- The other symbols represent the estimated conditional quantiles and therefore the boundaries of the prediction intervals $\mathrm{PI}_{0.8}(\boldsymbol{x}_i)$. They are plotted in the same order as the response. They are circles ($\circ$), if they cover the observed y_i of the test data and crosses ($+$), if they fail to do so.
- Note that the Figures therefore only represent one simulation step.

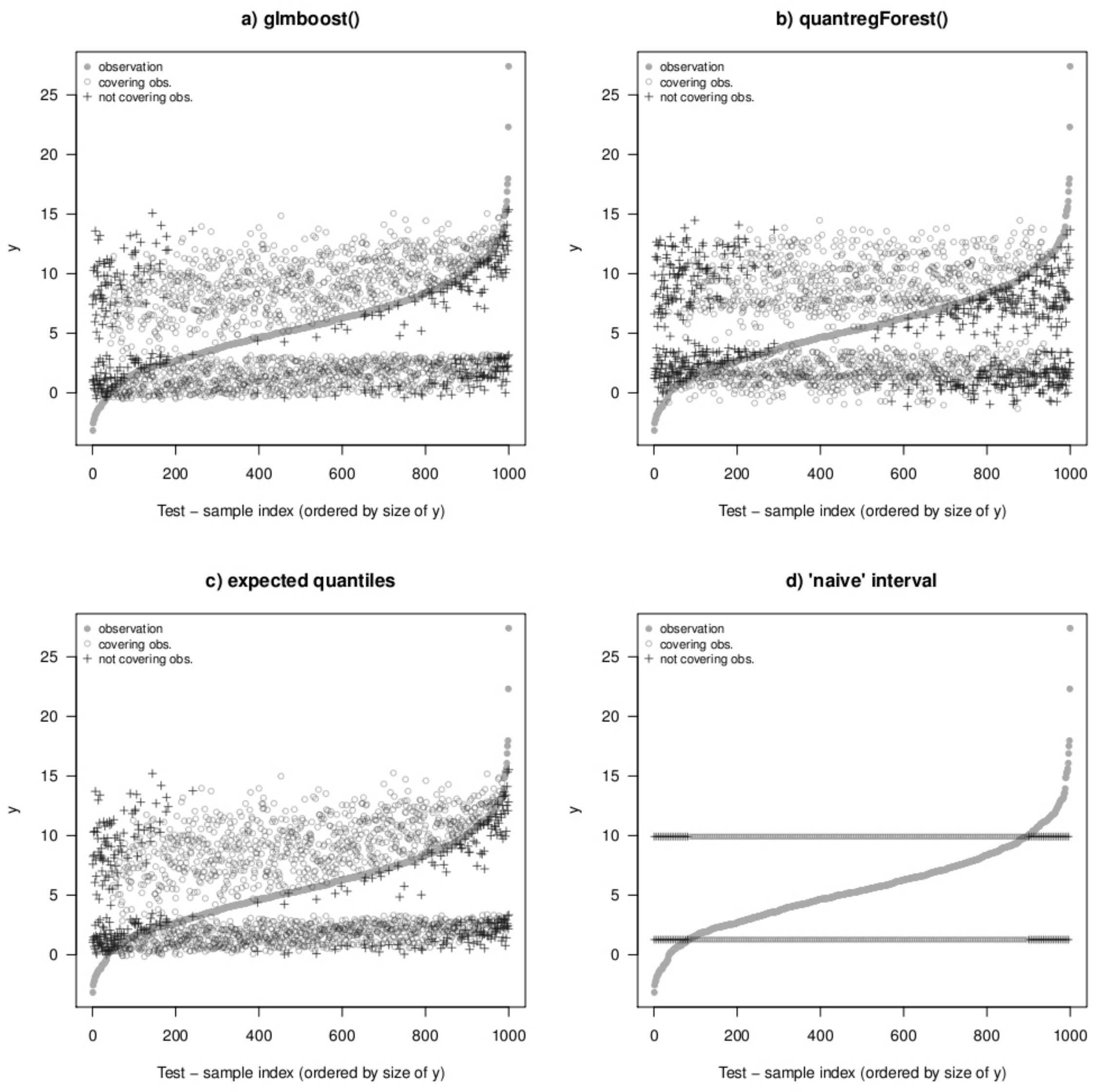

Figure 6.12: Results for Setup 2. Figures in sectors represent only one simulation step.

