Senior Thesis in Mathematics

Constructing Prediction Intervals for Random Forests

Author:
Benjamin Lu

Advisor:
Dr. Jo Hardin

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

May 5, 2017

Acknowledgements

I would like to thank Professor Jo Hardin for sharing her time and expertise as my research advisor. I would also like to thank the Mathematics Department at Pomona College for the knowledge, support, and guidance it has offered me over the four years leading up to this thesis.

Contents

Chapter 1

Introduction

Random forests are an ensemble, supervised machine learning algorithm widely used to build regression models, which predict the unknown value of a continuous variable of interest, called the response variable, given known values of related variables, called predictor variables. The model is built using labeled training data—observations that consist of both known predictor variable values and known response variable values. Despite their widespread use, random forests are currently limited to point estimation, whereby the value of the response variable of an observation is predicted but no estimate of the variability associated with the prediction is given. As a result, the value of random forest predictions is limited: A prediction of a single point can be difficult to interpret or use without an understanding of its variability.

Prediction intervals compensate for this shortcoming of point estimation. Rather than giving a single point estimate of an unknown response value, a prediction interval gives a range of values that one can be confident, to a certain level, contains the true value. Although prediction intervals are superior to point estimation in this respect, the development of a prediction interval construction method for random forests has been limited, in part because of an inadequate understanding of the underlying statistical properties of random forests. Random forests are a complex algorithm, consisting of multiple base learners that use an unusual step-like regression function and introduce random variation throughout.

We propose a method of constructing prediction intervals for random forests by direct estimation of prediction error. In Chapter 2, we define prediction intervals and offer an example of a regression model for which a closed-form expression for prediction intervals has been already derived. In

Chapter 3, we introduce two relevant resampling methods: bootstrapping, which is inherent in the random forest algorithm, and jackknifing, which is used in relevant literature (Wager et al. (2014)). Chapter 4 presents an overview of the random forest algorithm and identifies terms and features that are important to our direct prediction error estimation method. In Chapter 5, we first review related work in the literature before presenting our prediction error estimation method, examining its properties, and providing empirical results that suggest its validity. In Chapter 6, we present our prediction interval construction method using the procedures discussed in Chapter 5 and show how it performs on a variety of benchmark datasets.

Chapter 2

Prediction Intervals

Point estimation of a continuous random variable—commonly a variable that can take on any value in a non-empty range of the real numbers—provides a single estimated value of the true value. Because point estimation alone provides limited information, an estimate of the variability associated with the point estimate is often desired. This estimated variability can be used to provide a range of values within which the variable is estimated to be. One such range is a prediction interval.

Definition 2.0.1. Let $\alpha \in [0, 1]$. A *$(1-\alpha)100\%$ prediction interval* for the value of an observation is an interval constructed by a procedure such that $(1-\alpha)100\%$ of the $(1-\alpha)100\%$ prediction intervals constructed by the procedure contain the true individual value of interest.

A prediction interval gives a range of estimated values for a variable of an observation drawn from a population. In contrast, a confidence interval gives a range of estimated values for a parameter of the population.

Definition 2.0.2. Let $\alpha \in [0, 1]$. A *$(1-\alpha)100\%$ confidence interval* for the parameter of a population is an interval constructed by a procedure such that $(1-\alpha)100\%$ of the $(1-\alpha)100\%$ confidence intervals constructed by the procedure contain the true parameter of interest.

We offer the following example to clarify the distinction between prediction intervals and confidence intervals.

Example 2.0.1. Suppose that we are interested in the distribution of height among 40-year-old people in the United States. A prediction interval would

give a range of plausible values for the height of a randomly selected 40-year-old person in the United States. On the other hand, a confidence interval would give a range of plausible values for some parameter of the population, such as the mean height of all 40-year-old people in the United States.

Many statistical and machine learning methods for point estimation have been created, including generalized linear models, loess, regression splines, LASSO, and support vector regression (James et al. (2013)). The extent to which procedures for constructing confidence intervals and prediction intervals have been developed varies by method. Example 2.0.2 presents a regression method for which interval construction procedures are well-established.

Example 2.0.2. Assume the normal error regression model:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i \qquad \text{for } i = 1, \ldots, n \tag{2.1}$$

where:

y_i is the observed value of the response variable of the i^{th} observation,
x_i is the observed value of the explanatory variable of the i^{th} observation,
β_0 and β_1 are parameters, and
ϵ_i are independent and identically distributed $N(0, \sigma^2)$.

Let $\hat{\beta}_0, \hat{\beta}_1$ be least squares estimators of their corresponding parameters, so that the estimate of the regression function is

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x \tag{2.2}$$

where $\hat{y}$ is the estimated response value. Then a $(1-\alpha)100\%$ prediction interval for the unknown response y_h of a new observation with known predictor value x_h is given by

$$\hat{y}_h \pm t_{(\frac{\alpha}{2}, n-2)} \sqrt{MSE\left(1 + \frac{1}{n} + \frac{(x_h - \overline{x})^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}\right)}, \tag{2.3}$$

where $t_{(\frac{\alpha}{2}, n-2)}$ is the $\frac{\alpha}{2}$ percentile of the t-distribution with $n-2$ degrees of freedom, $\overline{x}$ denotes the average of the n observed predictor variable values, and

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{2.4}$$

A $(1-\alpha)100\%$ confidence interval for the mean response $E(Y_h)$ of observations with predictor value x_h is given by

$$\hat{y}_h \pm t_{(\frac{\alpha}{2}, n-2)} \sqrt{MSE\left(\frac{1}{n} + \frac{(x_h - \overline{x})^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}\right)}. \tag{2.5}$$

Notice that the center of the prediction interval for the response of an observation with predictor value x_h is the same as the center of the confidence interval for the mean response of observations with predictor value x_h. In particular, both are $\hat{y}_h$. However, the upper and lower bounds of the prediction interval are each farther from $\hat{y}_h$ than the corresponding confidence interval bounds because of the additional MSE term. In the least squares setting, the expression for a prediction interval contains the additional MSE term to estimate the variability of the response values of the individual observations within the population. Since a confidence interval estimates the mean response rather than an individual response, the expression for a confidence interval does not include the additional MSE term.

While the procedures for prediction interval and confidence interval construction are well-established in the example above, they have not been developed fully developed for random forests.

Chapter 3

Resampling Methods

Resampling methods are useful techniques for data analysis because they offer an alternative to traditional statistical inference that is often simpler, more accurate, and more generalizable. In some cases, when the statistical theory has not been sufficiently developed or the assumptions required by existing theory are not met, resampling methods are the only feasible approach to inference. While the implementation of resampling methods is generally computationally expensive, the increasing amount of computational power available in modern computers is making these methods more readily accessible in practice. In this chapter, we review two resampling methods in particular: bootstrapping and jackknifing. The former is a fundamental component of the random forest algorithm, and the latter has been used in literature related to this thesis (Wager et al. (2014)).

3.1 Bootstrapping

The bootstrap is a resampling method developed by Efron (1979). It has many uses, but the most relevant for our purposes are stabilizing predictions and estimating variance. In this section, we will define the bootstrap sample, which is the basis of bootstrapping, and present an example of its application. First, we define related concepts.

Definition 3.1.1. Let X be a real-valued random variable. The *cumulative distribution function* F of X is defined by

$$F_X(x) = P(X \leq x). \tag{3.1}$$

Any real-valued random variable can be described by its cumulative distribution function, which encodes many important statistical and probabilistic properties of the random variable. As such, an estimate of the cumulative distribution function can be very valuable. Indeed, estimating some aspect of a random variable's cumulative distribution function is often the goal or an intermediate step of statistical inference.

Definition 3.1.2. Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a sample from a population with cumulative distribution function F, usually unknown. The *empirical distribution function* $\hat{F}$ is defined by

$$\hat{F}(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(x_i \leq t), \tag{3.2}$$

where $\mathbb{1}$ is the indicator function defined by

$$\mathbb{1}(x \leq t) = \begin{cases} 1 & x \leq t \\ 0 & x > t. \end{cases} \tag{3.3}$$

With these foundational elements, we can now define a bootstrap sample.

Definition 3.1.3. Let $\hat{F}$ be an empirical distribution function obtained from a sample $\mathbf{x} = (x_1, \ldots, x_n)$. A *bootstrap sample* $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$ is a random sample of observations independently drawn from $\hat{F}$.

In practice, a bootstrap sample $\mathbf{x}^*$ is obtained by resampling with replacement from the original sample $\mathbf{x}$, so that some observations from $\mathbf{x}$ may appear in $\mathbf{x}^*$ more than once and others may not appear at all.

We began our discussion of bootstrapping by mentioning that it can be used for variance estimation. In the following subsection, we examine the use of bootstrapping for estimating the variance of a statistic as a general example. Its use in prediction stabilization will be covered in Chapter 4.

3.1.1 Application: Variance Estimation

Let θ be a parameter of interest of F, and let $\hat{\theta} = s(\mathbf{x})$ be the statistic used to estimate the parameter. We are often interested in the standard error of $\hat{\theta}$, denoted $\mathrm{SE}(\hat{\theta})$ and defined as the standard deviation of the sampling distribution of $\hat{\theta}$. Bootstrapping offers a non-analytic way to estimate $\mathrm{SE}(\hat{\theta})$,

outlined in Algorithm 1, which is convenient when the sampling distribution is complicated or unknown. This approach, which is discussed further in the example below, is valid in many cases because, under certain conditions, the distribution of $\hat{\theta}^* - \hat{\theta}$ approximates the distribution of $\hat{\theta} - \theta$, where $\hat{\theta}^*$ is the bootstrap statistic, described in Definition 3.1.4.

Definition 3.1.4. Let $\mathbf{x}$ be a sample from a population and $\hat{\theta} = s(\mathbf{x})$ be a statistic used to estimate a parameter θ. A corresponding *bootstrap statistic* $\hat{\theta}^*$ is obtained by applying s to a bootstrap sample $\mathbf{x}^*$. That is, $\hat{\theta}^* = s(\mathbf{x}^*)$.

Algorithm 1 Bootstrap Algorithm to Estimate $\mathrm{SE}(\hat{\theta})$

procedure
obtain a sample $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ from the population
define a statistic of interest, $\hat{\theta} = s(\mathbf{x})$
    for b in $1, \ldots, B$ **do**
        obtain the b^{th} bootstrap sample $\mathbf{x}^{*,\mathbf{b}} = (x_1^{*,b}, x_2^{*,b}, \ldots, x_n^{*,b})$
        calculate $\hat{\theta}^*(b) = s(\mathbf{x}^{*,\mathbf{b}})$
    end for
estimate $\mathrm{SE}(\hat{\theta})$ by the sample standard deviation of the B replicates,

$$\widehat{\mathrm{SE}_B(\hat{\theta})} = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\theta}^*(b) - \bar{\hat{\theta}}^*)^2}, \tag{3.4}$$

where

$$\bar{\hat{\theta}}^* = \frac{1}{B}\sum_{b=1}^{B}\hat{\theta}^*(b). \tag{3.5}$$

    return $\widehat{\mathrm{SE}_B(\hat{\theta})}$
end procedure

Example 3.1.1. Let F be an exponential distribution with rate parameter $\lambda = 0.5$, $n = 100$ be the size of samples drawn from F, and $B = 5000$ be the number of bootstrap samples generated. Suppose that the value of λ is unknown. Let the parameter of interest be the mean $\mu = \frac{1}{\lambda}$ of the distribution, and let the statistic used to estimate the parameter be the sample mean,

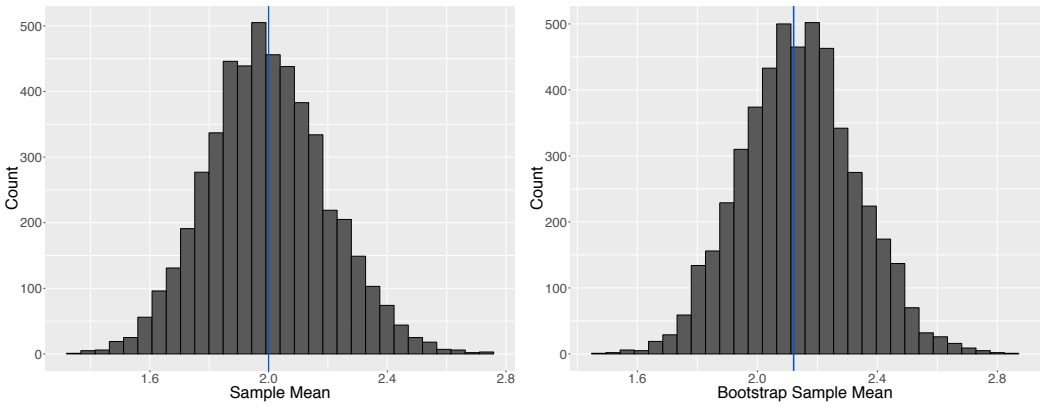

Figure 3.1: A histogram of the estimates of μ obtained by drawing 100 observations from the exponential distribution ($\lambda = 0.5$) 5000 times (LEFT). A histogram of the estimates of μ obtained by resampling with replacement from a 100-observation sample of the exponential distribution 5000 times (RIGHT).

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{3.6}$$

We are interested in estimating $\text{SE}(\hat{\mu})$, the standard deviation of the sampling distribution of $\hat{\mu}$. The left panel of Figure 3.1 displays a histogram of the sampling distribution of $\hat{\mu}$, with the blue line drawn at $\mu = 2$. It was generated by obtaining B samples of size n from F, calculating $\hat{\mu}_i$ for the i^{th} sample, and plotting the frequencies of $\hat{\mu}_1, \hat{\mu}_2, \ldots, \hat{\mu}_B$. Using these B sample statistics, we can estimate $\text{SE}(\hat{\mu})$ with the usual estimate of the standard deviation of a distribution:

$$\widehat{\text{SE}(\hat{\mu})} = \sqrt{\frac{1}{B-1}\sum_{i=1}^{B}(\hat{\mu}_i - \bar{\hat{\mu}})}, \tag{3.7}$$

where

$$\bar{\hat{\mu}} = \frac{1}{B}\sum_{i=1}^{B}\hat{\mu}_i. \tag{3.8}$$

In practice, however, we do not obtain repeated samples of n observations from the population. Accordingly, Algorithm 1 can be used to estimate $\text{SE}(\hat{\mu})$ using only one sample of size n. The right panel of Figure 3.1 displays a histogram of the B bootstrap statistics $\hat{\mu}_1^*, \ldots, \hat{\mu}_B^*$ obtained by following Algorithm 1, with the blue line drawn at the mean of our original sample. We note two points of comparison between the left and right panels of Figure 3.1. First, observe that the left distribution is centered at 1.9959, close to the true population mean of 2, whereas the right distribution is centered at 2.1388, which is close to the original sample mean of 2.1341. This is to be expected: In general, assuming an unbiased statistic is used, the sampling distribution will be centered at the true population parameter, while the bootstrap sampling distribution will be centered at the sample statistic. The second point to note is that the estimated standard deviations of the two distributions—that is, the two estimates of $\text{SE}(\hat{\mu})$—are approximately equal. The estimated standard deviation of the sampling distribution is 0.2014, while the estimated standard deviation of the bootstrap sampling distribution is 0.1908. While a discussion of the conditions under which the bootstrap estimate of the standard error approximates the true standard error well is beyond the scope of this thesis, it suffices to say that the conditions hold in many practical applications, making Algorithm 1 a useful method for estimating the standard error of a statistic.

