2018

Random forest robustness, variable importance, and tree aggregation

Andrew Sage
Iowa State University

Random forest robustness, variable importance, and tree aggregation

by

Andrew John Sage

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Statistics

Program of Study Committee:
Ulrike Genschel, Co-major Professor
Dan Nettleton, Co-major Professor
Daniel Nordman
Craig Ogilvie
Vivekananda Roy

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2018

DEDICATION

To my family and friends who encourage me with their words and inspire me with their example.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS

I would like to acknowledge several members of the Iowa State University faculty who have been influential in my studies and research. I am especially grateful to my Co-major Professors Ulrike Genschel and Dan Nettleton for their valuable advice, which extends far beyond the writing of this dissertation. I would like to thank Professors Cinzia Cervato and Craig Ogilvie for giving me the opportunity to apply my research in an important and meaningful context. I am grateful to Professors Ogilvie, Daniel Nordman, and Vivekananda Roy for serving on my committee and to Professors Nordman and Roy, whose courses increased my understanding and appreciation of statistical theory. I would also like to thank Professor Stephen Vardeman, whose course in statistical machine learning greatly enhanced my understanding of the topics at the heart of this dissertation.

I am also grateful to those who were influential in my learning prior to my studies at Iowa State. Professor Stephen Wright of Miami University first introduced me to the excitement of performing original research. Professors Jim Hartman, Pam Pierce, and John Ramsay of The College of Wooster helped inspire my interest in learning mathematics and statistics as an undergraduate.

Finally, I would like to acknowledge the students I have had the pleasure of teaching at Iowa State University, Miami University, and Bloomfield High School. Working with them has provided me with energy that fuels my research.

ABSTRACT

Random forest methodology is a nonparametric, machine learning approach capable of strong performance in regression and classification problems involving complex datasets. In addition to making predictions, random forests can be used to assess the relative importance of explanatory variables. In this dissertation, we explore three topics related to random forests: tree aggregation, variable importance, and robustness. In Chapter 2, we show that the method of tree aggregation used in one popular random forest implementation can lead to biased class probability estimates and that it is often beneficial to combine the tree partitioning algorithm used in one implementation with the aggregation scheme used in another. In Chapter 3, we show that imputing missing values proir to assessing variable importance often leads to inaccurate variable importance measures. Using simulation studies, we investigate the impact on variable importance of six random-forest-based imputation techniques and find that some techniques are prone to overestimating the importance of variables whose values have been imputed, while other techniques tend to underestimate the importance of such variables. In Chapter 4, we propose a new robust approach for random forest regression. Adapted from a popular approach used in polynomial regression, our method uses residual analysis to modify the weights associated with training cases in random forest predictions, so that outlying training cases have less impact. We show, using simulation studies, that this approach outperforms existing robust techniques on noisy, contaminated datasets.

CHAPTER 1. GENERAL INTRODUCTION

1.1 Background

Since its creation by Breiman (2001), random forest methodology has emerged as a powerful nonparametric, machine learning approach for regression and classification. Based on ensembles of decision trees, random forests routinely handle nonlinear relationships and interactions, making them a popular choice for predictions involving complex datasets. Random forests are implemented in several `R` (R Core Team, 2016) packages including `randomForest` (Liaw and Wiener, 2002), `randomForestSRC` (Ishwaran et al., 2008; Ishwaran and Kogalur, 2007, 2016), and `party` (Hothorn et al., 2006a; Strobl et al., 2007, 2008).

Random forests consist of many individual classification or regression trees (Breiman et al., 1984), which are grown by recursively performing binary splits on training data. Individual decision trees are low bias, high variance predictors. Averaging predictions across a large number of trees, as is done in a random forest, reduces the variance of the resulting predictor, while maintaining low bias. In order to ensure that trees differ substantially enough to achieve this variance reduction, randomness is introduced in the tree-growing process in the following two ways:

1. Each tree is grown from a different bootstrap sample of the training data.

2. For each split, a randomly selected subset of explanatory variables is considered, rather than considering all explanatory variables.

Let $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$, represent a set of training data, which are the result of n draws from some unknown distribution and let $(\boldsymbol{x}, y)$ represent a new case from the same

distribution. In a regression setting, a random forest provides an estimate of the conditional mean, $E(Y|\boldsymbol{x})$. In a classification problem, a random forest can be used to predict the most likely response category, or to estimate $P(Y = c|\boldsymbol{x}) = E\left[\mathbb{1}(Y = c)|\boldsymbol{x}\right]$, the conditional probability that Y takes on response category c. Note that $\mathbb{1}(\cdot)$ represents a generic indicator function.

Meinshausen (2006) showed that a random forest prediction for a new case can be expressed as a weighted average of $y_1, y_2, \ldots y_n$. This insight provides a method for estimating quantiles of the conditional distribution of Y given $\boldsymbol{x}$. Such estimates can be obtained using the `quanregForest` package (Meinshausen, 2016) in `R`.

1.2 Overview

1.2.1 Partitioning and Tree Aggregation

Several techniques have been proposed for determining splits in training data when growing decision trees. The `randomForest` and `randomForestSRC` packages implement the Classification and Regression Tree (CART) algorithm (Breiman et al., 1984). Although widely popular, this approach has been shown to favor splits based on explanatory variables that take on many different values, and potentially fails to take advantage of information provided by categorical predictors (Hothorn et al., 2004, 2006a,b). The `party` package implements *conditional inference trees* (Hothorn et al., 2006b), in which splits are determined using permutation tests. In addition to the differences in partitioning, tree predictions are aggregated differently in `party` than in `randomForest`. In Chapter 2, we show that the aggregation scheme used by `party` can result in biased class probability estimates in classification problems. Using two real datasets, in which probability estimates are of interest, we show that combining the partitioning approach used in `party` with the aggregation technique implemented in `randomForest` yields results that outperform either technique individually.

1.2.2 Variable Importance

One popular feature of random forests is their ability to quantify the importance of explanatory variables in prediction. Roughly speaking, the importance of an explanatory variable X_j is estimated by considering the decrease in predictive accuracy when values of X_j are randomly permuted, thereby removing any association between X_j and Y. A more thorough description of this process is given in Chapter 3.

Measuring variable importance becomes increasingly complicated when data are missing. Hapfelmeier et al. (2012, 2014a,b) introduce an approach for estimating variable importance in such settings. This approach is intended to capture the value of the information provided by the data present, and variables with large amounts of missing data are often ranked as less important than they would be if complete data were available. While this is not unreasonable, there are many applications in which a practitioner might wish to estimate how important a predictor variable would be if complete data were available. In these situations, missing values are often imputed before variable importance is estimated via random forests. In Chapter 3, we consider six popular random forest imputation techniques and show that some of these are prone to underestimating, or perhaps more surprisingly, overestimating the importance of variables whose missing values have been imputed. Our study provides insight on which imputation approaches are most appropriate when the primary objective is to measure the importance of explanatory variables.