The intervals in sector **c)** are constructed using the expected conditional quantiles by applying the true $\boldsymbol{\beta}_\tau$ to the predictor variables, which already takes into account the heteroscedasticity of the error term:

$$\begin{aligned} Q_\tau(Y|X = \boldsymbol{x}_i) &= \boldsymbol{x}_i^\top \boldsymbol{\beta} + (\boldsymbol{x}_i^\top \boldsymbol{\alpha})\Phi^{-1}(\tau) \\ &= \boldsymbol{x}_i^\top \boldsymbol{\beta}_\tau \\ \text{with} \quad \boldsymbol{\beta}_\tau &= \boldsymbol{\beta} + \boldsymbol{\alpha}\Phi^{-1}(\tau) \end{aligned}$$

For Setup 2 we observe that `glmboost()` as well as `quantregForest` do have problems covering the extreme observations of Y. Those are the data points on the left or the right side of the plots. But also using the expected conditional quantiles in Figure 6.13 **c)** leads to similar results.

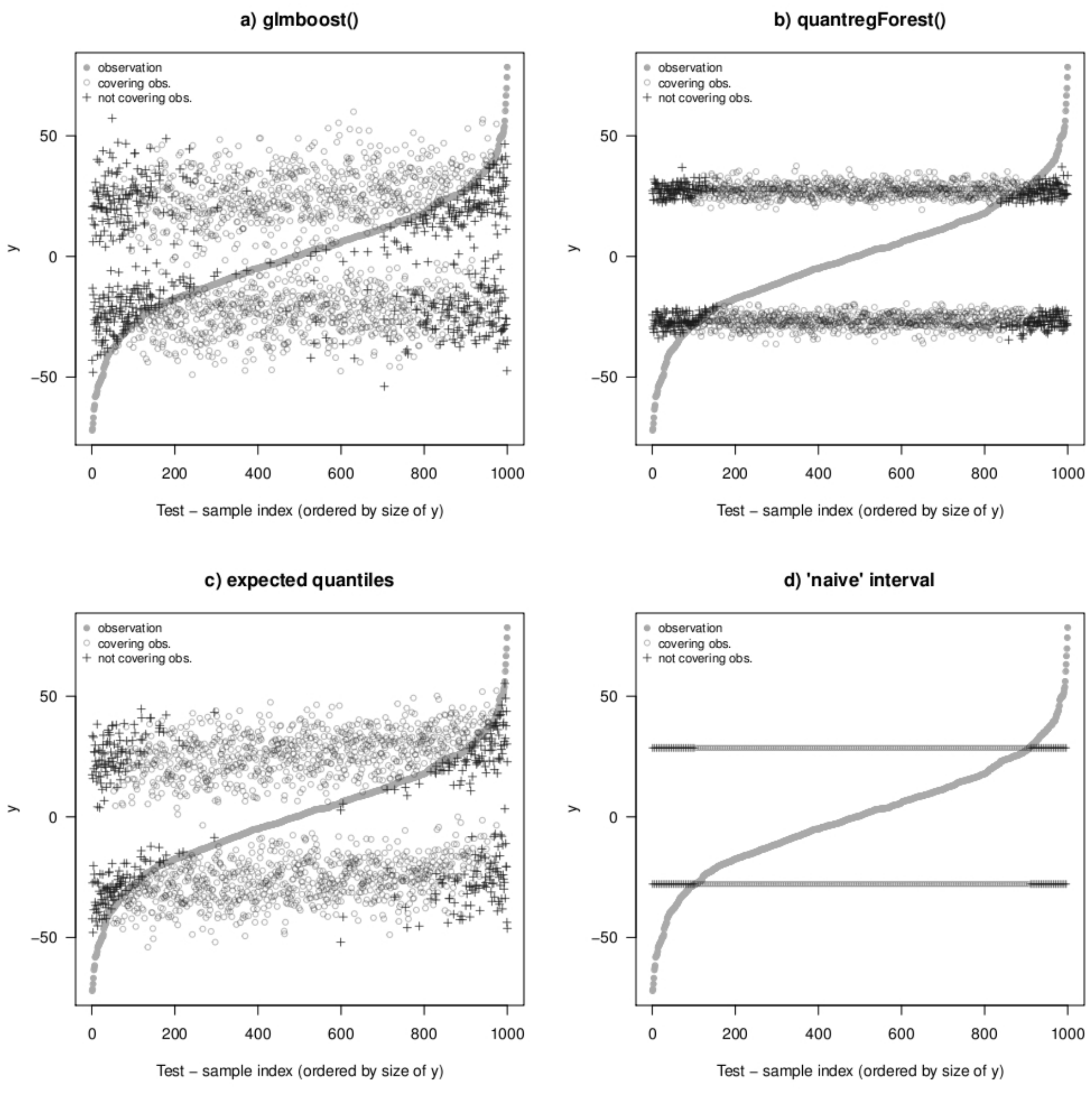

Figure 6.13: Results for Setup 3. Figures represent only one simulation step.

In the complex high dimensional data situation in Setup 3 we observe an important aspect about the intervals estimated by `quantregForest()`. Unlike the boosting intervals they succeed in holding the coverage over the whole sample. Yet they achieve this by returning similar values than the 'naive' intervals. In this high dimensional setting with many predictors without effect on the response, `quantregForest()` tends to estimate the conditional quantiles simply as the empirical sample quantiles. This secures the sample coverage as can be seen in the simulation results, but is far away from the true expected conditional quantiles presented in sector **d)**.

The intervals estimated by `glmboost()` fail to hold a coverage of 80% over the whole sample in this special setting. Yet the provided intervals are much more similar to the intervals constructed by using the true coefficients $\boldsymbol{\beta}_\tau$.

This setups therefore underline the importance of looking also at the conditional coverage, as focusing on the sample coverage in this case is clearly misleading. The random forest intervals fail to adopt to the realizations of X, yet show much better results concerning the sample coverage than the boosting approach.

6.2.2 Nonlinear setups