3.2 The Jackknife

The jackknife is another resampling method, usually used to estimate the bias and variance of an estimator. Because of its resemblance to the bootstrap, we will only briefly define the jackknife sample and some associated estimates.

Definition 3.2.1. Let $\mathbf{x}$ be a sample of size n. The ith *jackknife sample* $\mathbf{x_{(i)}} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ is obtained by removing the i^{th} observation from $\mathbf{x}$.

Definition 3.2.2. Let $\mathbf{x}$ be a sample from a population and $\hat{\theta} = s(\mathbf{x})$ be a statistic used to estimate a parameter θ. The ith *jackknife statistic* $\hat{\theta}_{(i)}$ is obtained by applying s to the i^{th} jackknife sample $\mathbf{x_{(i)}}$. That is, $\hat{\theta}_{(i)} = s(\mathbf{x_{(i)}})$.

Definition 3.2.3. Let $\mathbf{x}$ be a sample from a population and $\hat{\theta} = s(\mathbf{x})$ be a statistic used to estimate a parameter θ. The *jackknife estimate of the bias*

of $\hat{\theta}$ is given by

$$\widehat{\text{BIAS}}_J = (n-1)(\bar{\hat{\theta}}_{(\cdot)} - \hat{\theta}), \tag{3.9}$$

where

$$\bar{\hat{\theta}}_{(\cdot)} = \frac{1}{n}\sum_{i=1}^{n}\hat{\theta}_{(i)}. \tag{3.10}$$

Definition 3.2.4. Let $\mathbf{x}$ be a sample from a population and $\hat{\theta} = s(\mathbf{x})$ be a statistic used to estimate a parameter θ. The *jackknife estimate of the standard error of* $\hat{\theta}$ is given by

$$\widehat{\text{SE}}_J = \sqrt{\frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_{(i)} - \bar{\hat{\theta}}_{(\cdot)})^2}. \tag{3.11}$$

While we do not use the jackknife in our proposed method of prediction interval construction, related work in the literature employs the jackknife, as discussed in Subsection 5.1.1 (Wager et al. (2014)).

Chapter 4

Random Forests

Random forests are a widely used machine learning algorithm developed by Breiman (2001). They can be categorized into random forests for regression and random forests for classification. The former has a continuous response variable, while the latter has a categorical response variable. This thesis focuses on random forests for regression. Random forests have a number of properties that make them favorable choices for modeling. They are usually fairly accurate, they can automatically identify and use the explanatory variables most relevant to predicting the response and therefore have a built-in measure of variable importance, they can use both categorical and continuous explanatory variables, they do not assume any particular relationship between any variables, their predictions are asymptotically normal under certain conditions, and they are consistent under certain conditions (Wager (2016), Scornet et al. (2015)). This chapter is devoted to examining the random forest algorithm and identifying key features and terms that will be used in later chapters. Because a random forest is a collection of classification or regression trees (CART), we first examine the CART algorithm.

4.1 Classification and Regression Trees

Breiman's classification and regression tree (CART) algorithm is one of many decision tree models, but for convenience, we will use the term "decision tree" to refer to Breiman's CART (Gordon et al. (1984)). Whether used for regression or classification, the CART algorithm greedily partitions the predictor space to minimize the homogeneity of the observed response values within

each subspace. Algorithmically, the relevant difference between classification trees and regression trees is the way homogeneity is measured.

In what follows, let $Z = \{\mathbf{z_1}, \ldots, \mathbf{z_n}\}$ denote a set of n observations, where $\mathbf{z_i} = (x_{i,1}, \ldots, x_{i,p}, y_i)$ is the i^{th} observation, consisting of p explanatory variable values $x_{i,1}, \ldots, x_{i,p}$ and a response variable y_i. Let $\mathbf{x_i} = (x_{i,1}, \ldots, x_{i,p})$ denote the vector of explanatory variable values for $\mathbf{z_i}$. Let $|S|$ denote the cardinality of a set S unless otherwise specified.

The decision tree algorithm can be divided into three stages: tree growth, tree pruning, and prediction. We examine each in turn.

4.1.1 Tree Growth

Tree growth refers to the process of constructing the decision tree. It is done by using a sample Z to recursively partition first the predictor space and then the resulting subspaces. The decision tree consists of the resulting partition of the predictor space, along with the mean observed response value or majority observed response level within each subspace, depending on the type of response variable. Algorithm 2 outlines the specific procedure to grow a decision tree. The algorithm makes use of some terms that are defined below.

Definition 4.1.1. Let Z be a sample of n observations with p explanatory variables and a continuous response variable. Let D be a subspace of the predictor space. Then the *within-node sum of squares (WSS)* is given by

$$WSS(D) = \sum_{i \in \{1,\ldots,n\}: \mathbf{x_i} \in D} (y_i - \hat{y}_D)^2 \tag{4.1}$$

where $\hat{y}_D$ is the mean response of the observations in D.

Definition 4.1.2. Let Z be a sample of n observations with p explanatory variables and a categorical response variable with K levels. Let D be a subspace of the predictor space. Then the *Gini index* is given by

$$G(D) = \sum_{k=1}^{K} \hat{y}_{Dk}(1 - \hat{y}_{Dk}) \tag{4.2}$$

where $\hat{y}_{Dk}$ is the proportion of observations contained in D that have response level k.

Algorithm 2 indicates that a decision tree is obtained by recursively splitting the predictor space and resulting subspaces to minimize WSS or the Gini index at each step. It can be seen that Equation 4.1 is minimized when the response values within a subspace are closest to their mean. Similarly, Equation 4.2 is small when the proportion of observations within the subspace that have response level k is close to 0 or 1—in other words, when most of the observations in the subspace have the same response level. Thus, as stated earlier, a decision tree is constructed to minimize the homogeneity of the observed response within each subspace resulting from the partition.

4.1.2 Tree Pruning

The next stage in the decision tree algorithm is the pruning process, outlined in Algorithm 3. Decision trees are pruned because they tend to overfit the data, worsening their predictive accuracy. We introduce the following terms to broadly describe the pruning process. However, because, as discussed in Section 4.2, random forests consist of unpruned decision trees, we do not describe the pruning process and its associated terms in great detail here. Further discussion of the pruning process can be found in James et al. (2013).

Definition 4.1.3. Let T be a decision tree. A *node* of T is a subspace of the predictor space returned by Algorithm 2.

Definition 4.1.4. Let T be a decision tree. A *terminal node* of T is a node of T that is not itself split into two nodes.

In the tree pruning stage, we seek to minimize the sum of the WSS or Gini indices for the terminal nodes of the decision tree and a cost α multiplied by the number of terminal nodes. This is done by "backtracking" the recursive splitting process, rejoining nodes that were split, to obtain the optimal subtree. In order to determine the optimal cost to impose on the number of terminal nodes contained in the decision tree, K-fold cross-validation is employed using the error rate defined below.

Definition 4.1.5. Let T be a decision tree and Z be a set of n test observations. Then the *error rate* of T on Z is

$$E(T, Z) = \sum_{j=1}^{|T|} \sum_{i:\mathbf{x_i} \in D_j} (y_i - \hat{y}_{D_j})^2 \tag{4.3}$$

Algorithm 2 Recursive Decision Tree Construction Algorithm

procedure splitNode(set S of observations with p predictor variables, subspace D of the predictor space, integer $minSize$)
  if $|S| < minSize$ **then**
    return D
    break
  end if
  for r in $1, \ldots, p$ **do**
    if the r^{th} predictor variable is numeric **then**
      define the functions D_1, D_2 by

$$D_1(r, s) = \{\mathbf{x} \in D : x_r < s\}$$
$$D_2(r, s) = \{\mathbf{x} \in D : x_r \geq s\}$$

      where x_r is the value of the r^{th} predictor variable of $\mathbf{x}$
    end if
    if the r^{th} predictor variable is categorical **then**
      define the functions D_1, D_2 by

$$D_1(r, s) = \{\mathbf{x} \in D : x_r \in s\}$$
$$D_2(r, s) = \{\mathbf{x} \in D : x_r \notin s\}$$

      where x_r is the value of the r^{th} predictor variable of $\mathbf{x}$
    end if
  end for
  if the response variable is numeric **then**
    identify (r, s) that minimizes $WSS(D_1(r, s)) + WSS(D_2(r, s))$
  end if
  if the response variable is categorical **then**
    identify (r, s) that minimizes $G(D_1(r, s)) + G(D_2(r, s))$
  end if
  splitNode($\{\mathbf{z} \in S : \mathbf{x} \in D_1(r, s)\}, D_1(r, s), minSize$)
  splitNode($\{\mathbf{z} \in S : \mathbf{x} \in D_2(r, s)\}, D_2(r, s), minSize$)
end procedure

if the response value is numeric, and

$$E(T, Z) = \sum_{j=1}^{|T|} \sum_{i:\mathbf{x_i} \in D_j} \mathbb{1}(y_i \neq \hat{y}_{D_j}) \tag{4.4}$$

if the response value is categorical, where $|T|$ denotes the number of terminal nodes of T, D_j denotes the j^{th} terminal node of T, $\mathbb{1}$ denotes the indicator function, and $\hat{y}_{D_j}$ denotes the mean observed response of D_j if the response is numeric and the majority observed level of D_j if the response is categorical.

4.1.3 Tree Prediction

As described in Subsections 4.1.1 and 4.1.2, a decision tree T is built on a sample Z consisting of n observations, each with p observed explanatory variable values and an observed response variable value. To use T to predict the response value y_h of a new observation $\mathbf{x_h}$, we identify the terminal node that contains $\mathbf{x_h}$. If the response variable is continuous, then we predict y_h to be the mean response of the observations from the sample Z that are in that terminal node. If the response variable is categorical, then we predict y_h to be the response level most frequently observed among observations from the sample Z that are in that terminal node. We denote the decision tree's predicted response of an observation $\mathbf{z_h}$ by $T(\mathbf{x_h})$. So, if $D \ni \mathbf{x_h}$, then

$$T(\mathbf{x_h}) = \frac{1}{|\{i : \mathbf{x_i} \in D\}|} \sum_{i:\mathbf{x_i} \in D} y_i \tag{4.8}$$

if the response variable is continuous, and

$$T(\mathbf{x_h}) = \underset{k \in K}{\operatorname{argmax}} \sum_{i:\mathbf{x_i} \in D} \mathbb{1}(y_i = k) \tag{4.9}$$

if the response variable is categorical, where K is the set of possible levels for the categorical response variable.

Notice that, whether the response variable is continuous or categorical, the test observation's response is predicted using the observed response values of observations in the same terminal node as the test observation. This notion of being in the same terminal node will be important in Chapters 5 and 6, so we define the following term.

Definition 4.1.6. Let T be a decision tree. Two observations $\mathbf{z_1}$ and $\mathbf{z_2}$ are *cohabitants* if $\mathbf{x_1}$ and $\mathbf{x_2}$ are in the same terminal node of T.

Algorithm 3 Decision Tree Pruning Algorithm

procedure PRUNETREE(decision tree T; sample Z used to build T; sequence of numbers $\alpha_1, \ldots, \alpha_M$; integer K)
  Divide Z into K partitions $Z_1, \ldots, Z_K$ of approximately equal size
  for k in $1, \ldots, K$ **do**
    Build a decision tree T_k on $Z \backslash Z_k$ using Algorithm 2
    for m in $1, \ldots, M$ **do**
      Let $C(m, k)$ be defined by

$$C(m,k) = \min_{T_{k,m} \subset T_k} E(T_{k,m}, Z_k) + \alpha_m \left|T_{k,m}\right| \tag{4.5}$$

      where the minimum is taken over all possible subtrees $T_{k,m}$ of T_k, and $|T_{k,m}|$ denotes the number of terminal nodes of $T_{k,m}$
    end for
  end for
  Let $n \in \{1, \ldots, M\}$ be the value that minimizes

$$C(n, \cdot) = \frac{1}{K} \sum_{k=1}^{K} C(n,k) \tag{4.6}$$

  Let T_{α_n} be the subtree of T that minimizes

$$\sum_{j=1}^{|T_{\alpha_n}|} WSS(D_j) + \alpha_n \left|T_{\alpha_n}\right| \tag{4.7}$$

  where $|T_{\alpha_n}|$ denotes the number of terminal nodes of T_{α_n}, and D_j denotes the j^{th} terminal node of T_{α_n}
  return T_{α_n}
end procedure

4.2 Random Forests

A random forest, which we denote φ, built on a sample Z is a collection of B decision trees, with each decision tree built on its own bootstrap sample Z^{*b} of the original sample Z. The procedure for constructing each decision tree on its bootstrap sample follows Algorithm 2, with one difference: Instead of choosing from all of the predictor variables in order to identify the node split that minimizes the resulting subspaces' WSS or Gini indices, each decision tree can only choose from a random subset of the predictor variables when identifying the node split that minimizes the resulting subspaces' WSS or Gini indices. Each time a node is split during the construction of the decision tree, the subset of predictor variables that can be used to split the node is randomly chosen. The size of the subset is a parameter that is chosen by the user and can be optimized using cross-validation. However, conventionally, if p is the total number of predictor variables, then the number of randomly chosen predictor variables that can be used to split a node is $p/3$ or $\sqrt{p}$. The decision trees in a random forest are not pruned.

Because each of the B decision trees is built on its own bootstrap sample of Z, it is likely that some observations in Z will not be used in the construction of a particular decision tree. This phenomenon figures prominently in Chapter 5, so we offer the following definitions.

Definition 4.2.1. Let φ be a random forest consisting of B decision trees $T_1^*, \ldots, T_B^*$, with the b^{th} decision tree built on the b^{th} bootstrap sample $Z^{*,b}$ of the original sample Z. An observation $\mathbf{z} \in Z$ is *out of bag for the b^{th} decision tree* if $\mathbf{z} \notin Z^{*,b}$.