1.2.3 Robustness

Because random forest predictions are highly local, they are not as susceptible to the influence of outliers as other techniques such as linear regression. However, Roy and Larocque (2012) showed that improvements in performance can be achieved when robust measures are implemented in random forest regression. The estimated 0.5 quantile of the conditional

distribution for Y given $\boldsymbol{X}$ is a robust predictor (Meinshausen, 2006; Roy and Larocque, 2012). Li and Martin (2017) showed that modifying the training case weights described by Meinshausen potentially improves robustness. We further explore this idea, introducing a new robust approach for iteratively adjusting training case weights in Chapter 4 of this dissertation. Our approach is motivated by the robust approach for locally weighted polynomial regression introduced by Cleveland (1979). We use residual analysis to identify and down weight training cases with outlying response values, and show through simulations that this approach achieves strong performance on noisy, contaminated training data.

1.3 Role of Authors

Andrew Sage performed the investigations and was the primary author for all papers included in this dissertation. Ulrike Genschel and Dan Nettleton provided advice on the direction of the research and contributed to editing each of the manuscripts.

CHAPTER 2. TREE AGGREGATION FOR RANDOM FOREST CLASS PROBABILITY ESTIMATION

A paper submitted to *Statistical Analysis and Data Mining*

Andrew J. Sage, Ulrike Genschel, and Dan Nettleton

Abstract

In classification problems, random forests are often used to estimate the probability of a new case falling into each of C possible categories. These probabilities are routinely of interest, for example, in risk analysis. Different methods have been proposed for both growing random forests and aggregating predictions from individual trees, but comparative studies are limited. In this paper, we compare and contrast prominent random forest techniques, with particular emphasis on the aggregation of tree predictions. We consider two real datasets where class probability estimates are of interest, and demonstrate that combining the partitioning algorithm used in one approach with the aggregation technique used in another can result in performance that is superior to either approach individually.

2.1 Introduction

Random forest methodology is a well-known approach for classification and regression problems that is especially useful when there are a large number of predictor variables, or when many possible interactions exist. Introduced by Breiman (2001), random forests can be used to estimate the probability of an unobserved categorical response variable Y taking on category c for $c = 1, 2, \ldots, C$.

Two of the most popular approaches for growing random forests are implemented using the `randomForest` (Liaw and Wiener, 2002) and `party` (Hothorn et al., 2006a; Strobl et al., 2007, 2008) packages in `R` (R Core Team, 2016). The `randomForest` package grows individual trees using Breiman's Classification and Regression Tree (CART) algorithm (Breiman et al., 1984). The `cForest` function in the `party` package grows random forests using permutation tests to determine splits. Hothorn et al. (2006b) refer to trees grown in this manner as *conditional inference trees*.

Hothorn et al. (2006b) and Strobl et al. (2007) show that CART leads to biased variable selection, favoring continuous predictors or categorical predictors with many possible categories. Strobl et al. (2007) show that unbiased measures of variable importance can be obtained with `cForest`, when trees are grown from subsamples, rather than bootstrap samples, of the training data. Conditional inference trees have been employed extensively in the applied literature, especially when measuring variable importance (c.f. Scott et al. (2011); Arpaci et al. (2014)). While it stands to reason that `cForest` should benefit from its ability to utilize information contained in predictor variables that take on only a few values, its predictive performance relative to `randomForest`, under specific loss functions, has not been extensively studied.

The `randomForest` and `party` packages differ not only with respect to partitioning, but also in the way that information from individual trees is aggregated to obtain probability

estimates. A description of the `cForest` approach can be found in Hothorn et al. (2004) while Meinshausen (2006) discusses the algorithm used by `randomForest`. These papers focus primarily on regression and survival analysis. Less attention has been given in the literature to the effect of aggregation schemes on probability estimates in classification problems.

In this work, we examine probability estimates resulting from the aggregation schemes implemented by `randomForest` and `cForest` in classification problems. We show that the different aggregation approaches sometimes result in considerable discrepancies between the probability estimates, even if applied to the exact same forest. While the `cForest` partitioning approach is capable of superior performance in applications with a mix of categorical and numerical predictors, it is prone to overestimating the probability of the most likely response, while underestimating the probability of less likely outcomes. This bias results from a disproportionate influence of large, pure terminal nodes. Systematically underestimating the probability of unlikely events can result in serious consqeuences when the probability estimates are used, for example, to assess the risk of risk of defect or failure of large investments.

We further show that overestimation of the probability of the most likely outcome is most severe when tuning parameters are set to allow for deep trees with small terminal nodes. Requiring terminal nodes much larger than `cForest` defaults helps mitigate this concern, but does not eliminate it entirely. We demonstrate that combining the `cForest` partitioning algorithm with the aggregation approach used by `randomForest` can lead to performance superior to either technique individually. Finally, we show that optimal settings for tuning parameters pertaining to terminal nodesize depend heavily on the aggregation approach used and that the `randomForest` aggregation approach is less sensitive to the choice of tuning parameters than the `cForest` approach.

We structure the remainder of the manuscript as follows. In Section 2.2, we provide an overview of the partitioning algorithms employed by `randomForest` and by `cForest`. We

continue with a detailed discussion of the aggregation approaches utilized by each package, and conclude the section with a discussion of random forest tuning parameters. In Section 2.3, we conduct a simulation study and demonstrate that the `cForest` aggregation scheme can lead to biased probability estimates. In Section 2.4, we assess the performance of the `randomForest` and `cForest` partitioning and aggregation approaches on two real datasets, where class probabilities are of interest. We summarize our conclusions in Section 4.5.

2.2 Random Forest Fundamentals

Although we focus primarily on tree aggregation in this study, we begin with a brief overview of the partitioning algorithms under consideration to grow a random forest on a set of training data, consisting of n observations on p predictor variables.

2.2.1 Partitioning

The CART algorithm, implemented in `randomForest`, grows each individual tree using a newly generated bootstrap sample of the training set. Let $\mathbb{1}(\cdot)$ denote a generic indicator function that takes value one if the condition specified within parentheses is true and zero otherwise. A node is split, i.e. partitioned into two resulting subnodes, based on the values of a function of the form $\mathbb{1}(x \in \mathcal{S})$, where x is one of the predictor variables and $\mathcal{S}$ is a set of the form $(-\infty, \tau]$, $\tau \in \mathbb{R}$, if x is quantitative or a subset of categories if x is categorical. The variable x and the set $\mathcal{S}$ are selected so that the two resulting subnodes achieve maximal homogeneity with respect to the response variable. Although homogeneity of the response values in a node can be measured in multiple ways, the most common approach for a categorical response variable involves the use of the Gini index as a measure

of impurity. The Gini index for a node P is defined as

$$I_P = n_P \cdot \sum_{c=1}^{C} \pi_P(c)[1 - \pi_P(c)], \tag{2.1}$$

where n_P denotes the number of training cases in node P, and $\pi_P(c)$ represents the proportion of cases in P with response c, $c = 1, 2, \ldots, C$. If all training cases in node P have the same response, $I_P = 0$, while the value of I_P increases as the node becomes less pure. Let L and R denote the nodes resulting from a possible split. Then the decrease of impurity (increase in homogeneity) associated with the split can be measured by

$$\Delta I(P, L, R) = I_P - (I_L + I_R). \tag{2.2}$$

For each split in a tree, a new subset of $m < p$ predictor variables is randomly selected. Considering each variable in the randomly selected subset, all possible splits are evaluated, and the split that maximizes $\Delta I(P, L, R)$ is selected. Partitioning continues until all nodes are either pure, or smaller than a predetermined size, or until all training cases have identical values for the predictor variables randomly selected for the split.