As for the linear case, we will also present results from simulation studies from three different setups. Every setup includes heteroscedastic error terms and of course predictors with nonlinear effects (compare to Fenske et al. (2009)).

- **Setup 1** One predictor 'sin' effect:

$$Y_i = 2 + 3\sin(1.5x_i) + \left(0.5 + 1.5(x_i - 1.5)^2\right)\varepsilon_i$$

- **Setup 2** One predictor 'log' effect:

$$Y_i = 2 + 1.5\log(x_i) + (0.7 + 0.5x_i)\,\varepsilon_i$$

- **Setup 3** Multiple predictors additive linear and nonlinear effects:

$$\begin{aligned} Y_i &= 2 + 3\sin(1.5x_{i1}) + 1.5\log(x_{i2}) + 2x_{i3} - 2x_{i4} + 0\cdot(x_{i5} + x_{i6}) \\ &+ \left(0.7 + 1.5(x_{i1} - 1.5)^2 + 0.5(x_{i2} + x_{i3})\right)\varepsilon_i \end{aligned}$$

In every setup ε_i is gaussian distributed $\varepsilon_i \overset{\text{iid}}{\sim} N(0,1)$ and the components of $\boldsymbol{x}_i$ are realizations of $X \sim U(0,3)$.

Results concerning the sample coverage of prediction intervals constructed by `gamboost()`, `quantregForest()` as well as the unconditional 'naive' intervals are presented in Table 6.8.

Once again the intervals estimated by component-wise boosting show very reasonable results for all three setups. The intervals based on random forest do have problems with setups consisting of only one predictor variable. They do improve significantly in Setup 3 which includes 6 predictor variables, compared to the one-dimensional predictor in the first two setups.

Yet the algorithm that gets closest to covering 80% of the test data set in every setup is the 'naive' interval absolutely ignoring the effect of any predictor variable.

	`gamboost`	`quantregForest`	'naive' intv.
Setup 1			
min. sample coverage	0.7400	0.6680	**0.7550**
mean sample coverage	0.7890	0.7264	**0.7964**
max. sample coverage	**0.8230**	0.7800	0.8350
Setup 2			
min. sample coverage	0.7490	0.6810	**0.7530**
mean sample coverage	0.7927	0.7274	**0.7981**
max. sample coverage	0.8360	0.7760	**0.8360**
Setup 3			
min. sample coverage	0.7400	0.7500	**0.7530**
mean sample coverage	0.7833	0.7980	**0.7985**
max. sample coverage	**0.8250**	0.8360	0.8410

Table 6.8: Results for the three nonlinear data setups concerning sample coverage. The values of the best performing algorithm in each row is printed bold.