Definition 4.2.2. Let φ be a random forest consisting of B decision trees $T_1^*, \ldots, T_B^*$, with the b^{th} decision tree built on the b^{th} bootstrap sample $Z^{*,b}$ of the original sample Z. An observation $\mathbf{z} \in Z$ is *in bag for the b^{th} decision tree* if $\mathbf{z} \in Z^{*,b}$.

The random forest's prediction of the response value y_h of an observation $\mathbf{z_h}$ with observed predictor values $\mathbf{x_h}$ is the mean of the values predicted by the decision trees of the random forest in the case of a continuous response variable, and the level most frequently predicted by the decision trees of the random forest in the case of a categorical response variable. This aggregation of the bootstrap decision trees' predictions stabilizes the random forest prediction by reducing its variability and the risk of overfitting. We denote

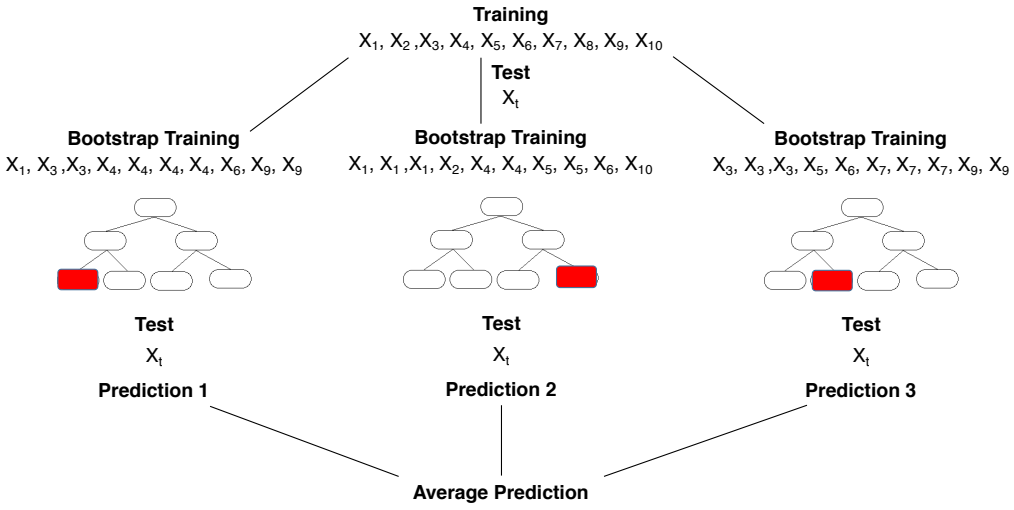

Figure 4.1: A diagram of the random forest construction and prediction procedure.

the random forest's prediction of the response value of $\mathbf{z_h}$ by $\varphi(\mathbf{x_h})$. Using this notation, the random forest prediction of the response of an observation $\mathbf{z_h}$ is given by

$$\varphi(\mathbf{x_h}) = \frac{1}{B}\sum_{b=1}^{B} T_b^*(\mathbf{x_h}) \tag{4.10}$$

if the response variable is continuous, and

$$\varphi(\mathbf{x_h}) = \underset{k\in K}{\operatorname{argmax}} \sum_{b=1}^{B} \mathbb{1}(T_b^*(\mathbf{x_h}) = k) \tag{4.11}$$

if the response variable is categorical, where K is the set of possible levels for the categorical response variable.

Chapter 5

Variability Estimation

One of the foremost challenges to constructing valid prediction intervals for random forests is developing suitable estimates of variability. Since random forests have a complex structure, it can be difficult to identify appropriate estimators and to prove their validity. In this chapter, we review different variability estimation methods that have been advanced in the literature before turning to our own proposed methods.

5.1 Related Work

5.1.1 Random Forest Standard Error Estimation

Work has been done to develop estimators of the standard error (or variance) of a random forest estimator,

$$se(\hat{\theta}_B^{RF}(\mathbf{x_h})) \equiv \sqrt{Var(\hat{\theta}_B^{RF}(\mathbf{x_h}))}, \tag{5.1}$$

where $\hat{\theta}_B^{RF}(\mathbf{x_h})$ is the predicted response value of observations with predictor variable values $\mathbf{x_h}$ using a random forest with B trees. This standard error measures how random forest predictions of the response of observations with predictor variable values $\mathbf{x_h}$ vary across different samples. Notice that we use the notation $\hat{\theta}_B^{RF}(\mathbf{x_h})$ rather than simply $\hat{y}_h$ or $\varphi(\mathbf{x_h})$, even though these values in practice are the same. We adopt such notation because, in this setting, a random forest prediction is conceptualized as a statistic, the sampling variability of which is the value of interest. Thus, $se(\hat{\theta}_B^{RF}(\mathbf{x_h}))$ measures how far the statistic—in this case, the *random forest prediction* of the response of

observations with predictor variable values $\mathbf{x_h}$ obtained by building B trees on a sample—is from its *expected value*, which is the average random forest prediction of the response obtained by building B trees across different samples. It does not measure how far the *random forest prediction* of the response is from the *true response value*, as we shall see at the end of this subsection, and it is therefore unsuitable for the construction of prediction intervals.

Sexton and Laake (2009) and Wager et al. (2014) propose estimators of $se(\hat{\theta}_B^{RF}(\mathbf{x_h}))$. We first review the work by Sexton and Laake (2009), who propose three different methods, before examining the work by Wager et al. (2014). Note that each of the three methods requires two levels of bootstrapping. This is because, as in Algorithm 1, the sampling variability of the statistic of interest is empirically estimated by calculating the statistic on many bootstrap samples of the original sample; since the statistic of interest in this case is the random forest prediction of an observation's response, each bootstrap sample itself must be bootstrapped in order to construct the random forest used to predict the observation's response.

Sexton and Laake (2009) call their first estimator of $se(\hat{\theta}_B^{RF}(\mathbf{x_h}))$ the "Brute Force" estimator.

Definition 5.1.1. Let $\hat{\theta}_B^{RF}(\mathbf{x_h})$ be the predicted response value of an observation $\mathbf{z_h}$ by a random forest fit on a sample Z of size n. For m in $1, \ldots, M$, let $Z^{*,m}$ denote the m^{th} bootstrap sample from Z, and let the predicted response of $\mathbf{z_h}$ be the random forest fit on the m^{th} bootstrap sample of Z be denoted by

$$\hat{\theta}_B^{RF,*,m}(\mathbf{x_h}) = \frac{1}{B}\sum_{b=1}^{B} T_b^{*,m}(\mathbf{x_h}), \tag{5.2}$$

where $T_b^{*,m}$ is the decision tree built on the b^{th} bootstrap sample of the m^{th} bootstrap sample of Z. Then the *Brute Force estimator of the standard error of the random forest at* $\mathbf{x_h}$ is

$$\widehat{se}^{BF}(\hat{\theta}_B^{RF}(\mathbf{x_h})) = \left[\frac{1}{M-1}\sum_{m=1}^{M}(\hat{\theta}_B^{RF,*,m}(\mathbf{x_h}) - \bar{\hat{\theta}}_B^{RF,*}(\mathbf{x_h}))^2\right]^{1/2} \tag{5.3}$$

where

$$\bar{\hat{\theta}}_B^{RF,*}(\mathbf{x_h}) = \frac{1}{M}\sum_{m=1}^{M}\hat{\theta}_B^{RF,*,m}(\mathbf{x_h}) \tag{5.4}$$

is the average random forest prediction of the response of $\mathbf{z_h}$ over the M bootstrap samples.

Sexton and Laake (2009) call this the Brute Force estimator because of the large number of decision trees that are constructed in the procedure. Because a random forest is fit on each of the M bootstrap samples, which entails bootstrapping each bootstrap sample B times, a total of MB decision trees are constructed using this approach. The computational demand of the Brute Force estimator leads the authors to recommend against its general use, but they use it as a baseline against which the other two methods can be compared.

The second method, which they call the "Biased Bootstrap" estimator, is similar to the Brute Force estimator. Effectively, the only difference between the two is that the Biased Bootstrap estimator fits a smaller random forest, consisting of R trees with $R < B$, to each of the M bootstrap samples.

Definition 5.1.2. Let $\hat{\theta}_B^{RF}(\mathbf{x_h})$ be the predicted response value of an observation $\mathbf{z_h}$ by a random forest fit on a sample Z of size n. For m in $1, \ldots, M$, let $Z^{*,m}$ denote the m^{th} bootstrap sample from Z, and let the predicted response of $\mathbf{z_h}$ by the random forest fit on the m^{th} bootstrap sample of Z be denoted by

$$\hat{\theta}_R^{RF,*,m}(\mathbf{x_h}) = \frac{1}{R}\sum_{r=1}^{R} T_r^{*,m}(\mathbf{x_h}), \tag{5.5}$$

where $R < B$ and $T_r^{*,m}$ is the decision tree built on the r^{th} bootstrap sample of the m^{th} bootstrap sample of Z. Then the *Biased Bootstrap estimator of the standard error of the random forest at* $\mathbf{x_h}$ is

$$\widehat{se}^{BB}(\hat{\theta}_B^{RF}(\mathbf{x_h})) = \left[\frac{1}{M-1}\sum_{m=1}^{M}(\hat{\theta}_R^{RF,*,m}(\mathbf{x_h}) - \bar{\hat{\theta}}_R^{RF,*}(\mathbf{x_h}))^2\right]^{1/2} \tag{5.6}$$

where

$$\bar{\hat{\theta}}_R^{RF,*}(x) = \frac{1}{M}\sum_{m=1}^{M}\hat{\theta}_R^{RF,*,m}(x). \tag{5.7}$$

Sexton and Laake (2009) show that the Biased Bootstrap estimator of the standard error is biased upward if $R < B$ because the Monte Carlo error in $\hat{\theta}_R^{RF}$ is greater than the Monte Carlo error in $\hat{\theta}_B^{RF}$. Moreover, the Monte Carlo error of the Biased Bootstrap estimator of the standard error is larger

than the Monte Carlo error of the Brute Force estimator. However, Sexton and Laake (2009) quantify this error and show how it can be approximately controlled by varying R.

The third method, which the authors call the Noisy Bootstrap estimator, corrects for the bias of the Biased Bootstrap estimator but still requires two levels of bootstrapping.

Definition 5.1.3. Maintain the same notation as Definitions 5.1.1 and 5.1.2. Then the *Noisy Bootstrap estimator of the standard error of the random forest at* $\mathbf{x_h}$ is

$$\widehat{se}^{NB}(\hat{\theta}_B^{RF}(\mathbf{x_h})) = \left[\widehat{Var}^{BB}(\hat{\theta}_B^{RF}(\mathbf{x_h})) - \widehat{bias}[\widehat{Var}^{BB}(\hat{\theta}_B^{RF}(\mathbf{x_h}))]\right]^{1/2} \tag{5.8}$$

where

$$\widehat{Var}^{BB}(\hat{\theta}_B^{RF}(\mathbf{x_h})) \equiv [\widehat{se}^{BB}(\hat{\theta}_B^{RF}(\mathbf{x_h}))]^2 \tag{5.9}$$

from Definition 5.1.2, and

$$\widehat{bias}[\widehat{Var}^{BB}(\hat{\theta}_B^{RF}(\mathbf{x_h}))] = \frac{1/R - 1/B}{MR(R-1)} \sum_{m=1}^{M}\sum_{r=1}^{R}(T_r^{*,m}(\mathbf{x_h}) - \hat{\theta}_R^{RF,*,m}(\mathbf{x_h}))^2. \tag{5.10}$$

Wager et al. (2014) approach the task of estimating $se(\hat{\theta}_B^{RF}(\mathbf{x_h}))$ by focusing on jackknife procedures. They examine two estimators: the Jackknife-after-Bootstrap estimator, and the Infinitesimal Jackknife estimator. In both cases, they identify bias-corrected versions of their estimators, which we also describe. We begin with the Jackknife-after-Bootstrap estimator and its bias-corrected version.

Definition 5.1.4. Let $\hat{\theta}_B^{RF}(\mathbf{x_h})$ be the predicted response value of an observation $\mathbf{z_h}$ by a random forest fit on a sample Z of size n. The Monte Carlo approximation to the *Jackknife-after-Bootstrap estimate of the standard error of the random forest at* $\mathbf{x_h}$ is

$$\widehat{se}^{J}(\hat{\theta}_B^{RF}(\mathbf{x_h})) = \left[\frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_{(-i)}^{RF}(\mathbf{x_h}) - \hat{\theta}^{RF}(\mathbf{x_h}))^2\right]^{1/2} \tag{5.11}$$

where

$$\hat{\theta}_{(-i)}^{RF}(\mathbf{x_h}) = \frac{1}{|\{b : \langle \mathbf{z_i}\rangle_b = 0\}|}\sum_{b:\langle \mathbf{z_i}\rangle_b = 0} T_b^*(\mathbf{x_h}). \tag{5.12}$$

Here, and throughout this thesis, $\langle \mathbf{z} \rangle_b$ denotes the number of times $\mathbf{z}$ appears in the b^{th} bootstrap sample of the random forest. Thus, Equation 5.12 gives the mean predicted response of $\mathbf{z_h}$ over only the trees for which the i^{th} observation of Z is out of bag.

The bias-corrected version of the Jackknife-after-Bootstrap estimator is given in Definition 5.1.5.

Definition 5.1.5. Maintain the same notation as in Definition 5.1.4. The bias-corrected Monte Carlo approximation to the Jackknife-after-Bootstrap estimate of the standard error of the random forest at $\mathbf{x_h}$ is

$$\widehat{se}^{J-U}(\hat{\theta}_B^{RF}(\mathbf{x_h})) = \left[\left(\widehat{se}^{J}(\hat{\theta}_B^{RF}(\mathbf{x_h})) \right)^2 - (e-1)\frac{n}{B^2} \sum_{b=1}^{B} (T_b^*(\mathbf{x_h}) - \overline{T}^*(\mathbf{x_h}))^2 \right]^{1/2} \tag{5.13}$$

where $\overline{T}^*(\mathbf{x_h})$ is the mean of the $T_b^*(\mathbf{x_h})$.

Next, we present the Infinitesimal Jackknife estimator proposed by Wager et al. (2014).