When a set of predictor variables contains both continuous and categorical variables, the CART algorithm has been shown to favor splits on continuous predictors or categorical predictors with many possible categories. Because relatively many splits are possible for such variables, high values of $\Delta I(P, L, R)$ can often be achieved even for a variable not strongly associated with the response. As a consequence, CART-based partitioning sometimes fails to effectively utilize information provided by predictor variables with only a few possible splits. This issue is illustrated by Hothorn et al. (2006b) and Strobl et al. (2007).

Hothorn et al. (2006b) address the variable selection bias shown by CART using permutation tests, rather than Gini index, to determine the best split for each node. Specifically, within each node, upon randomly selecting a subset of predictor variables, permutation tests are used to first identify the predictor variable having the strongest association with the response variable and then to determine the best split on that variable. Because the choice of

which variable to split on is based on performing a single permutation test for each variable, regardless of the number of possible splits on that variable, this method does not suffer from the selection bias that affects CART-based partitioning. Strobl et al. (2007) show that while conditional inference trees reduce variable selection bias, compared to CART, this bias is not removed entirely unless trees are grown from subsamples, rather than bootstrap samples. This modification is implemented as the default setting for the `cForest` function.

Regardless of the method used to grow trees, once trees have been grown, predictions are made by moving a new case through each tree in accordance with its predictor variable values and the splitting rules determined from the training data. Once a case lands in a terminal node, response categories of training cases in that terminal node are used to estimate the probability of the new case taking on each possible response category. There are, however, multiple ways that this information is combined across trees to estimate each probability. These aggregation techniques are discussed in Subsection 2.2.2.

2.2.2 Tree Aggregation

In the following, we describe three different aggregation approaches used by the `randomForest` and `cForest` algorithms. We begin by introducing notation applicable to all three aggregation approaches.

Let $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$ denote a set of training data of size n, where $y_i \in \{1, 2, \ldots, C\}$ for $i = 1, \ldots, n$; that is y_i falls into one of C possible response categories. Given a new case $(\boldsymbol{x}^*, Y^*)$, where Y^* is unknown, we seek to estimate $\mathrm{P}(Y^* = c)$, the probability Y^* takes the value associated with response category c for $c \in \{1, 2, \ldots, C\}$. For

each aggregation scheme, an estimate of $P(Y^* = c)$ is given by

$$\widehat{\mathrm{P}}(Y^* = c|\boldsymbol{x}^*) = \frac{\sum_{t=1}^{T} w_t(\boldsymbol{x}^*) h_t(\boldsymbol{x}^*, c)}{\sum_{t=1}^{T} w_t(\boldsymbol{x}^*)}, \tag{2.3}$$

for some functions w_t and h_t, where w_t represents the weight given to tree t and h_t represents a prediction or probability estimate associated with tree t. Choices for w_t and h_t differ by aggregation scheme and will be discussed in more detain in sections 2.2.2.1–2.2.2.3. Expressions for w_t and h_t depend on additional quantities that are defined as follows.

Let $b_t(j)$ denote the number of times training case j occurs in the bootstrap sample or subsample used to grow tree t for $t = 1, 2, \ldots, T$. For each tree t and new case $\boldsymbol{x}^*$, let $N_t(\boldsymbol{x}^*)$ denote the collection of indices of training cases lying in the same terminal node as $\boldsymbol{x}^*$. That is,

$$N_t(\boldsymbol{x}^*) = \{j : (\boldsymbol{x}_j, y_j) \text{ is in the same terminal node as } \boldsymbol{x}^* \text{ for tree } t,\ j = 1, 2, \ldots n\}.$$

Let

$$a_t(\boldsymbol{x}^*, c) = \sum_{j \in N_t(\boldsymbol{x}^*)} b_t(\boldsymbol{x}_j) \mathbb{1}(y_j = c), \tag{2.4}$$

denote the number of category c responses in the terminal node containing $\boldsymbol{x}^*$ in tree t.

2.2.2.1 Equal Weighting of Tree Proportions (EW)

Since each tree produces an estimate of $P(Y^* = c|\boldsymbol{x}^*)$, which is given by the proportion of training cases in the same terminal node as $\boldsymbol{x}^*$ that have response category c, an overall random forest estimate can be found by averaging these proportions across trees. All trees

are given equal weight so $w_t(\boldsymbol{x}^*) = 1$, resulting in the estimate

$$\widehat{\mathrm{P}}_{EW}(Y^* = c|\boldsymbol{x}^*) = \frac{\sum_{t=1}^{T} h_t(\boldsymbol{x}^*, c)}{T}, \tag{2.5}$$

where $h_t(\boldsymbol{x}^*, c)$ represents the proportion of training cases in the same terminal node as $\boldsymbol{x}^*$ that take on response c, in tree t, i.e.,

$$h_t(\boldsymbol{x}^*, c) = \frac{a_t(\boldsymbol{x}^*, c)}{\sum_{j \in N_t(\boldsymbol{x}^*)} b_t(\boldsymbol{x}_j)}. \tag{2.6}$$

For a binary response variable, Y, this aggregation approach can be implemented using the `randomForest` package by treating Y as a numeric variable, taking on values 0 and 1. When the response variable is numeric, i.e., in regression, `randomForest` predicts the value of a response variable Y^*, by averaging the response values for all training cases in the same terminal node as $\boldsymbol{x}^*$ followed by averaging these predictions across all trees. Estimating the expected response, $\mathrm{E}\left(\mathbb{1}(Y^* = c|\boldsymbol{x}^*)\right)$, in this manner results in the estimate $\widehat{\mathrm{P}}_{EW}(Y^* = c|\boldsymbol{x}^*)$ given in (2.5).

We note that in regression problems, `randomForest` determines the best split using a mean square error loss function, instead of the Gini index, which is used for classification. However, mean square error and Gini index are equivalent for binary response variables. Therefore, this approach can be implemented directly, using random forest regression when $C = 2$. For $C > 2$, one can grow a random forest using Gini index, and then calculate probability estimates in accordance with (2.5) and (2.6).

2.2.2.2 Proportional Weighting of Tree Proportions (PW)

The `cForest` aggregation approach assigns weights to individual training cases according to the number of times a training case lands in the same terminal node as $\boldsymbol{x}^*$. This approach

is equivalent to averaging proportions given by individual trees, except the weight of tree t is proportional to the number of training cases in the same terminal node as $\boldsymbol{x}^*$ in tree t.