These intervals clearly can hold the sample coverage. Yet for any fixed $\boldsymbol{x}_{\text{new}}$ as for the conditional interpretation they would fail, as for some $\boldsymbol{x}_{\text{new}}$ they would cover every realization of Y and for others none.

Results for Setup 3 are presented in Figure 6.14. The expected quantiles for sector **d)** are computed using $\Phi^{-1}(\tau)$ as the inverse of the gaussian distribution function:

$$\begin{aligned} Q_\tau(Y|X=\boldsymbol{x}_i) &= 2 + 3\sin(1.5x_{i1}) + 1.5\log(x_{i2}) + 2x_{i3} - 2x_{i4} + 0\cdot(x_{i5}+x_{i6}) \\ &+ \quad \Phi^{-1}(\tau)\cdot\big(0.7 + 1.5(x_{i1}-1.5)^2 + 0.5(x_{i2}+x_{i3})\big) \end{aligned}$$

In Setup 3 boosting as well as random forest intervals show very good results, similar to the expected quantiles in sector **d)**.

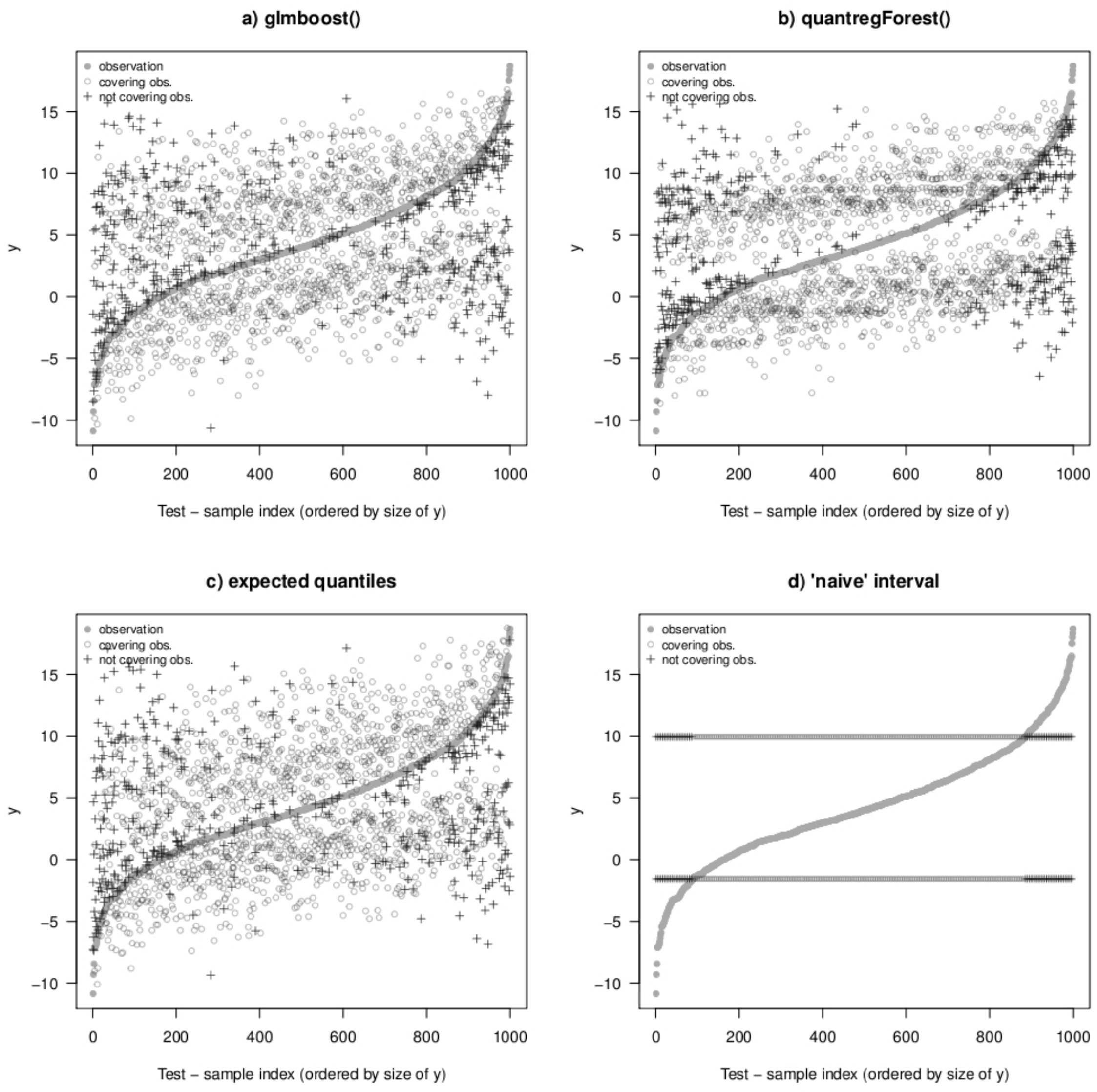

Figure 6.14: Results for Setup 3. Figures represent only one simulation step.

6.3 Conclusions from the simulation studies

Let us recall the two major questions from the beginning of this chapter:

1. How do prediction intervals estimated by quantile regression forests and by boosting perform in different setups concerning the coverage and the correct estimation of the conditional quantiles?
   - Linear effects of one or more predictor variables including high-dimensional situations with more predictors than observation ($p > n$).
   - Additive potential nonlinear effects of one or more predictor variables.
2. How do the two major tuning parameters of boosting affect the performance of prediction intervals?

Concerning the first question, we analyzed the performance once based on the conditional coverage and once based on the sample coverage. As already pointed out in Chapter 2, we believe that the conditional interpretation of prediction intervals is more appropriate. This view is also supported by the results of Section 6.2, where the sample coverage failed to show severe problems in the estimation of the conditional quantiles by the random forest approach in the high dimensional linear setup. Furthermore, 'naive' intervals that only refer on the unconditional sample quantiles of the training data were the best performing algorithm in every single setup - if one considers only the sample coverage.

Hence, one first result of these simulation studies is the confirmation that the conditional coverage is the one to pay attention for, when it comes to confirm the correct coverage of prediction intervals.