Definition 5.1.6. Let $\hat{\theta}_B^{RF}(\mathbf{x_h})$ be the predicted response value of an observation $\mathbf{z_h}$ by a random forest fit on a sample Z of size n. The Monte Carlo approximation to the *Infinitesimal Jackknife estimate of the standard error of the random forest at* $\mathbf{x_h}$ is

$$\widehat{se}^{IJ}(\hat{\theta}_B^{RF}(\mathbf{x_h})) = \left[\sum_{i=1}^{n} \left(\frac{1}{B} \sum_{b=1}^{B} (\langle \mathbf{z_i} \rangle_b - 1)(T_b^*(\mathbf{x_h}) - \overline{T}^*(\mathbf{x_h})) \right)^2 \right]^{1/2}. \tag{5.14}$$

Finally, we define the bias-corrected Infinitesimal Jackknife estimator.

Definition 5.1.7. Maintain the same notation as in Defintion 5.1.6. The bias-corrected Monte Carlo approximation to the Infinitesimal Jackknife estimate of the variance of the random forest is

$$\widehat{se}^{IJ-U}(\hat{\theta}_B^{RF}(\mathbf{x_h})) = \left[\left(\widehat{se}^{IJ}(\hat{\theta}_B^{RF}(\mathbf{x_h})) \right)^2 - \frac{n}{B^2} \sum_{b=1}^{B} (T_b^*(\mathbf{x_h}) - \overline{T}^*(\mathbf{x_h}))^2 \right]^{1/2}. \tag{5.15}$$

While the estimators proposed by Sexton and Laake (2009) and Wager et al. (2014) are informative in understanding the structure of random forests, the sources of variation in random forests, and possible ways of estimating such variation, they do not resolve the issue that this thesis seeks to address. Sexton and Laake (2009) and Wager et al. (2014) seek to estimate how random forest predictions of the response of observations with a certain set of predictor variable values vary from random forest to random forest. However, this variability does not enable the construction of prediction intervals for random forests. To see why, consider the following two possibilities:

1. The response variable of observations with a certain set of predictor variable values has high variance while the random forest predictions of those observations have low variance. This might happen if the random forest predictions estimate the mean response with high precision. The left panel of Figure 5.1 offers a visualization of this phenomenon. In that plot, the conditional response is distributed $N(0, 16)$, while the random forest predictions are distributed $N(0, 4)$.

2. The response variable variance and the random forest prediction variance are equal, but their means are different. This might happen if the random forest predictions are biased in finite samples. The right panel of Figure 5.1 provides a visualization of this phenomenon. In that plot, the conditional response is distributed $N(0, 16)$, while the random forest predictions are distributed $N(10, 16)$.

In either of the above scenarios, the variance of the random forest predictions estimated by Sexton and Laake (2009) and Wager et al. (2014)—in other words, the variance of the red distributions in Figure 5.1—is not the value needed to construct valid prediction intervals.

5.1.2 Quantile Regression Forests

Meinshausen (2006) proposes a method of estimating the cumulative distribution function $F_Y(y|\mathbf{X} = \mathbf{x_h}) = P(Y \leq y|\mathbf{X} = \mathbf{x_h})$ of the response variable Y conditioned on particular values $\mathbf{x_h}$ of the predictor variables $\mathbf{X}$. Unlike the work by Sexton and Laake (2009) and Wager et al. (2014), this method can be used to directly obtain prediction intervals of the response, according to Meinshausen (2006). Algorithm 4 details the method of estimating the conditional cumulative distribution function of the response variable Y—we

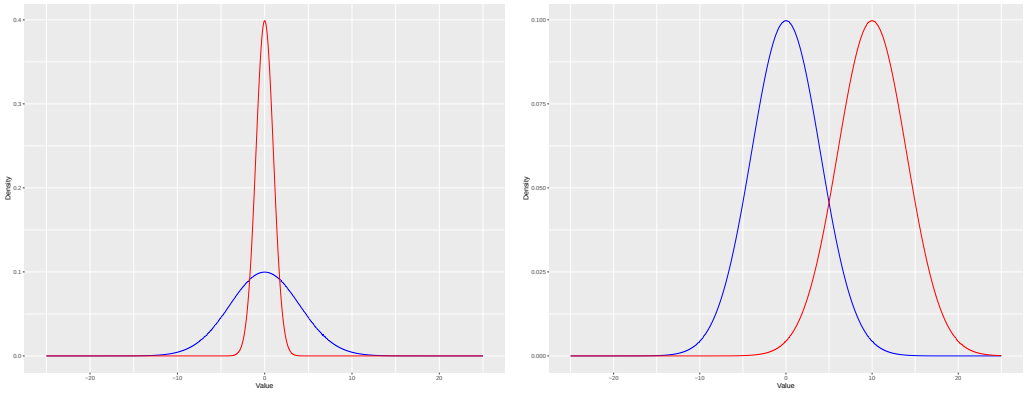

Figure 5.1: Example plots of the response variable density given a set of predictor variable values (blue) and the random forest predicted response density for observations with that set of predictor variable values (red).

denote this estimate $\hat{F}$. Note that in Equation 5.16, the weight function is indexed with respect to and calculated for each of the observations in the original sample Z, not the bootstrap sample on which the decision tree is built. Note also that the numerator of Equation 5.16 contains an indicator function that returns only a 0 or a 1. In particular, it does not count the number of times an observation appears in the terminal node.

The algorithm proposed by Meinshausen (2006) begins by identifying each decision tree's terminal node containing the test observation. For each observation used to build the random forest, it iterates through each terminal node identified in the previous step and calculates the frequency with which the observation appears at least once in the terminal node, where the frequency is evaluated with respect to the total number of observations that appear at least once in the terminal node (Equation 5.16). For each observation, the algorithm averages the frequencies across all decision trees in the random forest (Equation 5.17), and it uses the resulting "weight" to construct an empirical conditional cumulative distribution function (Equation 5.18).

Meinshausen (2006) proves that, under a set of assumptions, the empirical conditional cumulative distribution function $\hat{F}_Y(y|\mathbf{X} = \mathbf{x_h})$ obtained by this method converges in probability to the true conditional cumulative distribution function $F_Y(y|\mathbf{X} = \mathbf{x_h})$ as the sample size approaches infinity. If the empirical conditional cumulative distribution function closely approximates the true function, then a $(1 - \alpha)100\%$ prediction interval can be

Algorithm 4 Quantile Regression Forest Algorithm

procedure ESTIMATECDF(sample Z, random forest φ built on Z, test observation $\mathbf{z_h}$)
  let n denote the sample size of Z
  let B denote the number of trees in φ
  let $v_b(\mathbf{x})$ be the index of the b^{th} tree's terminal node that contains $\mathbf{x}$
  for i in $1, \ldots, n$ **do**
    for b in $1, \ldots, B$ **do**
      let $w_i^b(\mathbf{x_h})$ be defined by

$$w_i^b(\mathbf{x_h}) = \frac{\mathbb{1}(v_b(\mathbf{x_i}) = v_b(\mathbf{x_h}))}{|\{j : v_b(\mathbf{x_j}) = v_b(\mathbf{x_h})\}|} \tag{5.16}$$

    end for
    let $w_i(\mathbf{x_h})$ by defined by

$$w_i(\mathbf{x_h}) = \frac{1}{B} \sum_{b=1}^{B} w_i^b(\mathbf{x_h}) \tag{5.17}$$

  end for
  Compute the estimated conditional cumulative distribution function $\hat{F}_Y(y|\mathbf{X} = \mathbf{x_h})$ by

$$\hat{F}_Y(y|\mathbf{X} = \mathbf{x_h}) = \sum_{i=1}^{n} w_i(\mathbf{x_h}) \mathbb{1}(Y_i \leq y) \tag{5.18}$$

  return $\hat{F}_Y(y|\mathbf{X} = \mathbf{x_h})$
end procedure

obtained by subsetting the domain of the empirical conditional cumulative distribution function at appropriate quantiles. That is, the interval given by $[\hat{Q}_{\alpha_1}(\mathbf{x_h}), \hat{Q}_{\alpha_2}(\mathbf{x_h})]$ where

$$\hat{Q}_{\alpha_i}(\mathbf{x_h}) = \inf\{y : \hat{F}_Y(y|\mathbf{X} = \mathbf{x_h}) \geq \alpha_i\} \tag{5.19}$$

is a $(1-\alpha)100\%$ prediction interval if $\alpha_2 - \alpha_1 = \alpha$.

Meinshausen (2006) uses Algorithm 4 to generate a prediction interval for each observation in five different benchmark datasets. The empirical capture rates of the prediction intervals—that is, the empirically observed frequency with which the prediction intervals contain the true values—range from 90.2% to 98.6%, suggesting that more repetitions of the simulations are needed in order to verify the intervals' validity and robustness and to better understand the asymptotic properties of the empirical conditional cumulative distribution function.

5.1.3 Leave-One-Out Prediction Intervals

Steinberger and Leeb (2016) propose a prediction interval construction method in the linear regression setting based on leave-one-out residuals. They consider the scenario in which they have a sample Z of size n with p predictor variables and a continuous response variable such that the sample obeys the linear model

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{i,j} + \epsilon_i \tag{5.20}$$

for $i = 1, \ldots, n$, where ϵ_i is an error term that is independent of $\mathbf{x_i}$.

According to their method, for $i = 1, \ldots, n$ we obtain the leave-one-out residual

$$e_{(-i)} = y_i - \hat{y}_{(-i)}, \tag{5.21}$$

where $\hat{y}_{(-i)}$ denotes the predicted response of the i^{th} sample observation $\mathbf{z_i}$ by the regression model fit on $Z \backslash \{\mathbf{z_i}\}$. The leave-one-out residuals $e_{(-1)}, \ldots, e_{(-n)}$ can be used as an empirical cumulative distribution function

$$\hat{F}_e(t) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(e_{(-i)} \leq t). \tag{5.22}$$

Then, given a test observation $\mathbf{z_h}$, the $(1-\alpha)100\%$ prediction interval of the response is given by

$$\left[\hat{y}_h + \hat{Q}_{\frac{\alpha}{2}}, \hat{y}_h + \hat{Q}_{1-\frac{\alpha}{2}}\right] \tag{5.23}$$

where $\hat{y}_h = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_{h,j}$ is the value of the test response predicted by the regression model fit on Z, and $\hat{Q}_q = \inf\{t : \hat{F}_e(t) \geq q\}$ denotes the empirical q quantile of $\hat{F}_e(t)$. Steinberger and Leeb (2016) show that their prediction interval construction method for linear regression models is asymptotically valid as n approaches infinity—roughly speaking, as the sample size increases, the probability that a prediction interval constructed via this method captures the true response value approaches $1-\alpha$, as desired—under certain conditions.

As we shall see, the method proposed by Steinberger and Leeb (2016) and our proposed method are similar in form.

5.2 Direct Estimation of Global and Local Prediction Error

In the previous section, we reviewed developments in three areas of the literature that are relevant to our aim of producing a valid prediction interval construction method. Although informative, our review suggests that there is still not yet a method of constructing prediction intervals for random forests that has been shown to work in practice with finite samples. The methods reviewed in Subsection 5.1.1 estimate the standard error of random forest predictions, but they are computationally expensive, requiring two levels of bootstrapping. More importantly, the standard error of random forest predictions is not the variability measure needed to construct prediction intervals, as shown in Figure 5.1 and the accompanying text. Subsection 5.1.2 reviews quantile regression forests, which show promise as a method of prediction interval construction. However, the method has not been thoroughly tested across different data-generating models and shown empirically to yield $(1-\alpha)100\%$ prediction intervals that actually contain the true value of interest $(1-\alpha)100\%$ of the time given finite samples. Finally, Subsection 5.1.3 reviews a leave-one-out prediction interval construction procedure in the context of linear regression rather than random forests. In this section, we present our proposed methods of estimating a variability measure

that appears to be precisely the measure needed to construct valid prediction intervals for random forests. In particular, we propose two methods of estimating the mean squared prediction error of random forest predictions. Algorithms 5 and 6 outline the proposed procedures, which we call $MSPE1$ and $MSPE2$, respectively.

Note that we abandon the convention from Subsection 5.1.1 of using $\hat{\theta}_B^{RF}(\mathbf{x_h})$ to denote the random forest prediction of $\mathbf{z_h}$. This reflects our shift from considering the random forest prediction as a statistic with a sampling variability of interest to considering the random forest prediction simply as a prediction of a particular observation's response.

Algorithm 5 Variance Estimation Method 1

procedure GETMSPE1(sample Z, random forest φ)
    let n be the sample size of Z
    let B be the number of trees in φ
    let $\langle \mathbf{z} \rangle_b$ be the number of times $\mathbf{z}$ appears in the b^{th} bootstrap sample

    # FOR EACH OBSERVATION
    for i in $1, \ldots, n$ **do**

        # IDENTIFY THE TREES FOR WHICH IT IS OUT OF BAG
        let $S = \{b \in \{1, \ldots, B\} : \langle \mathbf{z_i} \rangle_b = 0\}$

        # GET ITS MEAN PREDICTED RESPONSE OVER THOSE TREES
        let $\hat{y}_{(-i)}$ be defined by

$$\hat{y}_{(-i)} = \frac{1}{|S|} \sum_{b \in S} T_b^*(\mathbf{x_i}). \tag{5.24}$$

    end for
    return

$$MSPE1 = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_{(-i)} - y_i)^2 \tag{5.25}$$

end procedure

Algorithm 6 Variance Estimation Method 2

procedure getMSPE2(sample Z, random forest φ, test observation $\mathbf{z_h}$)
  let n be the sample size of Z
  let B be the number of trees in φ
  let $\langle \mathbf{z} \rangle_b$ be the number of times $\mathbf{z}$ appears in the b^{th} bootstrap sample
  let $v_b(\mathbf{x})$ be the index of the b^{th} tree's terminal node that contains $\mathbf{x}$

  # for each decision tree
  for b in $1, \ldots, B$ **do**

    # get the sample observations that are out-of-bag cohabitants of the test observation
    let $S_b = \{i \in \{1, \ldots, n\} : \langle \mathbf{z_i} \rangle_b = 0 \wedge v_b(\mathbf{x_i}) = v_b(\mathbf{x_h})\}$

    # for each out-of-bag cohabitant
    for i in S_b **do**

      # identify the trees for which it is out of bag
      let $Q_i = \{b \in \{1, \ldots, B\} : \langle \mathbf{z_i} \rangle_b = 0\}$

      # get its mean predicted response over those trees
      let $\hat{y}_{(-i)}$ be defined by

$$\hat{y}_{(-i)} = \frac{1}{|Q_i|} \sum_{b \in Q_i} T_b^*(\mathbf{x_i}) \tag{5.26}$$

    end for
  end for

  # count how many times each observation is an out-of-bag cohabitant of the test observation
  let c_i be the total number of times i appears in $S_1, \ldots, S_B$
  return

$$MSPE2 = \frac{1}{\sum_{i=1}^{n} c_i} \sum_{i:c_i>0} c_i (\hat{y}_{(-i)} - y_i)^2 \tag{5.27}$$

end procedure

5.2.1 Introduction and Intuition

$MSPE1$ attempts to directly measure the mean squared prediction error of the random forest. It iterates through every observation in the sample used to construct the random forest. For each observation, it identifies the decision trees for which the observation is out of bag and calculates the mean prediction of the observation's response using only those decision trees (Equation 5.24) . Its estimate of the mean squared prediction error is then the sum of the squared difference between each observation's true response and its mean prediction using only those decision trees for which the observation is out of bag (Equation 5.25).