Let $h_t(\boldsymbol{x}^*, c)$ be defined as in (2.6). Then

$$w_t(\boldsymbol{x}^*) = \sum_{j \in N_t(\boldsymbol{x}^*)} b_t(\boldsymbol{x}_j). \tag{2.7}$$

An estimate of $\mathrm{P}(Y^* = c|\boldsymbol{x}^*)$ is given by

$$\widehat{\mathrm{P}}_{PW}(Y^* = c|\boldsymbol{x}^*) = \frac{\sum_{t=1}^{T} w_t(\boldsymbol{x}^*) h_t(\boldsymbol{x}^*, c)}{\sum_{t=1}^{T} w_t(\boldsymbol{x}^*)}. \tag{2.8}$$

The probability estimate in (2.8) can be equivalently obtained by pooling, across all trees in a forest, the contents of the terminal nodes containing $\boldsymbol{x}^*$ and then finding the proportion of all response values in the pool equal to c.

2.2.2.3 Tree Voting (TV)

For completeness, we include a description of a third possible aggregation approach that is implemented in the `randomForest` package when the response variable is categorical.

In this approach, each tree predicts the response category for a new case by taking the most frequently occurring response in the terminal node containing $\boldsymbol{x}^*$. That is, each tree casts a "vote" for the response category it determines to be most probable. The function $h_t(\boldsymbol{x}^*, c)$ in (2.3), can be written as

$$h_t(\boldsymbol{x}^*, c) = \mathbb{1}\left(a_t(\boldsymbol{x}^*, c) > a_t(\boldsymbol{x}^*, c') \text{ for all } c' \neq c\right). \tag{2.9}$$

In the case of a tie, i.e., if $\max\{(a_t(\boldsymbol{x}^*, c) : c = 1, \ldots, C\}$ is achieved for more than once choice of c, let $\tilde{c}$ represent the class that is randomly chosen from the maximizers and set $h_t(\boldsymbol{x}^*, \tilde{c}) = 1$, and $h_t(\boldsymbol{x}^*, \tilde{c}) = 0$ for all $c \neq \tilde{c}$.

The probability that a new case takes on response category c is estimated by the proportion of trees voting for category c. Predictions from each tree are given equal weight, so $w_t(\boldsymbol{x}^*) = 1$ for $t = 1 \dots T$ and for all $\boldsymbol{x}^*$. Therefore, equation (2.3) results in the probability estimate

$$\widehat{\mathrm{P}}_{TV}(Y^* = c|\boldsymbol{x}^*) = \frac{\sum_{t=1}^{T} h_t(\boldsymbol{x}^*, c)}{T}, \tag{2.10}$$

where $h_t(\boldsymbol{x}^*, c)$ is as defined in (2.9).

While this technique is popular when the task is simply classification (i.e. determining the most likely response category), it is ill-suited for class probability estimation. This method only considers the most likely response from each tree, without accounting for the proportion of cases taking on a particular response category. These proportions contain valuable information when class probability estimates are of interest. For the remainder of the paper, we therefore focus on the aggregation approaches described in sections 2.2.2.1 and 2.2.2.2.

2.2.2.4 Proposed Method

It is important to note that any aggregation approach can be used with any partitioning algorithm. While `cForest` implements proportional weighting of tree proportions, this choice is independent of the `cForest` partitioning algorithm. We propose combining the `cForest` partitioning approach, which implements conditional inference trees, with the equally weighted tree aggregation approach used by `randomForest` for regression. We show, in the ensuing sections, that this combined approach can be advantageous. Code for obtaining estimates in this manner is available on Github (Sage, 2017).

Before we study and compare performance of the partitioning and aggregation algorithms discussed in this section, we briefly explain the role of tuning parameters, necessary for the

implementation of `randomForest` and `cForest` in R. These tuning parameters must be set carefully in order to achieve each method's optimal performance.

2.2.3 Tuning Parameters

An important consideration in the use of random forest methodology is the choice of tuning parameters. In `randomForest`, there are two parameters that must be set with care. The first is the number of explanatory variables to consider for each split. The second is a terminal nodesize parameter, which is defined so that nodes smaller than this value are not split any further. These parameters are called *mtry* and *nodesize*, respectively. In `cForest`, there are three related tuning parameters. The *mtry* parameter is defined the same way in `cForest` as in `randomForest`, and the *minsplit* parameter in `cForest` is analagous to *nodesize*. In addition, `cForest` uses a parameter called *minbucket* which specifies the smallest nodesize allowed for a terminal node. For the remainder of the paper, we refer to the *nodesize*, *minsplit*, and *minbucket* collectively as *terminal nodesize parameters*.

Tuning parameters control the complexity of a random forest model. It is important to choose parameters that appropriately reflect the complexity of the datset being considered, thereby yielding optimal performance. Individual trees can be thought of as low bias, high variance predictors. This is especially true when terminal nodesizes are small. By averaging predictions of many different trees, random forests are intended to reduce variance while maintaining low bias. The *mtry* parameter is motivated by the need for each tree to be different in order to benefit from averaging across trees. If *mtry* is too large, trees will be similar to one another, and averaging estimates will do little to reduce variance. If the *mtry* value is small, then important variables might not be considered for splits sufficiently often, resulting in poor predictive performance. If terminal nodesizes are too large, then terminal nodes will contain many cases that are very different than $\boldsymbol{x}^*$, resulting in a biased

prediction. On the other hand, if terminal nodesizes are too small, the random forest is prone to overfitting.

The optimal choices for tuning parameters vary considerably between applications and tuning needs to be done for each dataset on which predictions are to be made. Because trees are grown on bootstrap samples or subsamples of the training data, not every case is used to grow each tree. The cases that do not occur in the sample used to grow a tree are referred to as out-of-bag (OOB) cases and are used to assess performance, in a manner similar to cross-validation. This procedure was originally suggested by Breiman (2003) and although subsequent literature (Bylander, 2002; Mitchell, 2011; Janitza, 2017) has shown that OOB error is often pessimistic, it is still a reasonable predictor of test error and can be used to guide the choice of tuning parameters. By default, `randomForest` uses values of $mtry = p/3$ and $nodesize = 5$ for regression problems, while `cForest` uses $mtry = 5$,$minsplit = 20$, and $minbucket = minsplit/3$.

As we assess the performance of different aggregation techniques, it is important to consider the impact of tuning parameters. In Sections 2.3 and 2.4, we show that the optimal settings for these parameters differ considerably depending on whether equal or proportional weighting is used, and that proportionally weighted tree aggregation is more sensitive to suboptimal nodesize settings than equal weighting. We also provide examples in which optimal performance is obtained using terminal nodesize settings much larger than their defaults.