When we come to the question itself, the answer varies depending on the data situation. No algorithm showed absolute satisfying results in every setup. Yet we can sum up the results in a few points:

- `glmboost()` and `gamboost()` show excellent results for linear and nonlinear effects of univariate or low dimensional (p<10) predictor variables. The conditional quantiles are estimated correctly, and therefore the conditional coverage of the resulting intervals is fulfilled.
- `quantregForest()` due to its design has some problems when it comes to setups with only one predictor variable, no matter if the effect is linear or nonlinear. Results improve with additional predictors.
- When it comes to multiple predictors (p>10) with included variable selection, `glmboost()` tends to select the correct variables. Nevertheless, the resulting intervals have problems to hold the conditional coverage for values at the border of the X space.

- For high dimensional settings with $p > n$, both algorithm fail to hold the conditional coverage. The results of intervals estimated by `glmboost()` improve considerably when the sample size is raised to $n > p$.

- `quantregForest()` tends to return the unconditional sample quantiles for high dimensional settings instead of conditional quantiles. This serves for the sample coverage, but the intervals cannot hold the conditional coverage. This does not much improve for a raised sample-size.

- `glmboost()` showed in two high dimensional setups with severe heteroscedasticity an unprecedented bias in the estimation of the intercept. We did not find a solution to this problem, yet suspect that the amount of heteroscedasticity of the error term and the variance of the response have something to do with the appearance of this bias.

We have to point out that our simulation studies bear some limitations: We only focused on settings where standard parametric tools to construct prediction intervals are not feasible. Therefore every model included at least heteroscedastic error terms. Furthermore, we had no setting with interactions between predictor variables. This could have been much more favorable for quantile regression forests. We also focused mainly on settings with a reasonable high sample size of 1000 observations.

Concerning the second question of interest our results indicate that for applying the boosting approach to quantile regression one can safely fix the step-length $\nu = 0.5$ and optimize m_{stop}. Compared to setting $\nu = 0.1$, as it is the default value in `mboost`, the algorithm delivers comparable results while needing only about 20% of the otherwise necessary boosting iterations.