The intuition behind the claim that $MSPE1$ directly estimates the random forest's mean squared prediction error can be described as follows. If we want to estimate the mean of a random forest's squared prediction error, then we might do so directly by obtaining many observations not used to construct the random forest and calculating the sample mean squared prediction error for those observations. However, in practice, we usually use all of the observations that we have to construct the random forest, so an alternative is to iteratively treat each of our observations as unused in a procedure much like leave-one-out cross-validation; hypothetically, for each observation, we could omit it from the sample, build the random forest using this modified sample, and obtain the random forest prediction of the omitted observation's response. Calculating the prediction error of each observation in our sample in this manner and obtaining the squared mean of those errors allows us to estimate the desired value in approximately the same fashion, provided that the number of observations in the sample is sufficiently large. However, the random forest algorithm allows us to carry out the above procedure without constructing new random forests, one for each observation. Since the decision trees in the original random forest are grown on bootstrap samples, each tree is likely grown on a sample that has some observations out of bag. If there are a large number of trees, then each observation will be out of bag for many of them. We can treat the collection of trees for which a given observation is out of bag as a random forest built on the sample that has that single observation omitted. This modified procedure yields $MSPE1$.

Notice that some emphasis in the above description is given to estimating the mean squared prediction error using out-of-bag observations. This is to avoid overfitting: Recall from Algorithm 2 that the random forest algorithm constructs each decision tree in part to minimize the difference between in-

bag observations' actual responses and the decision tree's prediction of those responses. Therefore, the mean of the squared differences between in-bag observations' predicted and actual response values is likely a negatively biased estimate of the true mean squared prediction error. Using out-of-bag observations adjusts for this bias, since each decision tree is constructed without regard for out-of-bag observations; it is not built to minimize the difference between the out-of-bag observations' actual responses and its prediction of those responses.

$MSPE2$ is similar to $MSPE1$. The main difference is that $MSPE1$ calculates the squared prediction error once for *each* training observation, whereas $MSPE2$ calculates the squared prediction error for *a subset* of the training observations and may do so more than once for a given observation. In particular, $MSPE2$ calculates the squared prediction error only for the out-of-bag cohabitants of the test observation in each tree. It permits repeats, so that if an observation is an out-of-bag cohabitant of the test observation in m trees, its squared prediction error will be counted m times.

5.2.2 Estimator Properties

In this subsection, we investigate some of the properties of $MSPE1$ and $MSPE2$. We do so not by formal mathematical derivation, but by intuition and analysis of empirical simulations that use the two algorithms.

The restriction of $MSPE2$ to out-of-bag cohabitants of the test observation ensures that $MSPE2$ measures the squared prediction error of only observations that are in some sense close to the test observation, since decision trees' nodes are constructed by taking into account observations' distances from each other in the predictor space as well as the differences in their response values. This formulation is premised on the idea that observations that are close to the test observation in the predictor space are drawn from the population of interest; the prediction error of observations drawn from this population should closely resemble the prediction error of the test observation. This likely makes $MSPE2$ preferable when there is high but systematic variation in the variance of observations' response values with respect to the predictor space. However, this advantage of $MSPE2$ comes with a cost. In particular, $MSPE2$ may ignore some observations from the sample because they are not out-of-bag cohabitants of the test observation in any decision tree of the random forest. On the other hand, $MSPE1$ includes every observation from the sample in its calculation, so we expect $MSPE1$

to be more stable than $MSPE2$. This likely makes $MSPE1$ preferable when the response distribution has constant variance across the predictor space. In such a setting, all of the observations are equally useful in estimating the mean squared prediction error of a particular test observation. $MSPE1$ makes full use of this by including all of the observations in its calculation, whereas $MSPE2$ includes only the out-of-bag cohabitants of the test observation, thereby ignoring useful data. Thus, each estimator has its benefits and drawbacks.

We can observe these properties and others empirically through simulations.

Example 5.2.1. We used the MixSim package in R to generate 800 observations with 10 continuous explanatory variables and a continuous response variable. The observations are generated to consist of five clusters arranged in different ways along each of the 10 explanatory variables. Within each cluster, the observations' predictor variables are distributed uniformly, and their response variables are generated from the same normal distribution, $N(\mu_i, \sigma_i^2)$ for $i = 1, \ldots, 5$. Thus, while an observation's location in the predictor space is dispositive of the cluster to which it belongs and hence of the distribution from which its response value was sampled, the observation's response value is independent of its location in relation to other observations in the same cluster. That is, the group means are generated as a function of the predictor variables only insofar as each cluster has a different center in the predictor space.

Two datasets of 800 clustered observations with 10 continuous explanatory variables were generated. The first, denoted $SD1$, was defined by the parameters

$$\mu = (\mu_1, \ldots, \mu_5) = (1, 5, 9, 13, 17) \tag{5.28}$$
$$\sigma = (\sigma_1, \ldots, \sigma_5) = (1, 1, 1, 1, 1). \tag{5.29}$$

The second, denoted $SDMixed$, was defined by the parameters

$$\mu = (\mu_1, \ldots, \mu_5) = (1, 7, 17, 31, 49) \tag{5.30}$$
$$\sigma = (\sigma_1, \ldots, \sigma_5) = (1, 2, 3, 4, 5). \tag{5.31}$$

Notice that σ_i is the same across all clusters in $SD1$ but differs across clusters in $SDMixed$. Less obviously, the mean response value of each cluster is set so that there is at most approximately 2.5% overlap in response values between

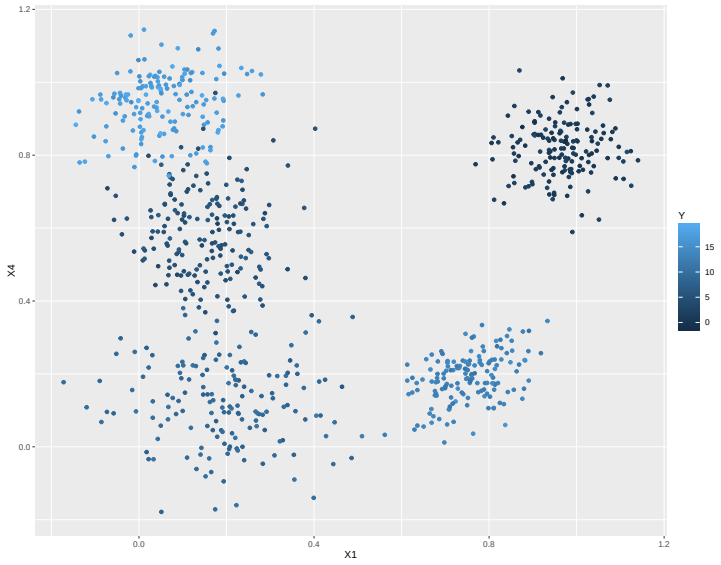

Figure 5.2: Data distributed as in $SD1$. The axes represent two of the explanatory variables, and the color of each point represents the point's response value.

clusters. Figures 5.2 and 5.3 show three dimensions of example datasets generated with the above parameters.

For each dataset, the following was performed 50 times. The data were randomly split into training and test sets, with 75% used for training. Using the randomForest R package, a random forest with 400 trees, node size of 10, and the default number of variables to subset at each split was fit to the training set. $MSPE1$ was calculated, and $MSPE2$ was calculated for each test observation.

Figure 5.4 shows the values of $MSPE1$ and $MSPE2$ generated for the simulation run on $SD1$. The figure suggests three properties of our proposed estimators.

1. It appears that $E(MSPE1) \approx E(MSPE2)$, where the expectation is taken with respect to the distribution of the population from which the data were sampled. We can see that the values of $MSPE2$ are roughly centered around the corresponding value of $MSPE1$. The distribution of $MSPE2$ and the corresponding value of $MSPE1$ even seem to generally fluctuate slightly together across repetitions, as shown by the

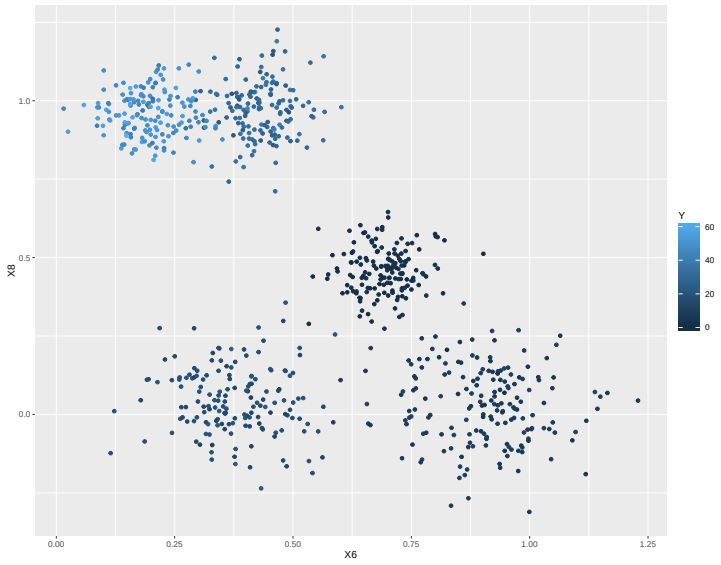

Figure 5.3: Data distributed as in *SDMixed*. The axes represent two of the explanatory variables, and the color of each point represents the point's response value.

smooth lines, which represent the loess curves fit to the $MSPE1$ and $MSPE2$ values.

2. It appears that $Var(MSPE2) > Var(MSPE1)$, as suggested by a comparison in Figure 5.4 of the variability of $MSPE1$ with the variability of $MSPE2$ for the same observation over the repetitions. This is consistent with our intuition that $MSPE1$ should yield a more stable estimate because it includes the random forest prediction error of all of the observations in the sample, rather than just a subset.

3. Both $MSPE1$ and $MSPE2$ appear to be fairly accurate, as shown by the closeness of their square-root values to the standard deviation of the clusters, $\sigma_j = 1$. It appears that both estimators are centered above 1. This is reasonable because they do not estimate just the standard deviation of the individual responses, which is how widely distributed the individual responses y_i are from their true average μ_j; instead, they estimate the deviation of the individual responses y_i from the random forest predictions of their values $\hat{y}_{(-i)}$. Since this prediction is usually not μ_j, the latter estimate tends to be greater than the former.

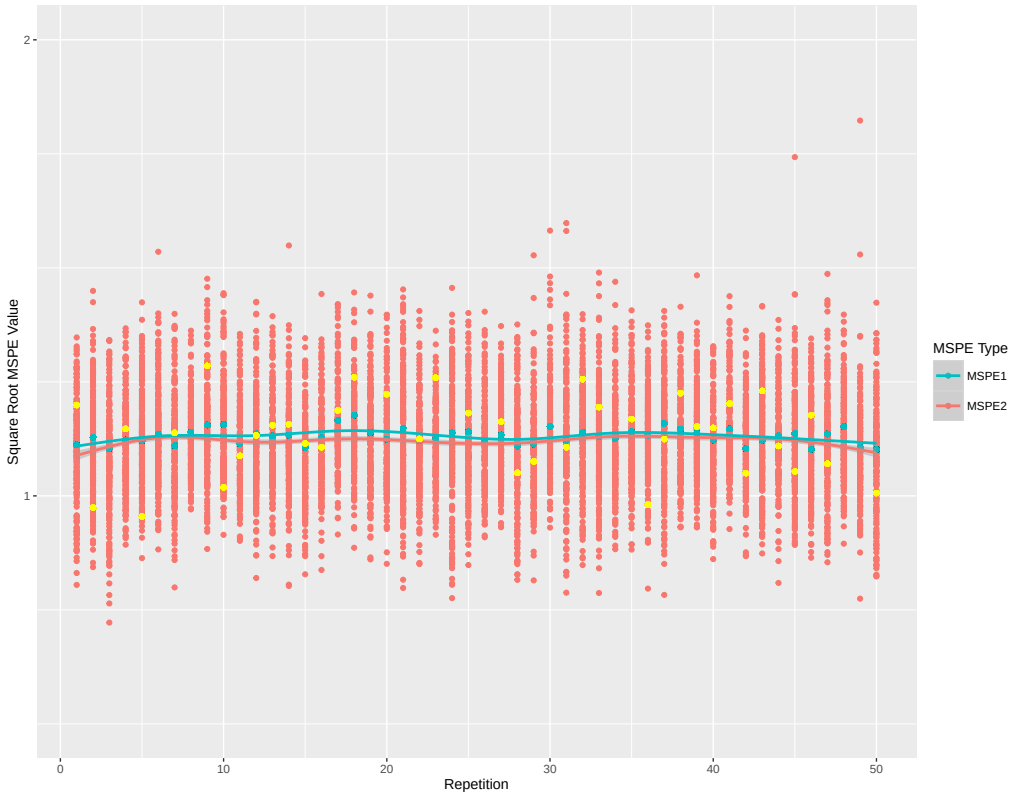

Figure 5.4: $MSPE1$ (blue) and $MSPE2$ (red) values for the $SD1$ clustered dataset. The yellow points represent $MSPE2$ values for the same observation over the repetitions for which the observation was in the test set. The smooth lines are the loess curves fit to the $MSPE1$ and $MSPE2$ values.

Figure 5.5 shows the values of $MSPE1$ and $MSPE2$ generated for the simulation run on $SDMixed$, and Figure 5.6 shows the values of $MSPE2$ by cluster. The trends shown in the figures are consistent with the three properties of our estimators listed above. In addition to these properties, the figures, particularly Figure 5.6, suggest a fourth property.