2.3 Simulation Study

In this section, we illustrate the differences between equal and proportional weighting of trees when aggregating proportions. We use a very simple simulation scenario that helps to clarify issues with the `cforest` aggregation technique. Section 4 presents an evaluation

of methods in more complex, application-driven scenarios. Throughout this section, we generate data from the model

$$\begin{aligned} Y_i &\sim Ber(\pi_i), \\ logit(\pi_i) &= \beta_0 + X_{1i}, \\ X_{ji} &\overset{iid}{\sim} \mathcal{N}(0,1), \\ 1 \leq j \leq 4, 1 \leq i \leq n, & \end{aligned} \tag{2.11}$$

where $Ber(\pi_i)$ denotes a Bernoulli distribution with success probability π_i, and $\mathcal{N}(0,1)$ denotes a standard normal distribution. Although we include four predictors, $X_j, 1 \leq j \leq 4$, which are independent and identically $\mathcal{N}(0,1)$ distributed, only X_1 is associated with the response. Throughout this section, we will call the event $Y = 1$ a success and the event $Y = 0$ a failure.

We begin by illustrating a situation in which the two aggregation schemes result in considerably different probability estimates. Since our intent for this section is to study differences in aggregation approaches, we only consider trees grown by the `cForest` partitioning technique. We simulated a training set of size $n = 100$ from model (2.11) with $\beta_0 = -2$, and grew a small forest of 20 trees, using tuning parameters *mtry*=2, *minsplit*=5, and *minbucket*=1. We then made a prediction for a single new case with $x_1 = 0.6207, x_2 = 1.8119, x_3 = 1.9120, x_4 = -1.3638$. Table 2.1 shows a summary of the terminal nodes containing the new case for each of the 20 trees.

Table 2.1: A summary of the terminal nodes containing a new case in a forest of 20 trees. The largest terminal nodes are the ones containing large numbers of training cases in the majority class. Table entries are ordered by terminal nodesize from largest to smallest.

Tree	Terminal Nodesize	No. Successes	Prop. Successes	Tree	Terminal Nodesize	No. Successes	Prop. Successes
8	24	0	0.000	17	14	0	0.000
14	23	0	0.000	4	12	0	0.000
1	22	0	0.000	12	11	0	0.000
10	22	0	0.000	13	4	2	0.500
18	22	0	0.000	11	3	2	0.667
15	20	0	0.000	2	2	1	0.500
16	20	0	0.000	3	2	1	0.500
5	16	0	0.000	7	2	1	0.500
6	16	0	0.000	9	2	2	1.000
19	16	0	0.000	20	2	1	0.500
				Sum	255	10	

The largest nodes containing this test case are all pure nodes and consist of cases from the majority class. The estimated success probabilities for the new case are $\widehat{\mathrm{P}}_{EW}(y^* = 1) \approx 0.2083$, and $\widehat{\mathrm{P}}_{PW}(y^* = 1) = \frac{10}{255} \approx 0.0392$. Note that $\widehat{\mathrm{P}}_{PW}(y^* = 1)$ is considerably lower as a consequence of the trees with large, pure terminal nodes receiving more weight than those with small terminal nodes in the calculation of this estimate. Based on model (2.11), the true success probability is $\frac{\exp(-2+0.6207)}{1+\exp(-2+0.6207)} \approx 0.2011$.

The preceding example is intended merely for illustrative purposes. In practice, forests much larger than 20 trees should be grown, and their performance needs to be evaluated on

more than one test case. Still, the example shows that large, pure terminal nodes affect proportional tree aggregation more than they affect aggregation by equal weighting, potentially resulting in very different probability estimates in the classification setting. This is typically not a concern in regression, as the response variable takes on many different values, allowing splitting to continue into approximately equally sized terminal nodes.

To further investigate the calibration of estimates produced by each aggregation approach, we simulated a training set of size 10,000 from model (2.11), using a value of $\beta_0 = -2.564$. This value was selected to create a dataset with an average true success probability of approximately 0.1. A random forest, consisting of 500 trees, was grown using `cForest` with default paramter settings of *minsplit* = 20, and *minbucket* = 7. The *mtry* parameter was set to 2. Figure 2.1 displays the probability estimates, plotted against the true probabilities, determined from model (2.11).

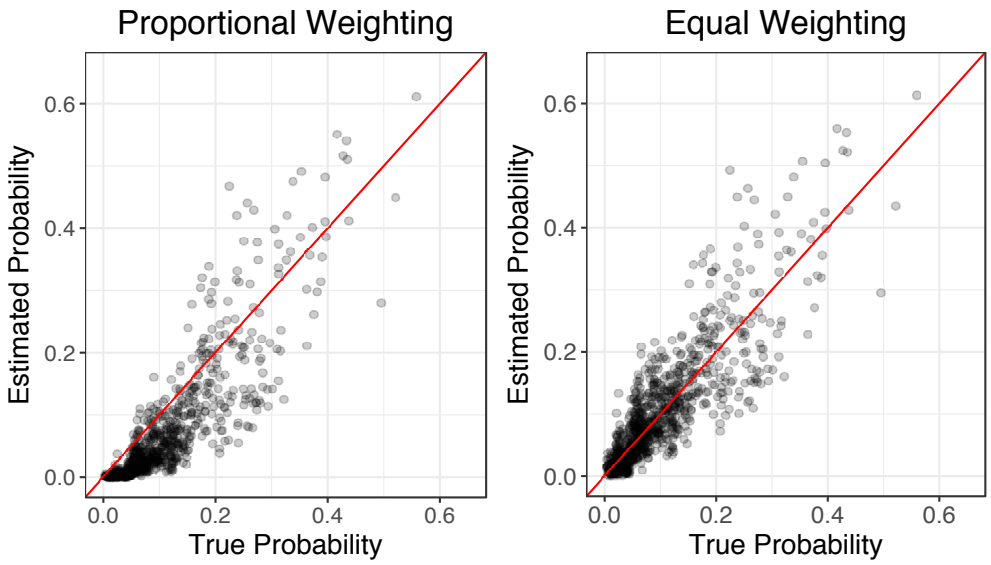

Figure 2.1: Probability estimates using equal and proportional weighting for the aggregation of tree predictions and default `cForest` parameter settings, *minsplit* = 20 and *minbucket* = 7.

We see in Figure 2.1 that proportional aggregation typically underestimates the success probabilities. This is especially true for cases whose true success probabilities are less than 0.1, which constitute the majority of the test cases. The smaller than expected probabilities are the result of large pure terminal nodes in trees yielding a success probability estimate of 0, heavily influencing the proportionally weighted probability estimates. On the other hand, the equally weighted approach results in estimates that do not appear to be systematically too high or too low. In fact, the mean success probability for the 1,000 test cases was 0.0984, from model (2.11). The mean estimated success probability from proportional weighting is a substantially lower 0.0619, while the mean estimate, resulting from equal weighting, was 0.0994, which is right on target. Depending on the application at hand, underestimating the risk of a rare event by almost 4 percentage points can result in significant losses.