When it comes to selecting m_{stop}, we were somehow surprised that we found for most of the settings no relevant impact on the conditional coverage of the resulting intervals, when we just used a fixed m_{stop} instead of applying time consuming measures to optimize this tuning parameter on an additional data set. Yet we do have to state that we did not try unreasonable high fixed m_{stop}, but values that were not far from the optimized parameter in most of the settings. Only in the additive nonlinear setup (Section 6.2.2) we noticed a relevantly reduced coverage for larger m_{stop}. Therefore we can not conclude that overfitting is no problem. Yet our findings indicate that overfitting at least kicks in very late for most of the setups used in this simulation studies.

Chapter 7

Prediction inference in practice

This chapter was deleted due to data privacy.

Chapter 8

Conclusion

At the beginning of this thesis, we quoted Niels Bohr, the winner of the Nobel Prize in Physics in 1922:

Prediction is very difficult, especially about the future.

It is a famous quote, and a common saying - we all have heard it several times before. Now, at the end of a thesis investigating prediction inference with ensemble methods, what do we answer if we hear it again?

Of course Bohr is right: It is easier to analyze data from a sample in order to describe the underlying structure instead of making predictions about future observations. Not because predictions make problems in their computation or are problematic from a methodological point of view. But simply because of the fact that predictions bear a sincere problem that cannot be solved by any statistical method: They sooner or later will be confronted by reality.

When we talk about the accuracy and the performance of prediction intervals we should keep that in mind: Other statistical tools never face this confrontation. Estimation methods may have a variance and a bias that trade off each other and help us to judge their precision. A prediction interval will be judged by a simple dichotomous variable: Does it cover the new observation or not?

We did ask this question over and over in simulation studies and in the practical example for the movie data. We focused on quantile regression with boosting with different base learners and on quantile regression forests.

When we started to work on this project, we soon stepped into a common pitfall concerning prediction intervals: We tried to see them as confidence intervals, just for predictions and not for parameters. As we have learned by now, this view may lead to false conclusions. We are used to judge an estimation procedure by considering the size of an confidence interval. Prediction intervals do not cover parameters but random variables. The size of a prediction interval tells us something about the variance of the observation which is not explained by the predictors. So of course, the size of the interval gives us an indication of the accuracy of a point-prediction. However, we cannot expect the interval to become arbitrarily small. As long as the future observation we want to cover is a random variable, this randomness controls the size of the interval.

The second pitfall we stepped in at the beginning is also the first important finding of this thesis: The adequate interpretation of the coverage for prediction intervals does not refer to a whole new sample. The more correct interpretation is the conditional one, conditioned on a fixed realization of X. Correct specified intervals should hold this conditional coverage for every possible realization of X, and therefore also over a whole sample. Nevertheless, as we showed, looking only at the sample coverage can be very misleading and should therefore be avoided. The sample coverage can give an indication if the intervals could be accurate. But it does not prove in any form that they hold the proposed coverage for every realization of X.

In simulation studies we therefore took a look on the conditional coverage for different limited data points, in different setups. We did only focused on non-standard situations, as we were interested in setups where parametric intervals based on the standard linear model do not work. However, as the boosting approach with OLS base learners eventually leads to the same results as standard regression we can conclude that they also work in standard situations, where one could also have used parametric intervals.

We were especially interested in the results of the boosting approach, and therefore selected setups that seemed to give the algorithm a reasonable chance to get the intervals right. As a result, in most cases we chose setups with 1000 observations in the training data and setups that did not include interactions between predictor variables. Knowing about this limitations, we can state that the results concerning the conditional coverage for the most relevant setups are more than promising for both considered ensemble methods.

Boosting with OLS and P-splines as base learners showed excellent results for setups with one to a handful predictor variables. Regression forest by their design have problems with setups including only one predictor but results improve with additional predictor variables. For high dimensional setups, both algorithms failed to hold the correct coverage in the $p > n$ case. Yet, we also have to state that we used a rather unfavorable setup with high variance and heteroscedasticity in every predictor. Considering this, it is still notable that at least the intervals of the boosting approach cover around 70 % of the new observations conditioned on different X, and results improved considerably when the sample size was raised.