4. $MSPE2$ appears to accurately account for the local distribution of the response. That is, in general, the estimate of the random forest prediction error generated by $MSPE2$ for a given test observation correctly reflects the standard deviation of the cluster to which the test observation belongs. This can be seen by the clear stratification of the

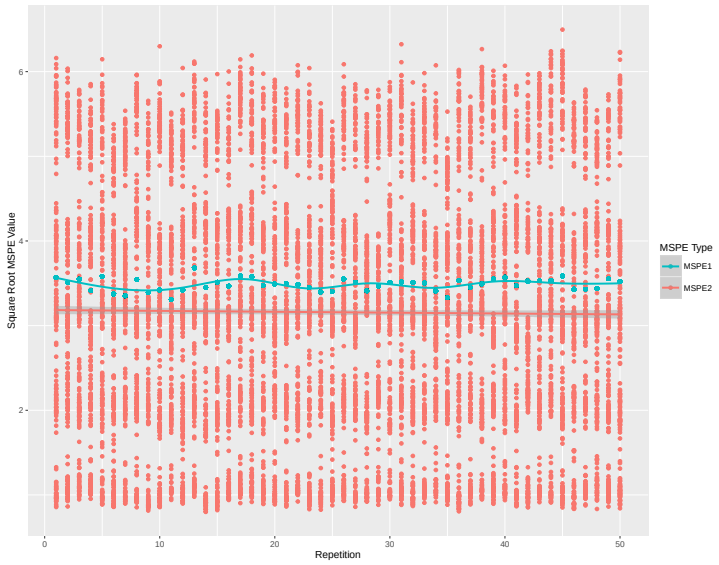

Figure 5.5: $MSPE1$ (blue) and $MSPE2$ (red) values for the $SDMixed$ clustered dataset. The smooth lines are the loess curves fit to the $MSPE1$ and $MSPE2$ values.

$MSPE2$ values by cluster in Figure 5.6. For $i = 1, \ldots, 5$, the square-root $MSPE2$ values generated for the observations belonging to cluster i are close to $\sigma_i = i$.

5.2.3 Proof of Principle: Linear Regression

Although written for random forest regression, the ideas motivating Algorithms 5 and 6 are broadly applicable. In this subsection, we apply these methods to the case of simple linear regression as a proof of principle.

Assume the simple normal error regression model from Example 2.0.2, but suppose that the data are heteroskedastic—the variance of the error term is correlated with the predictor variable. Specifically, suppose that our sample is generated from the following model:

$$y_i = 5 + 10x_i + \epsilon_i. \tag{5.32}$$

Here,

$$\epsilon_i \sim N\left(0, \left(\frac{17.5(x_i - x_{\min}}{x_{\max} - x_{\min}}\right)^2\right), \tag{5.33}$$

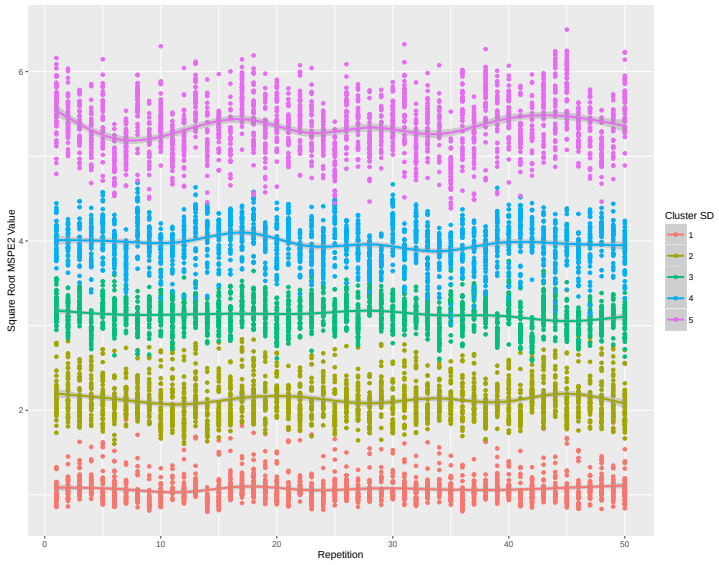

Figure 5.6: $MSPE2$ values by cluster for the $SDMixed$ clustered dataset. The smooth lines are the loess curves fit to the $MSPE2$ values by cluster.

where

$$x_{\min} = \min_{j \in \{1,\ldots,500\}} x_j \tag{5.34}$$

and

$$x_{\max} = \max_{k \in \{1,\ldots,500\}} x_k. \tag{5.35}$$

By construction, the error variance increases with the value of x. Figure 5.7 shows 500 observations generated from the model.

Suppose that we want to construct a prediction interval for the response of the test observation $\mathbf{z_h}$, whose predictor value x_h is shown in orange in Figure 5.7. Under these conditions, traditional statistical theory—the expression for a prediction interval set out in Equation 2.3—is not available to us because the data do not satisfy the necessary model assumptions. An alternative approach is to treat the observations that are close to x_h in the predictor space, colored green in Figure 5.7, as test observations: For each green observation, fit a regression model on the sample with the observation omitted. Then, predict the omitted observation's response value using the

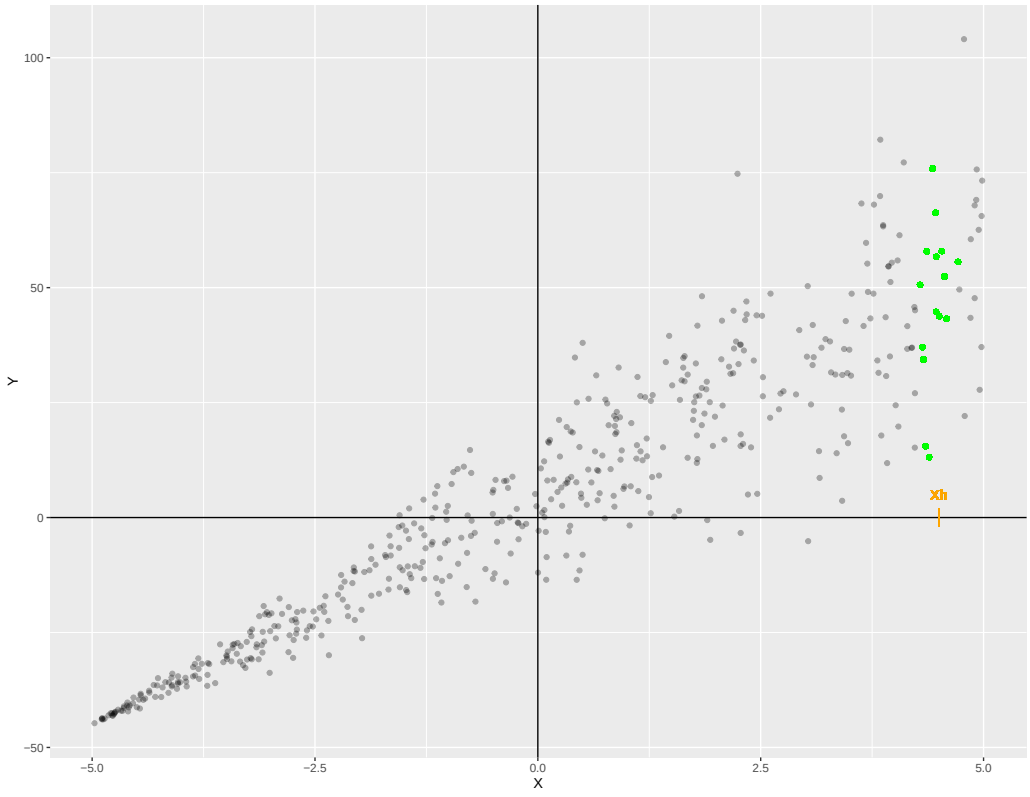

Figure 5.7: 500 observations generated from the model defined by Equations 5.32 and 5.33. The orange mark labeled x_h denotes the predictor value of the test observation. An algorithm analogous to $MSPE2$ would measure the prediction error only of observations, colored green, that are close to the test observation in the predictor space.

regression model, and measure the error of that prediction, which is the difference between the prediction and the true response value. Since the green observations are close to x_h in the predictor space, they have very similar distributions; they are essentially drawn from the same population. Thus, our estimate of the prediction error for the green observations should closely approximate the prediction error of our test observation, so we can use the observed prediction errors of the green observations to create a prediction interval for the test observation.

Notice that we would not want to perform the procedure described in the preceding paragraph for observations with, for example, $x < -4.5$ instead of the green observations, since the error terms there are distributed differently from the error terms at x_h; they are different populations. Measuring the prediction error of observations with $x < -4.5$ using the procedure described in the preceding paragraph would likely cause us to underestimate the prediction error of the test observation.

The above line of reasoning is, more or less, the idea motivating the formulation of $MSPE2$. Of course, the procedure described above does not perfectly follow the $MSPE2$ algorithm—linear regression does not have the notion of "cohabitants" that random forests do—but it is an adaptation of the algorithm to the linear regression setting that maintains the locality feature of the algorithm.

Proposition 5.2.1. The following is a *loose adaptation of* $MSPE2$ to the simple linear regression setting: Identify the S sample observations closest to the test observation x_h in the predictor space. For $s = 1, \ldots, S$, estimate the regression function using least squares on the entire sample except the s^{th} closest observation, then predict the s^{th} closest observation's response using that regression function. The mean squared difference between the S closest observations' true response values and the response values predicted by the corresponding regression functions is $MSPE2$. After obtaining $MSPE2$, estimate the regression function using least squares on the entire sample. Use the regression function to obtain the predicted response $\hat{y}_h$ of the test observation, and obtain $(1-\alpha)100\%$ prediction interval bounds by $\hat{y} \pm t_{(\frac{\alpha}{2}, S-1)}\sqrt{MSPE2}$.

We can empirically verify that Proposition 5.2.1 works. After obtaining 500 observations generated from the linear model, we split the dataset so that 75% of the observations are used to train the regression function, and the remaining 25% are held as test observations. We then followed Proposition

5.2.1 with $S = 10$ to obtain a 95% prediction interval for each test observation. Then, we calculated the percentage of the prediction intervals that actually contained the true response value. We repeated the process 1000 times and averaged the empirical capture rate over the 1000 repetitions. As the first row of Table 5.1 indicates, the empirical capture rate for the loose adaptation of $MSPE2$, denoted **L MSPE2**, was 0.949, almost exactly the desired rate.

Thus, there is evidence to suggest that Proposition 5.2.1 is able to generate valid prediction intervals in the simple linear regression setting even in the presence of heteroskedasticity. In comparison, the traditional method of constructing prediction intervals for simple linear regression does not yield valid intervals; as Table 5.1 indicates, the empirical capture rate of the traditional approach was 0.932.

Suppose next that the model generating the data is homoskedastic—the error variance is constant. Figure 5.8 shows 500 observations generated from such a model. If we wanted to construct a prediction interval for a test observation with predictor value x_h (labeled in orange in Figure 5.8) under these conditions, there is no reason to limit ourselves only to the observations, colored green in Figure 5.8, closest to x_h. All of the observations are equally useful in estimating prediction error because their error terms all come from the same population. Therefore, to construct a prediction interval, we can follow Proposition 5.2.1 but perform it for every observation in our sample rather than just the few that are closest to x_h. That is, set $S = n$, the number of observations in the sample.

The above line of reasoning is essentially the idea motivating the formulation of $MSPE1$. As in our discussion of the heteroskedastic dataset, the procedure described above does not perfectly follow the $MSPE1$ algorithm, but it is an adaptation of the algorithm to the linear setting.

Proposition 5.2.2. The following is a *loose adaptation of* $MSPE1$ to the simple linear regression setting: For $i = 1, \ldots, n$, where n is the number of observations in the sample, estimate the regression function using least squares on the entire sample except the i^{th} sample observation, then predict the i^{th} sample observation's response using that regression function. The mean squared difference between the sample observations' true response values and the response values predicted by the corresponding regression functions is $MSPE1$. After obtaining $MSPE1$, estimate the regression function using least squares on the entire sample. Use that regression function to ob-

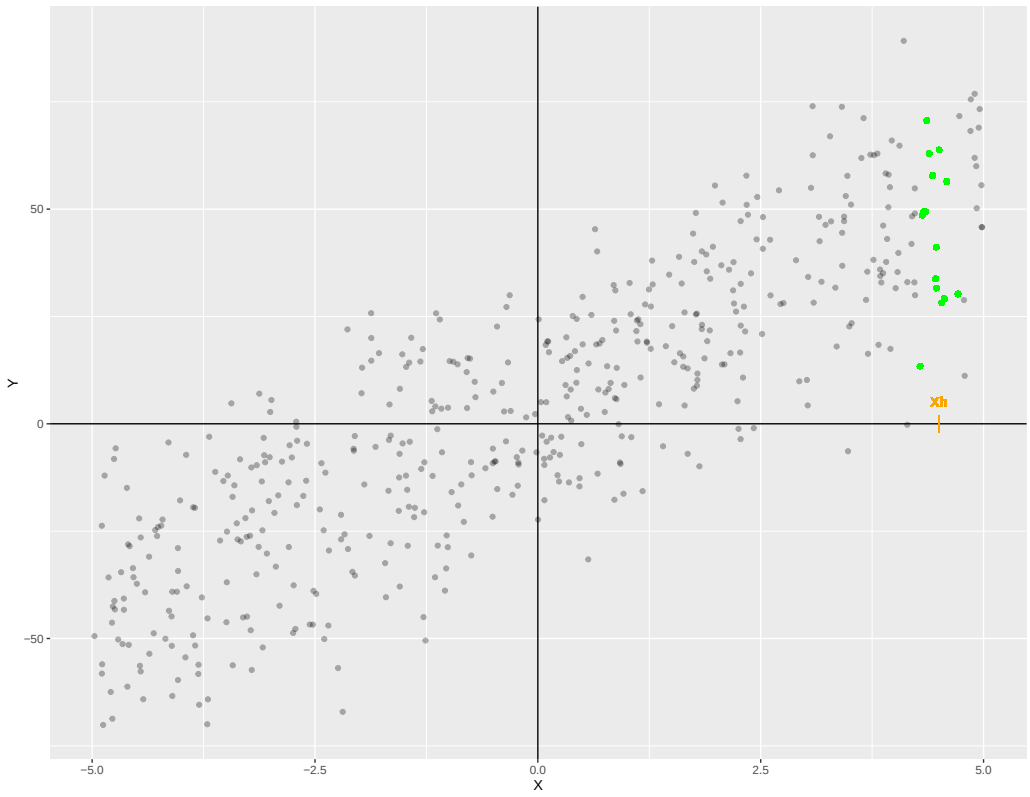

Figure 5.8: 500 observations generated from the model defined by Equations 5.32 and with constant error variance. The orange mark labeled x_h denotes the predictor value of the test observation. An algorithm analogous to $MSPE1$ would measure the prediction error of all observations, not just the ones, colored green, that are close to the test observation.

tain the predicted response $\hat{y}_h$ of the test observation, and obtain $(1-\alpha)100\%$ prediction interval bounds by $\hat{y} \pm t_{(\frac{\alpha}{2},n-2)}\sqrt{MSPE1}$.