The values of β_0 used in the preceding examples were selected in order to produce a dataset with far more failures than successes. In the literature, problems where one outcome is much more likely than another are referred to as as unbalanced classification problems. As is seen in Table 2.1, the pure terminal nodes that predict a failure are much larger than the pure terminal nodes that predict a success. A natural question is how the degree of imbalance affects probability estimates. To examine this question, values of β_0, given in Table 2.2, were chosen to create datasets based on model (2.11) with mean success probabilities set to $p = 0.05, p = 0.10, p = 0.20, p = 0.30, p = 0.40$, and $p = 0.50$. To get a clear sense of the distribution of conditional probability estimates, we simulated 10,000 cases for both the training and test data. The same values of X_1, X_2, X_3, and X_4 were used for each value of β_0. Due to randomness, the true proportions of successes in the training and test sets vary slightly from the selected values of p (see Table 2.2). Random forests, consisting of 500 trees, were grown using default `cForest` parameter settings for *minsplit* and *minbucket*, and also using *minbucket* = *minsplit* = 1 to grow trees of maximal depth. We continue to

use $mtry = 2$. Based on the simulated data sets and random forests grown, we obtained probability estimates using each aggregation scheme.

Table 2.2: In the Bernoulli response example the proportion of successes in each training and test set for different imbalance ratios. Due to random variability, these differ slightly from the expected success probability, given by p.

β_0	p	Train Prop.	Test Prop.
-3.371	0.05	0.0541	0.0496
-2.564	0.10	0.1066	0.1026
-1.650	0.20	0.2002	0.1978
-1.018	0.30	0.3030	0.2983
-0.490	0.40	0.3994	0.3953
0	0.50	0.5049	0.5021

Figure 2.2 displays boxplots of the distributions of the true success probabilities, from model (2.11), and estimated probabilities for each imbalance ratio, using default `cForest` settings, for both equal (EW) and proportional (PW) aggregation. It is immediately apparent that the median PW probability estimate is consistently less than the true median probability, while the distribution of EW estimates closely resembles the distribution of true probabilities. For PW, the extent of the difference appears to be related to the size of the imbalance between probability of success and failure.

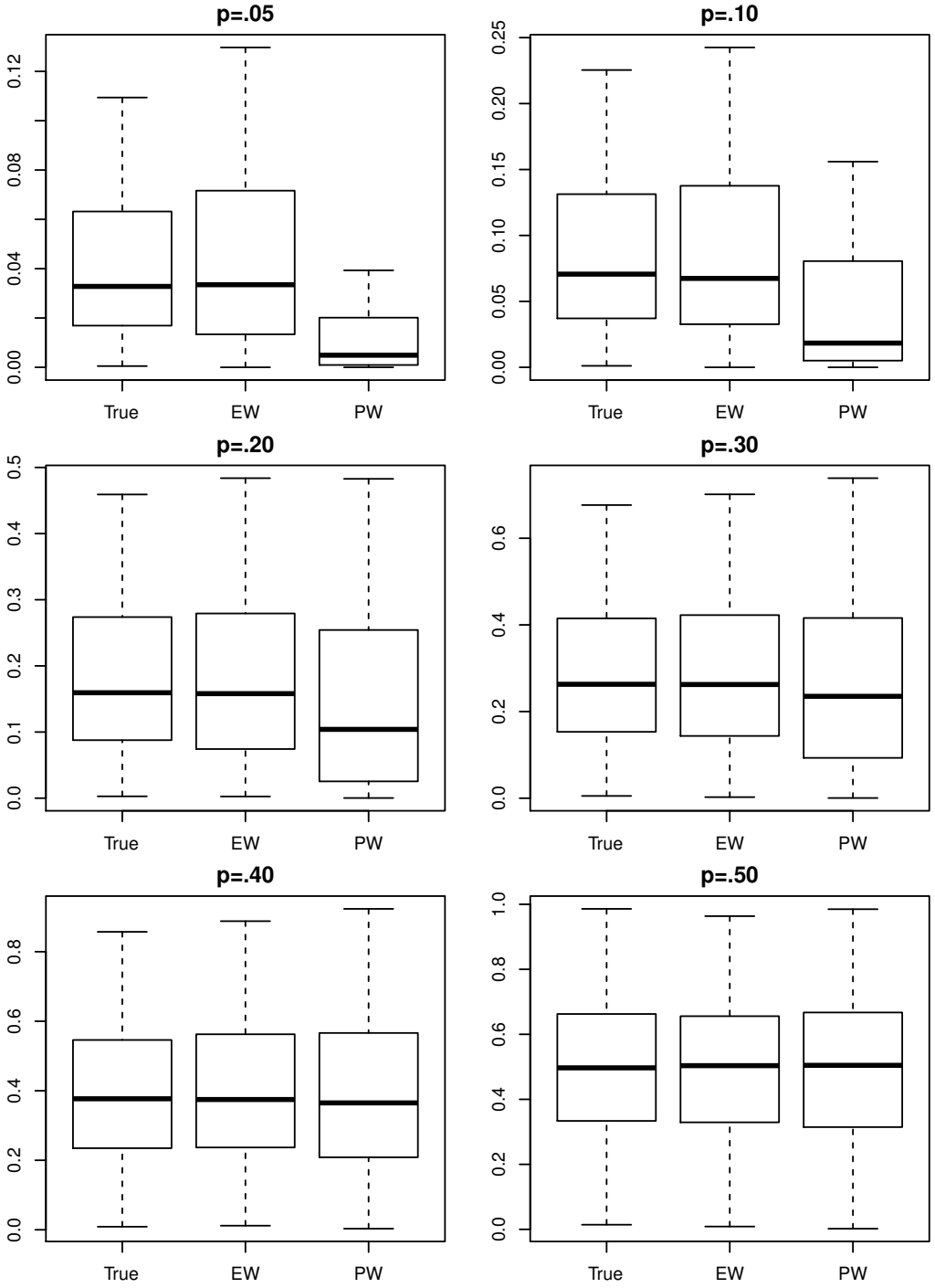

Figure 2.2: Distribution of true and estimated probabilities using various degrees of imbalance and the default `cForest` terminal nodesize parameter settings of $minsplit = 20$ and $minbucket = 7$.

While concerns stemming from proportional weighted averaging are most obvious in unbalanced situations, it is inappropriate to dismiss this concern in more balanced settings. We see in Figure 2.2 that as the data become more balanced, and the distribution of PW estimates becomes more appropriately centered, it still exhibits more variability than the distribution of true probabilities. The increased spread is a result of the PW estimates being pulled toward the extremes by large pure nodes. This issue becomes more apparent when trees of maximal depth are grown, as is seen in Figure 2.3. Even for $p = 0.5$, large pure terminal nodes influence PW predictions heavily, pulling the probability estimates for cases more likely to result in successes toward 1 and for cases more likely to result in failure toward 0. In either case PW estimates lead to underestimation of the uncertaintly associated with a prediction.