We found that the quantile regression approach in high dimensional setups with large amounts of variables without effect on the response tends to return the sample quantiles rather than correct specified conditional quantiles. That might be an indication for the algorithm having problems to identify the variables of relevance, and therefore for a new X all realizations of the original training data get the same weights. This does not affect the sample coverage but the conditional one. The reasons for the appearance of this effect should be further investigated.

For the boosting approach, we found in similar setups a seemingly structural bias in the estimation of the intercept. This may seem like a minor problem, nevertheless in our view it could prevent the intervals of showing better results especially at the borders of the X space in high dimensional linear settings. The reasons for the appearance of this problem and possible solutions should be further investigated.

Concerning the tuning parameters for the boosting approach, we concluded that the step length ν can be fixed higher as it is the default for common L_2 boosting. The reason for that are the small gradients with values τ or $(1-\tau)$ resulting from the *check-function*. The selection of a reasonable value for the second tuning parameter m_{stop} is crucial, although in our setups we only found rather slow overfitting behavior.

In the practical example predicting the movie attendance in the first week boosting with different base learners as well as quantile regression forests showed quite satisfying results. Given the complexity of the data situation, the delivered prediction intervals seem to hold the coverage at least for movies with not too extreme outcome. When we also gave a robust point estimator with the conditional median, quantile regression forests showed their big advantage in preventing quantile crossing, which in this special case can be seen as a very favorable property of this algorithm. If the data providers might be interested not only in the intervals itself but also in the interpretation of the effect their prediction variables have on the attendance of a new movie, this could be the point that makes boosting the algorithm of choice.

So, what do we do answer to Niels Bohr?

Of course it is difficult to make predictions concerning the future. But especially with today's computational power, there are ways to construct reliable intervals for future observations based on a combination of multiple predictions. The resulting intervals work well in many practical situations. They hold the proposed coverage, are very flexible and do not need any assumptions about underlying distribution functions. Of course we found setups where the results were not as satisfying as we would have wished. Also, there are aspects which we still cannot fully explain, as the problems with the intercept. Finally there is still work to be done which gives us the chance to conclude: We are on a good way. But as usual, further research is warranted.

Bibliography

Bühlmann, P. and T. Hothorn (2007). Boosting algorithms: regularization, prediction and model fitting. *Journal of Statistical Science 22(4)*, 477–505.

Bühlmann, P. and B. Yu (2003). Boosting with the l_2 loss: Regression and classification. *Journal of the American Statistical Association 98*, 324–339.

Breiman, L. (1996). Bagging predictors. *Machine Learning 24*, 123–140.

Breiman, L. (1998). Arcing classifiers. *The Annals of Statistics 26*(3), 801–824.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation 11*(7), 1493–1517.

Breiman, L. (2001). Random forests. *Machine Learning 45*, 5–32.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone (1984). *Classification and Regression Tress.* Boca Ration, Florida: Chapman and Hall/CRC.

Cieczynski, S. (2009). *Bayesianische Quantilregression.* Diploma-thesis, Ludwig-Maximilians-Universität München, Institut für Statistik.

Dietterich, T. G. (2000). Ensemble methods in machine learning. *Multiple Classifier Systems 1857-2000*, 1–15.

Efron, B., T. Hastie, L. Johnstone, and R. Tibshirani (2002). Least angle regression. *Annals of Statistics 32*, 407–499.

Eilers, P. and B. Marx (1996). Flexible smoothing with b-splines and penalties. *Journal for the History of Astronomy 2.*

Everitt, B. S. and T. Hothorn (2006). *A Handbook of Statistical Analyses Using R.* Boca Ration, Florida: Chapman and Hall/CRC.

Fahrmeir, L., T. Kneib, and S. Lang (2007). *Regression: Modelle, Methoden und Anwendungen.* Berlin: Springer Verlag.

Fahrmeir, L., R. Künstler, I. Pigeot, and G.Tutz (1996). *Statistik: Der Weg zur Datenanalyse.* Berlin: Springer Verlag.

Fenske, N. (2008). *Flexible Longitudinaldaten-Regression mit Anwendungen auf Adipositas.* Diploma-thesis, Ludwig-Maximilians-Universität München, Institut für Statistik.

Fenske, N., T. Kneib, and T. Hothorn (2009). Identifying risk factors for severe childhood malnutrition by boosting additive quantile regression. *Technical Report, Department of Statistics, University of Munich 052.*

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics 7*, 143–156.