Notice that, in addition to resembling the $MSPE1$ algorithm, Proposition 5.2.2 is essentially the prediction interval construction method proposed by Steinberger and Leeb (2016). The only difference is that Steinberger and Leeb (2016) uses the quantiles of the leave-one-out residuals to directly create the prediction intervals.

We can empirically verify that Proposition 5.2.2 works by performing a simulation similar to the one used to verify Proposition 5.2.1. The only differences are that the data in this simulation are homoskedastic, and the number of repetitions was increased. Specifically, we performed 6000 repetitions and obtained an empirical capture rate of 0.950 for our 95% prediction intervals, as desired. (This result is not shown in Table 5.1.) The empirical capture rate obtained for 95% prediction intervals generated using the traditional Equation 2.3 under these simulation conditions was also 0.950, as expected. In fact, we suspect that Proposition 5.2.2 and the traditional Equation 2.3 fundamentally operate in the same way and should yield the same results when applied under the same conditions, though this has not been investigated in great detail.

As mentioned in both the discussion of the heteroskedastic data and the discussion of the homoskedastic data, Propositions 5.2.1 and 5.2.2 are not identical to the $MSPE1$ and $MSPE2$ algorithms. Unlike their loosely adapted counterparts, the latter two algorithms both use bootstrapping. Moreover, $MSPE2$ does not consider a pre-determined number of closest out-of-bag sample observations, unlike Proposition 5.2.1. Therefore, to improve the relevance of this proof of principle, we introduce the following adaptations that more closely resemble the original $MSPE1$ and $MSPE2$ algorithms.

Proposition 5.2.3. The following is a *close adaptation of* $MSPE1$ to the simple linear regression setting: Obtain B bootstrap samples of the original sample. Estimate a regression function using least squares on each bootstrap sample. For each observation in the original sample, identify the regression functions for which the observation is out of bag, then predict the observation's response as the mean prediction of those regression functions. The mean squared difference between the n observations' true response values and the response values predicted by the regression functions for which the observations are out of bag is $MSPE1$. After obtaining $MSPE1$, average

S	L MSPE1	C MSPE1	L MSPE2	C MSPE2	Traditional
10	0.934	0.932	0.949	0.946	0.932
2	0.934	0.933	0.994	0.911	0.930

Table 5.1: Empirical capture rates of 95% prediction intervals constructed using adapted versions of $MSPE1$ and $MSPE2$ over 1000 repetitions. The data are heteroskedastic and described by Equations 5.32 and 5.33.

the predictions of the test observation's response across all B regression functions to obtain the predicted response $\hat{y}_h$ of the test observation, and obtain $(1-\alpha)100\%$ prediction interval bounds by $\hat{y} \pm t_{(\frac{\alpha}{2},n-2)}\sqrt{MSPE1}$.

Proposition 5.2.4. The following is a *close adaptation of* $MSPE2$ to the simple linear regression setting: Obtain B bootstrap samples of the original sample. Estimate a regression function using least squares on each bootstrap sample. For $b = 1, \ldots, B$, identify the S observations in the b^{th} bootstrap sample that are closest to the test observation in the predictor space. For each of the SB observations, identify the regression functions for which the observation is out of bag, then predict the observation's response as the mean prediction of those regression functions. The mean squared difference between the SB observations' true response values and the response values predicted by the regression functions for which the observations are out of bag is $MSPE2$. After obtaining $MSPE2$, average the predictions of the test observation's response across all B regression functions to obtain the predicted response $\hat{y}_h$ of the test observation, and obtain $(1-\alpha)100\%$ prediction interval bounds by $\hat{y} \pm t_{(\frac{\alpha}{2},SB-1)}\sqrt{MSPE2}$.

Table 5.1 shows the capture rates of the prediction intervals constructed using the loose (**L**) and close (**C**) adaptations of $MSPE1$ and $MSPE2$, as well as the prediction intervals constructed using the traditional interval formulation from Equation 2.3, on data generated from the model defined by Equations 5.32 and 5.33 over 1000 repetitions. Where relevant, $B = 500$ was the number of bootstrap samples, and the number S of closest observations to the test observation is listed in the table.

Observe that, when $S = 10$, the prediction intervals constructed using both the loose and the close adaptations of $MSPE2$ returned capture rates of 0.949 and 0.946, respectively, suggesting that the algorithms are correctly measuring the prediction error. Moreover, the prediction intervals

constructed using the adaptations of the $MSPE1$ algorithm had empirical capture rates about equal to the capture rates of prediction intervals constructed using the traditional method, which supports our hypothesis that the adaptations of $MSPE1$ and the traditional method are fundamentally similar.

Finally, observe that the loose and close adaptations of $MSPE2$ respectively returned higher than desired and lower than desired capture rates when $S = 2$. We are uncertain why this occurs; it is a potential subject of further investigation into the behavior of our estimators.

Chapter 6

Random Forest Prediction Interval Construction

Because they directly estimate a random forest's prediction error, $MSPE1$ and $MSPE2$ lend themselves naturally to prediction interval construction.

Proposition 6.0.1. Let φ be a random forest and $\mathbf{z_h}$ be an observation with response value y_h unknown. A 95% prediction interval for y_h is given by

$$\varphi(\mathbf{x_h}) \pm 2\sqrt{MSPE1} \tag{6.1}$$

and by

$$\varphi(\mathbf{x_h}) \pm 2\sqrt{MSPE2}. \tag{6.2}$$

Empirical simulation results suggest that our proposed methods of estimating variance work well. We ran simulations on the following datasets:

- Boston: Obtained from the MASS package in R, this dataset consists of 506 observations of 14 variables. Each observation is a town, and the variables include features such as the town's per capita crime rate, the average number of rooms per dwelling in the town, whether the town bounds the Charles River, and nitrogen oxides concentration in the town. The response variable is the median value of owner-occupied homes in thousands of dollars.

- Forest Fires: Obtained from the University of California, Irvine Machine Learning Repository, this dataset consists of 517 observations of 13 variables. Each observation is a forest fire event in the northeast

region of Portugal. The variables include features such as the spatial coordinates of the fire, the month of the year, the day of the week, the relative humidity, and the wind speed. The response variable is the burned area of the forest resulting from the fire, in hectares. The response variable was transformed using the log transform: $\ln(y+1)$.

- Ozone: Obtained from the mlbench package in R, this dataset consists of 366 observations of 13 variables. Each observation is one day. The variables include features such as the date; the humidity at Los Angeles International Airport (LAX); the pressure gradient from LAX to Daggett, California; the wind speed at LAX; and the temperature at El Monte, California. The response variable is the visibility, in miles, measured at LAX. Some of the 366 observations contain missing data; those observations were removed before the simulation.

- MPG: Obtained from the University of California, Irvine Machine Learning Repository, this dataset consists of 398 observations of 8 variables. Each observation is a vehicle model. The variables include features such as the weight, the model year, the origin, the displacement, and the number of cylinders. The response variable is the city-cycle fuel consumption in miles per gallon.

- Clustered1: This dataset was generated in the same manner as the datasets from Example 5.2.1. It consists of 500 observations with 10 predictor variables. The data were separated into four clusters. The response variable means by cluster are

$$\mu = (\mu_1, \ldots, \mu_4) = (0, 6, 16, 30), \tag{6.3}$$

  and the standard deviations by cluster are

$$\sigma = (\sigma_1, \ldots, \sigma_4) = (1, 2, 3, 4). \tag{6.4}$$

- Clustered2: This dataset was generated in the same manner as the datasets from Example 5.2.1. It consists of 500 observations with 10 predictor variables. The data were separated into four clusters. The response variable means by cluster are

$$\mu = (\mu_1, \ldots, \mu_4) = (0, 100, 1000, 5000), \tag{6.5}$$

and the standard deviations by cluster are

$$\sigma = (\sigma_1, \ldots, \sigma_4) = (10, 10, 10, 10). \tag{6.6}$$

- Clustered3: This dataset was generated in the same manner as the datasets from Example 5.2.1. It consists of 500 observations with 10 predictor variables. The data were separated into four clusters. The response variable means by cluster are

$$\mu = (\mu_1, \ldots, \mu_4) = (0, 100, 1000, 5000), \tag{6.7}$$

and the standard deviations by cluster are

$$\sigma = (\sigma_1, \ldots, \sigma_4) = (1, 40, 400, 2000). \tag{6.8}$$

For each dataset, we randomly split the data so that 75% of the observations were used for training and 25% were used as test observations. We built a random forest with 500 decision trees on the training set. The stopping criterion—that is, the minimum number of observations needed for a node to be split—was set to 15. One-third of the predictor variables were available at each node split. After building the random forest, we followed Algorithms 5 and 6 to obtain values of $MSPE1$ and $MSPE2$ for each test observation. We then constructed a 95% prediction interval for each test observation using Equations 6.1 and 6.2. After constructing a prediction interval for each test observation, we calculated the percentage of prediction intervals that actually contained the true response value. We repeated the entire process—from the initial splitting of the data into training and test sets to the end—1000 times and averaged the empirical capture rate over the 1000 repetitions. Table 6.1 displays the empirical capture rates obtained from the simulations.

From these results, we observe the following:

1. The empirical capture rates of the prediction intervals constructed from both estimators are around 95% for each of the standard, benchmark datasets: Boston, Forest Fires, Ozone, and MPG. This suggests that both of our proposed estimators seem to have yielded valid 95% prediction intervals for those datasets. $MSPE1$ yielded particularly accurate prediction intervals, with capture rates of 0.951, 0.951, 0.950, and 0.950 for the benchmark datasets, respectively.

Data	MSPE1	MSPE2
Boston	0.951	0.965
Forest Fires	0.951	0.948
Ozone	0.950	0.973
MPG	0.950	0.956
Clustered1	0.936	0.947
Clustered2	0.961	0.972
Clustered3	0.919	0.972

Table 6.1: Empirical capture rates of 95% prediction intervals constructed using $MSPE1$ and $MSPE2$ over 1000 repetitions on various datasets.

2. The prediction intervals generated from $MSPE1$ captured at a lower rate than desired for the Clustered1 dataset, while the prediction intervals generated from $MSPE2$ captured more or less at the desired rate. We suspect that this is because the localizing feature of the $MSPE2$ algorithm makes the prediction intervals generated from it narrowly tailored to measure the variability of the random forest's error in predicting the particular test observation in question, which differs across clusters because each cluster has a different standard deviation. On the other hand, we suspect that the $MSPE1$ algorithm captured at a lower rate because of its "averaging effect." By construction, $MSPE1$ tends to center around the overall mean squared prediction error across all clusters. Thus, the prediction intervals yielded by $MSPE1$ tend to be too wide for the clusters with low standard deviations and too narrow for the clusters with high standard deviations. In particular, they tend to be wider than necessary to yield a capture rate of 95% for observations from the clusters with low standard deviations, and they tend to be narrower than necessary to yield a capture rate of 95% for observations from the clusters with high standard deviations. But because the distribution of prediction errors is tapered at the ends, narrowing a 95% prediction interval by a set number of units will cause a greater decrease in capture rate than the increase in capture rate caused by widening a 95% prediction interval by the same number of units. Therefore, the "averaging effect" of $MSPE1$ tends to yield prediction intervals that capture below the desired rate if the response variable's variance

systematically changes across the predictor space.

3. Both $MSPE1$ and $MSPE2$ did not capture at the desired rate for the Clustered2 and Clustered3 datasets. We suspect that this is a general trend for datasets that are characterized by extremely pronounced shifts in the mean of the response variable's distribution across the predictor space. To see why, consider the fringes of each cluster in the predictor space. Because an individual decision tree is built on a bootstrap sample, it does not have the benefit of viewing all of the observations in the original sample. In particular, because some observations at the fringes of the clusters are likely out of bag for the decision tree, the decision tree may not have an accurate representation of where the boundaries between clusters are in the predictor space. Therefore, it is possible for the decision tree to create a terminal node in the predictor space that includes two different clusters—if, for example, it happens to be the case that all of the observations from one of the clusters in the terminal node are out of bag, so that the WSS is relatively small. When this occurs, the values of $MSPE1$ and $MSPE2$ will be overinflated, since the difference in cluster means is so dramatic—at the most extreme, the means are 5000 apart—relative to the cluster standard deviations—which are as small as 40 and 10—that the predictions of out-of-bag observations in the terminal node will have large errors. This phenomenon, we suspect, will tend to cause inflated capture rates, which we see in the results presented in Table 6.1 and in the boxplots of prediction interval bounds shown in Figures 6.1 and 6.2. There is one exception, however: The prediction intervals generated from $MSPE1$ on the Clustered3 dataset had a deflated capture rate. We suspect that this occurred because the the response variable in the Clustered3 dataset, unlike in the Clustered2 dataset, has dramatically different standard deviations by cluster. Thus, the "averaging effect" of $MSPE1$ described in the previous item takes effect and, we suspect, dominates the "fringe effect" described in this item.

The above discussion raises the question of whether the random forest model is suitable for certain types of datasets. Specifically, it suggests the somewhat counterintuitive idea that the random forest model may not be appropriate for data with discrete, stepwise shifts in response variable distribution across the predictor space. This seems counterintuitive because random forests are generally thought to require few assumptions about the

data; if anything, they themselves are commonly conceptualized as stepwise models that should perform especially well on this type of data. However, our results suggest that problems arise in the regions of the predictor space where such discrete shifts in the response variable distribution occur. Because the random forest cannot perfectly delineate the boundary between the two different response variable distributions, its predictions at such boundary regions tend to be much less accurate.

This idea, in turn, raises the methodological problem of determining how to assess the empirical validity of a prediction interval construction algorithm. Specifically, it is unclear whether or not we should require our algorithms, as a condition for their validity, to construct prediction intervals that obtain empirically the desired capture rates for datasets such as Clustered1, Clustered2, and Clustered3, especially if we conclude that the random forest is simply not an appropriate model for such data. Determining the extent to which poor empirical results should be attributed to the proposed estimators as opposed to the model and its fit is a difficult task, but it has important implications for the way we discuss and evaluate statistical tools.