Figures 2.2 and 2.3 suggest that problems arising from the PW aggregation scheme are more severe when trees of maximal depth are grown. Intuitively speaking, the sizes of large pure terminal nodes are unaffected by parameters restricting terminal nodesize, while nodes that are allowed to be small are given less weight, allowing the large pure terminal nodes to dominate PW probability estimate. While this suggests that we might be able to improve PW predictions by specifying large values for *minsplit* and *minbucket*, a downside is that an increase in these values might increase bias in tree predictions, which would be based on a large number of training cases, rather than just those most similar to the case being predicted. We address this topic in more detail in Section 2.4.

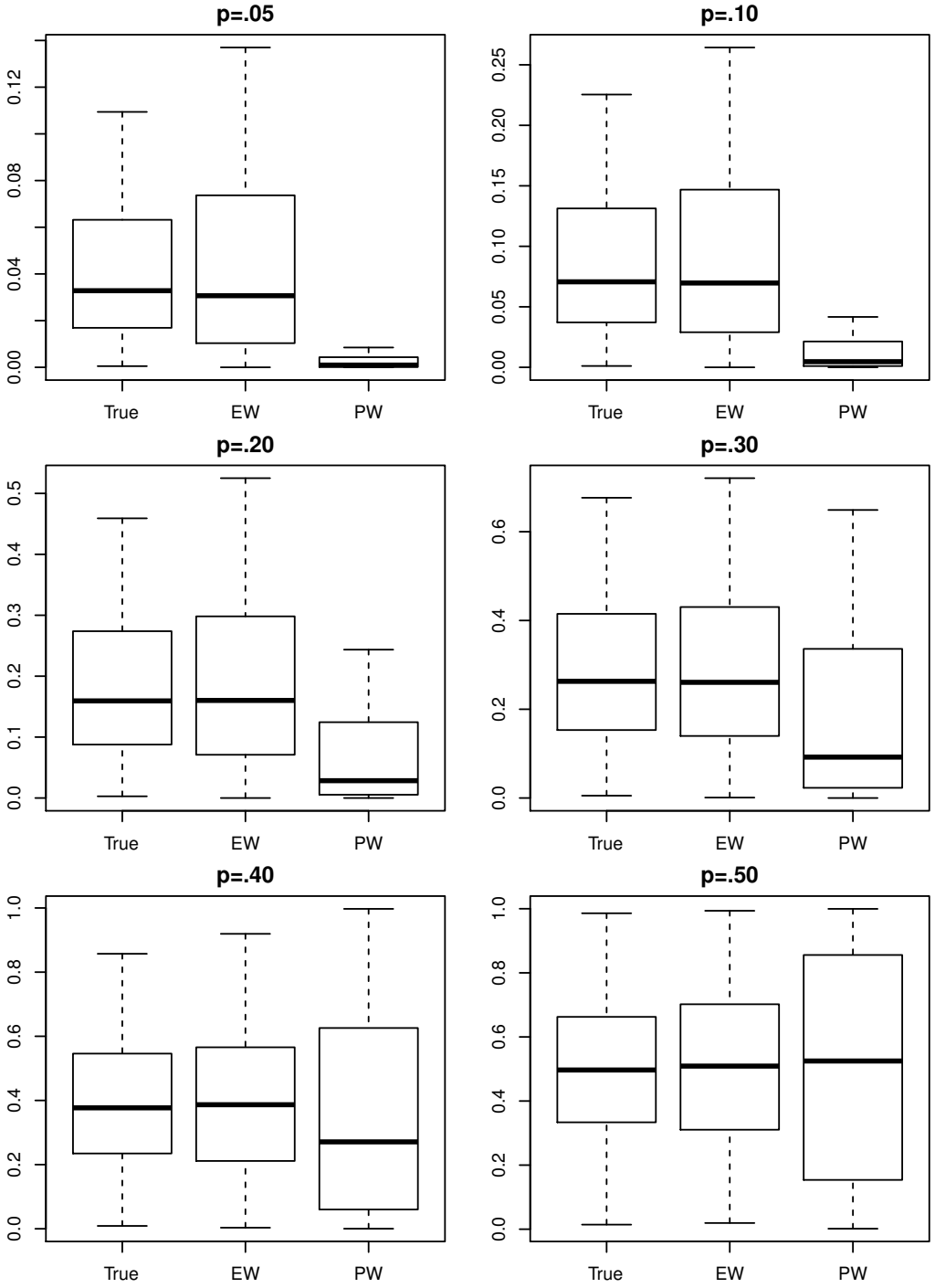

Figure 2.3: Distribution of true and estimated probabilities using various degrees of imbalance and terminal nodesize parameter settings of $minsplit = minbucket = 1$.

2.4 Data Applications

We have seen that aggregating tree predictions using a weighted average of tree proportions, as is done in `cForest`, potentially results in overestimation of the probability of the most likely response. This is especially a concern when working with unbalanced data. Possible remedies include requiring large terminal nodesizes, or weighting each tree equally when averaging. In this section, we explore the performance of the EW and PW aggregation approaches as well as the impact of increasing terminal nodesize using two real datasets in which class probability estimation is of interest.

The first dataset we consider is the credit card default (CCD) (Yeh and Lien, 2009) dataset available in the UCI machine learning repository (Lichman, 2013). The dataset consists of observations on 30,000 credit card holders in Taiwan. The response is binary, indicating whether the holder defaulted on a payment. Among the 23 predictor variables are numeric and categorical ones. The number of defaulting credit card holders is 6,636, accounting for 22.12% of observations.

The second dataset contains information on 8,748 students who declared a major in science, technology, engineering, or mathematics (STEM) at the beginning of their first year at a large public university between Fall 2011 and Fall 2014. The response variable is again binary, indicating whether the student left a STEM major by the start of the second year of enrollment. Note that students who left the institution before the start of the second year are not included in the dataset. The 30 predictor variables include numerical variables, such as standardized test scores, and categorical variables, such as gender and type of STEM major (e.g. biological sciences, engineering, etc.). While the original dataset is larger, we only consider a subset of the data containing the 8,748 students with complete data. A total of 873 students (9.98%) left STEM majors during their first year.

We used cross-validation to assess the performance of each partitioning and aggregation technique. We randomly divided the CCD dataset into 10 folds, each of size 3,000. Each fold was withheld once, and random forests, consisting of 500 trees, were trained on the remaining 27,000 observations and used to predict the 3,000 withheld cases. An analagous procedure was applied to the STEM dataset, using 9 folds of equal size. We examine performance using the partitioning and aggregation approaches employed by `cForest` and `randomForest` (for regression), along with our proposed method of combining the `cForest` partitioning algorithm with the `randomForest` regression aggregation approach. Table 2.3 summarizes the partitioning and aggregation approaches we considered.

Table 2.3: A summary of the partitioning and aggregation approaches we considered.