Forbes, E. (1996). Gauss and the discovery of ceres. *Statistical Science 11(2)*, 89–121.

Freund, Y. and R. E. Schapire (1996a). A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences 55(1)*, 119–139.

Freund, Y. and R. E. Schapire (1996b). Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, 148–156.

Friedman, J. (1977). A recursive partitioning decision rule for nonparametric classification. *IEEE Trans. Computers C-26*, 404–408.

Friedman, J., T. Hastie, and R. Tibshirani (2000). Additive logistic regression: a statistical view of boosting. *Annals of Statistics 28.*

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics 29*, 1189–1232.

Geraci, M. and M. Bottai (2007). Quantile regression for longitudinal data using the asymmetric laplace distribution. *Biostatistics 8*, 140–154.

Hastie, T. and R. Tibshirani (1986). Generalized additive models. *Journal of Statistical Science 1*, 297–318.

Hastie, T., R. Tibshirani, and J. Friedman (2001). *The Elements of Statistical Learning.* Springer Series in Statistics. New York, NY, USA: Springer New York Inc.

Hofner, B. (2008). *Variable Selection and Model Choice in Survival Models with Time-Varying Effects.* Diploma-thesis, Ludwig-Maximilians-Universität München, Institut für Statistik.

Hothorn, T., P. Bühlmann, T. Kneib, M. Schmid, and B. Hofner (2009). *mboost: Model-Based Boosting.* R package version 1.1-2.

Hothorn, T., K. Hornik, C. Strobl, and A. Zeileis (2009). *party: A Laboratory for Recursive Partytioning.* R package version 0.9-999.

Hothorn, T., K. Hornik, and A. Zeileis (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics 15(3)*, 651–674.

Kneib, T., T. Hothorn, and G. Tutz (2009). Variable selection and model choice in geoadditive regression models. *Technical Report, Department of Statistics, University of Munich 03.*

Koenker, R. (2005). *Quantile Regression.* New York: Cambridge University Press.

Koenker, R. (2009a). *Quantile Regression in R: A vignette.* http://cran.at.r-project.org/web/packages/quantreg/vignettes/rq.pdf.

Koenker, R. (2009b). *quantreg: Quantile Regression.* R package version 4.30.

Koenker, R. and K. Hallock (2001). Quantile regression. *The Journal of Economic Perspectives 15*, 143–156.

Liaw, A. and M. Wiener (2002). Classification and regression by randomforest. *R News 2*(3), 18–22.

Lin, Y. and Y. Jeon (2002). Random forests and adaptive nearest neighbors. *Technical Report, Department of Statistics University of Wisconsin 1055.*

Meinshausen, N. (2006). Quantile regression forests. *Journal Machine Learning Research 7*, 983–999.

Meinshausen, N. (2007). *quantregForest: Quantile Regression Forests.* R package version 0.2-2.

Morgan, J. and J. Sonquist (1963). Problems in the analysis of survey data, and a proposal. *The Journal of the American Statistical Association 58*, 415–434.

Mosteller, F. and J. Tukey (1977). *Data Analysis and Regression: A second Course in Statistics.* Reading, Mass.: Addison-Wesley.

R Development Core Team (2009). *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0.

Schmid, M. and T. Hothorn (2008). Boosting additive models using component-wise p-splines as base-learners. *Computational Statistics and Data Analysis 53(2)*, 298–311.

Strasser, H. and C. Weber (1999). On the asymptotic theory of permutation statistics. *Mathematical Methods of Statistics 8*, 220–250.

Strobl, C., A.-L. Boulesteix, and T. Hothorn (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics 1*, 8–25.

Therneau, T. M., B. Atkinson, and B. Ripley. (2009). *rpart: Recursive Partitioning.* R package version 3.1-44.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B 58*(1), 267–288.

Yue, K. and R. A. Moyeed (2001). Bayesian quantile regression. *Statstics & Probability Letters 54*, 437–447.

List of Figures

List of Tables

Erklärung

Hiermit versichere ich, dass ich die Diplomarbeit selbstständig angefertigt und nur die angegebenen Quellen verwendet habe.

München, den 15. Januar 2010

(Andreas Mayr)