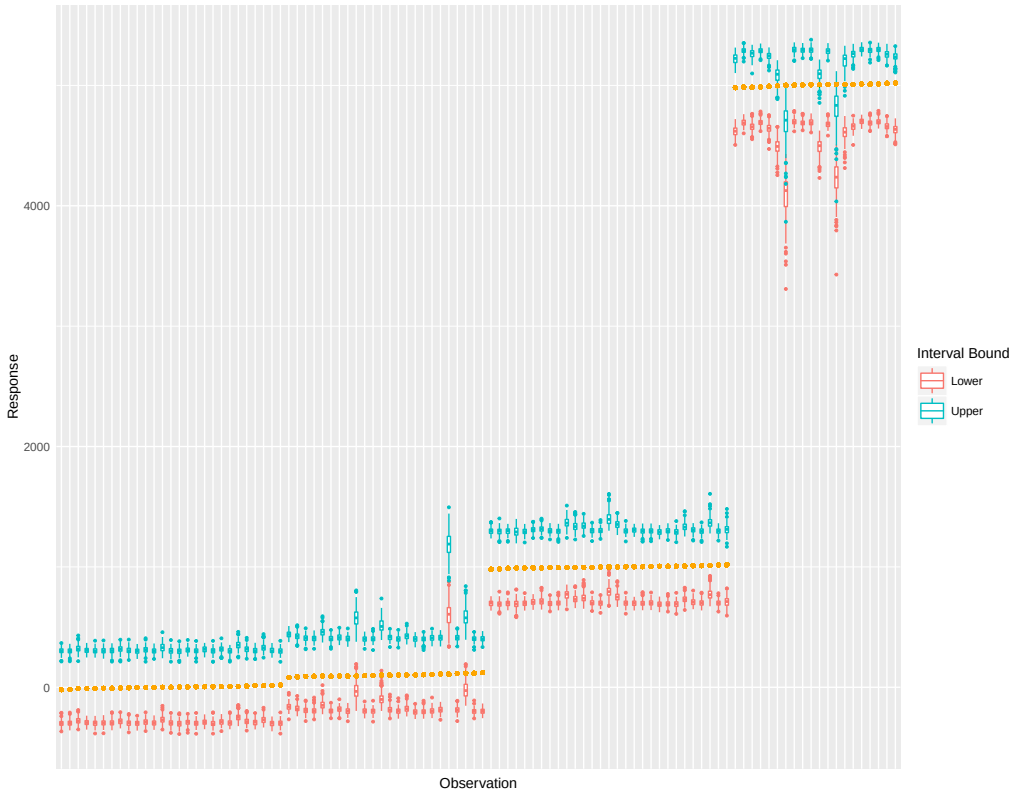

Figure 6.1: Boxplots of bounds of prediction intervals generated using Algorithm 5 ($MSPE1$) on the Clustered2 dataset. Observe that the distance between the lower and upper bounds of the prediction intervals is generally too wide, particularly given the scaling of the y-axis. We suspect that this is due to the "fringe effect." The capture rate for this simulation was 0.961.

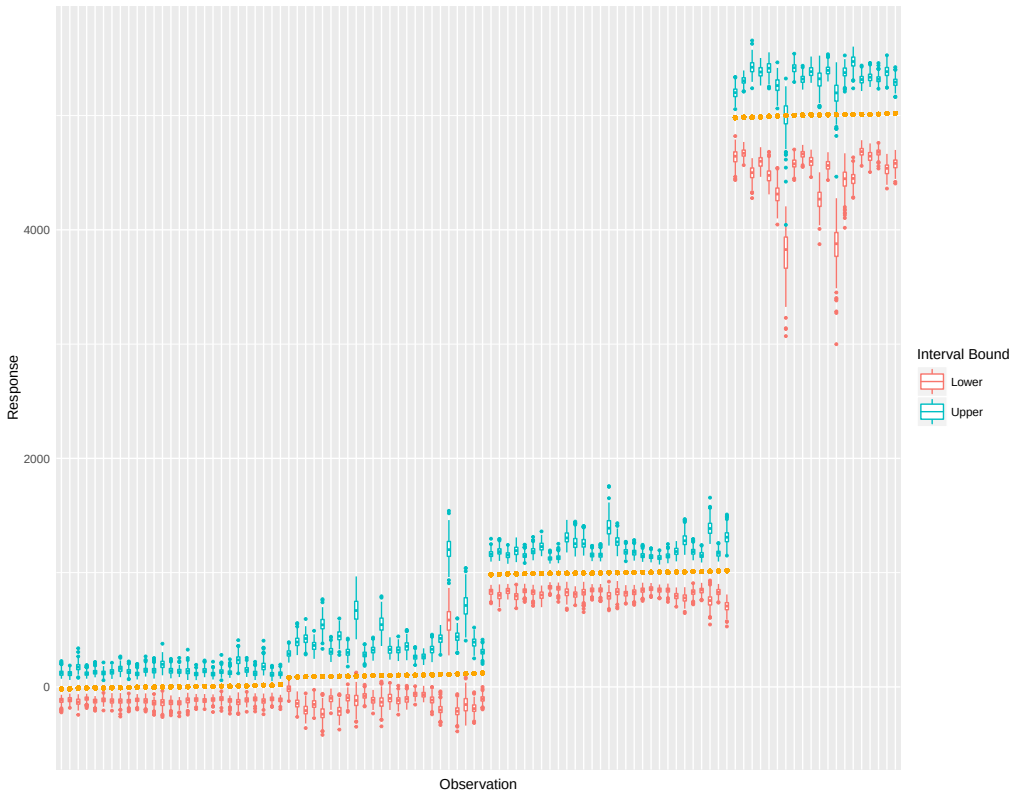

Figure 6.2: Boxplots of bounds of prediction intervals generated using Algorithm 6 ($MSPE2$) on the Clustered2 dataset. Observe that the distance between the lower and upper bounds of the prediction intervals is generally too wide, though it is slightly narrower than in Figure 6.1. We suspect that this is due to the "fringe effect." The capture rate for this simulation was 0.972.

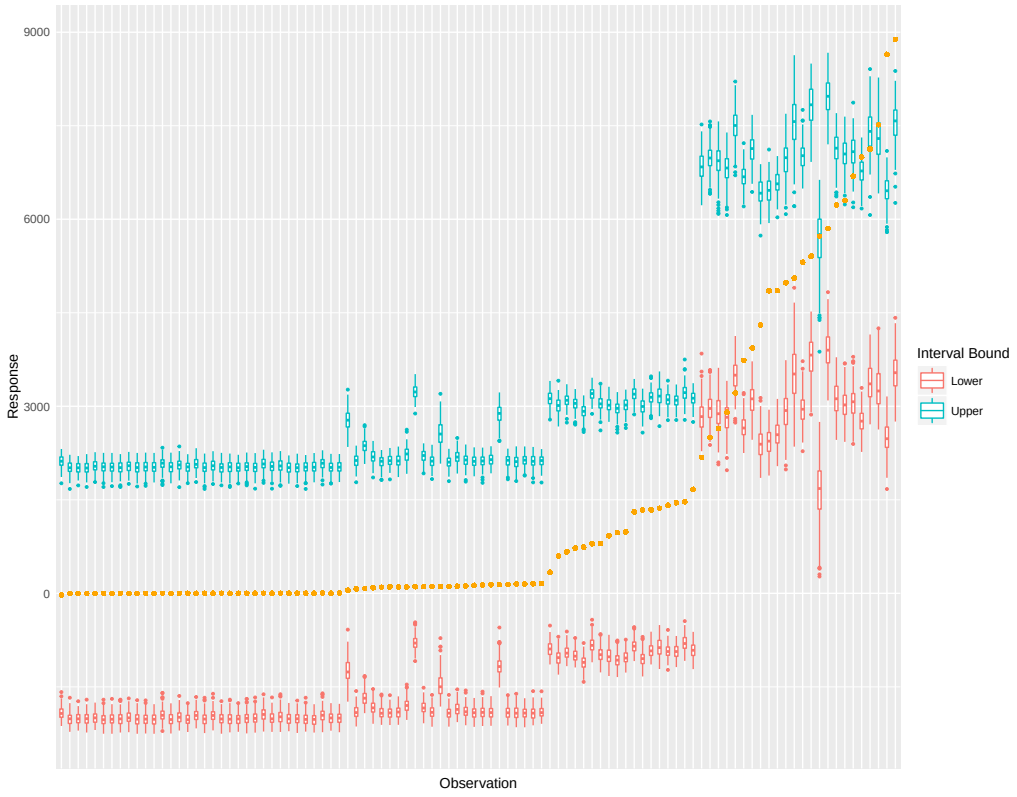

Figure 6.3: Boxplots of bounds of prediction intervals generated using Algorithm 5 ($MSPE1$) on the Clustered3 dataset. Observe that the distance between the lower and upper bounds of the prediction intervals is roughly constant across clusters, as expected. The capture rate for this simulation was 0.919. We suspect that this is because the "averaging effect" overshadowed the "fringe effect."

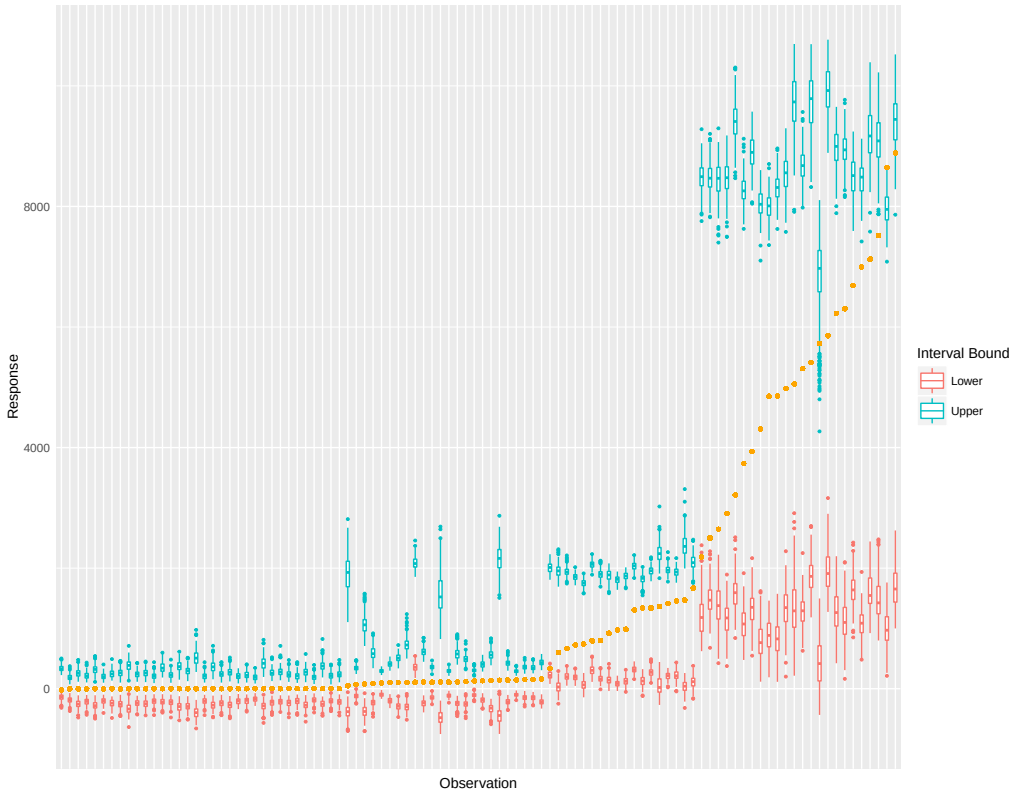

Figure 6.4: Boxplots of bounds of prediction intervals generated using Algorithm 6 ($MSPE2$) on the Clustered3 dataset. Observe that the average distance between the lower and upper bounds of the prediction intervals is different for each cluster, as expected. The capture rate for this simulation was 0.972. We suspect that this is due to the "fringe effect."

Chapter 7

Conclusion

Despite recent advancements in the literature, random forest predictions of the value of continuous response variables are still generally confined to point estimation, which is limited in utility because it provides only a single value as the prediction without an estimate of the variability associated with the prediction. We propose two estimators of random forest prediction error, $MSPE1$ and $MSPE2$, that can be used to construct prediction intervals, which quantify the uncertainty associated with a random forest prediction. Based on our intuition and analysis of empirical simulations, we identified important properties of these estimators. Most notably, both $MSPE1$ and $MSPE2$ generally appear to directly estimate the measure of variability needed to construct valid prediction intervals. However, they do so in different ways: $MSPE1$ estimates the random forest prediction error over all observations, whereas $MSPE2$ does so only for a particular subset of observations. We then used our proposed estimators to construct prediction intervals for random forests on a variety of datasets, including standard, benchmark datasets and artificial, clustered datasets designed to help us evaluate the estimators. We found that the prediction intervals generated by both $MSPE1$ and $MSPE2$ appear to be valid on the benchmark datasets, but we observed that they did not perform well on the clustered datasets. The results obtained from the clustered datasets raised important substantive questions about the structure and suitability of random forests, as well methodological questions about the way we empirically evaluate our estimators.

The results obtained so far, particularly on the standard, benchmark datasets, are promising. However, additional research remains to be done. Future work on the subject might include comparing our prediction inter-

val construction method with the method proposed by Meinshausen (2006), gaining a better understanding of the robustness of the estimators to a variety of conditions, determining the appropriate multiplier of $MSPE1$ and $MSPE2$ to use to construct prediction intervals, and proving the theoretical properties of the estimators.

Bibliography

Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL `http://dx.doi.org/10.1023/A:1010933404324`.

John Bryan. Developing inference: Frameworks for random forests using bag of little bootstraps & related methods. Senior Thesis, Pomona College, April 2016.

B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. doi: 10.1214/aos/1176344552.

Bradley Efron. Nonparametric standard errors and confidence intervals. *The Canadian Journal of Statistics*, 9(2):139–158, 1981.

Bradley Efron. Jackknife-after-bootstrap standard errors and influence functions. *Journal of the Royal Statistical Society, Series B (Methodological)*, 54(1):83–127, 1992.

Bradley Efron. Estimation and accuracy after model selection. *Journal of the American Statistical Association*, 2013.

Bradley Efron and Robert Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1994.

A. D. Gordon, L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. *Biometrics*, 40(3):874, 1984. doi: 10.2307/2530946.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer, 2013.

Michael H. Kutner, Christopher J. Nachtsheim, John Neter, and William Li. *Applied Linear Statistical Models*. McGraw-Hill/Irwin, 5th ed. edition, 2005.

Nicolai Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7:984–999, Jun 2006.

Erwan Scornet, Gerard Biau, and Jean-Philippe Vert. Consistency of random forests. *The Annals of Statistics*, 43(4):1716?1741, 2015. doi: 10.1214/15-AOS1321.

Joseph Sexton and Petter Laake. Standard errors for bagged and random forest estimators. *Computational Statistics and Data Analysis*, 2009.

Lukas Steinberger and Hannes Leeb. Leave-one-out prediction intervals in linear regression models with many variables. *ArXiv: 1602.05801*, Feb 2016.

Stefan Wager. Asymptotic theory for random forests. *arXiv:1405.0352*, May 2016.

Stefan Wager, Trevor Hastie, and Bradley Efron. Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *Journal of Machine Learning Research*, 15:1625–1651, May 2014.