Method	Partitioning Algorithm	Aggregation Algorithm
`cForest`	Permutation Tests	Proportional Weighting
`randomForest`	CART (Gini Index)	Equal Weighting
Combined	Permutation Tests	Equal Weighting

We evaluate the performance of each partitioning and aggregation algorithm based on various terminal nodesize settings (*minsplit* in `cForest` and *nodesize* in `randomForest`), which are given in Table 2.4. Because an initial exploration showed that the default 3:1 ratio of *minsplit* to *minbucket* is often optimal and results are largely insenstive to changes in *minbucket*, we kept the default ratio.

Predictions are evaluated using a log loss function, which for vectors of estimates $\widehat{\mathbf{p}}$ and true responses $\mathbf{Y}$, is defined as

$$L(\widehat{\mathbf{p}}, \mathbf{y}) = -\sum_{i=1}^{n} [\mathbb{I}(Y_i = 1)\log(\widehat{p}_i) + \mathbb{I}(Y_i = 0)\log(1 - \widehat{p}_i)]$$

where $\widehat{p}_i = \widehat{\mathrm{P}}(Y_i = 1|\boldsymbol{x}_i)$. This loss function is a popular choice for evaluating class probability estimators. It is a proper scoring rule in the sense that its expectation is minimized by letting $\widehat{p}_i = \mathrm{P}(Y_i = 1|\boldsymbol{x}_i)$ for each i. Other proper scoring rules for class probability estimation include Brier score, which is equivalent to sum of squared errors when the response variable is binary, and boosting loss (Buja et al., 2005).

Table 2.4: Values of tuning parameters considered using cross-validation.

Parameter	Values Considered for CCD	Values Considered for STEM
mtry	4, 8, 16	3, 5, 10, 20
minsplit (or *nodesize*)	1, 10, 25, 50, 100, 200, 300, 500	1, 5, 10, 25, 50, 75, 100, 150, 200, 250, 300, 400, 500

For each of the three prediction methods, and each terminal nodesize, we determined the optimal *mtry* value by minimizing $L(\widehat{\mathbf{p}}, \mathbf{y})$ on OOB cases. Since each test case is predicted exactly once, we expect the mean estimated credit card default probability and the mean estimated probability of leaving STEM to be close to 0.2212, and 0.0998, respectively, when averaging across all folds. Average probability estimates that differ from these suggest a calibration problem.

Figure 2.4 shows, as a function of terminal nodesize, the average estimated probability of a customer defaulting on credit card loans, or a student leaving STEM for each technique. We see that `cForest` severly underestimates these probabilities when the terminal nodesize parameter is small. The disparity decreases as the terminal nodesize grows, but does not completely vanish. This behavior is especially pronounced in the STEM application, where even a terminal nodesize setting of 500 results in an average probability of leaving STEM that is about one-half of one percentage point lower than expected. Conversely the `randomForest` and combined approaches each appear to slightly overestimate the average probability of

defaulting on a loan, or leaving STEM when the terminal nodesize is small, but quickly converge to the proportions seen in the training data as nodesize grows.

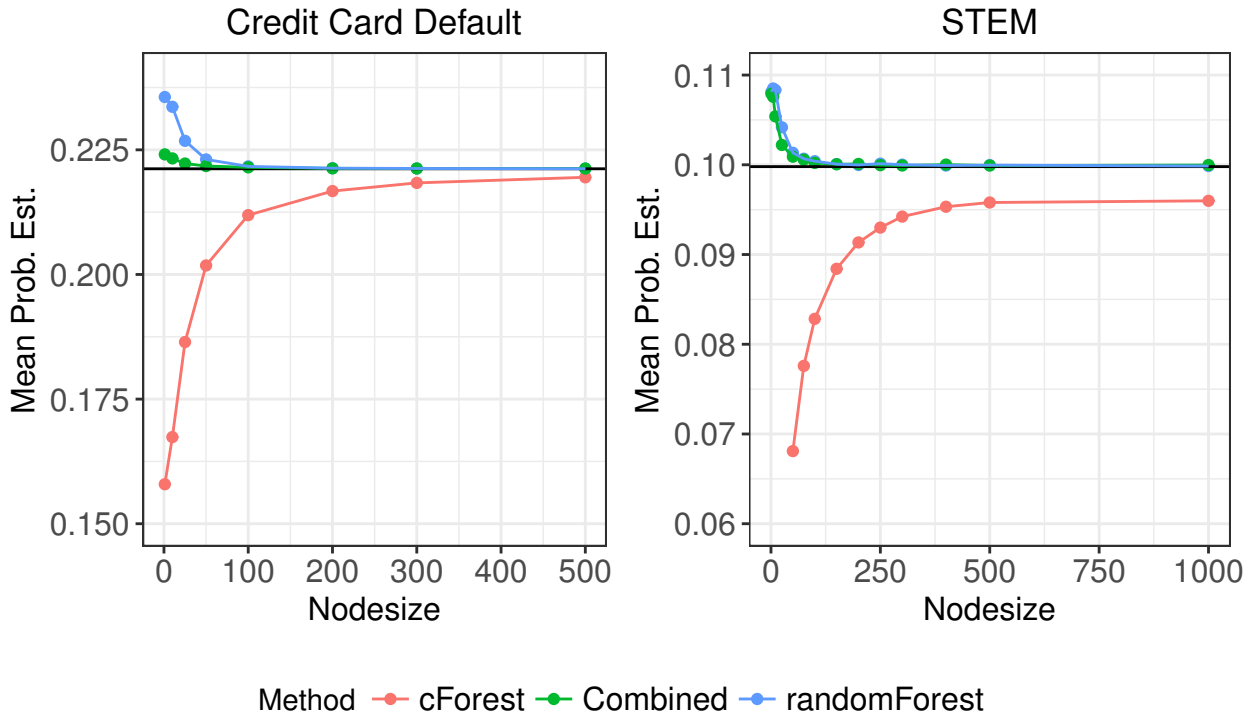

Figure 2.4: The average estimated probability of default or leaving STEM is plotted against the value of the terminal nodesize parameter. The average `cForest` estimates are consistently low, especially for small terminal nodesizes.

Figure 2.5 shows the values of the log loss function on the holdout sets, averaging across folds. Plots (a) and (b) include a large enough range on the vertical axis to display all values, while plots (c) and (d) focus on regions of interest. The convex shape is consistent with the discussion of tuning parameters and model complexity in Section 2.2.3. We see that `cForest` performs poorly when small terminal nodesize parameters are used, while the combined approach is less sensitive to small nodesizes. In the CCD dataset, the proposed approach achieves the best performance for each terminal nodesize, with a clear advantages

for small terminal nodesizes. In the STEM dataset, the `cForest` and combined approaches outperform `randomForest`. The combined approach performs best for terminal nodesizes less than 200, and is approximately equivalent to `cForest` for terminal nodesize settings of 200 or greater. Optimal performance is obtained using nodesizes considerably larger than than `cForest` and `randomForest` default values of 20 and 5, respectively. As nodesizes grow very large, performance deteriorates for all three methods.

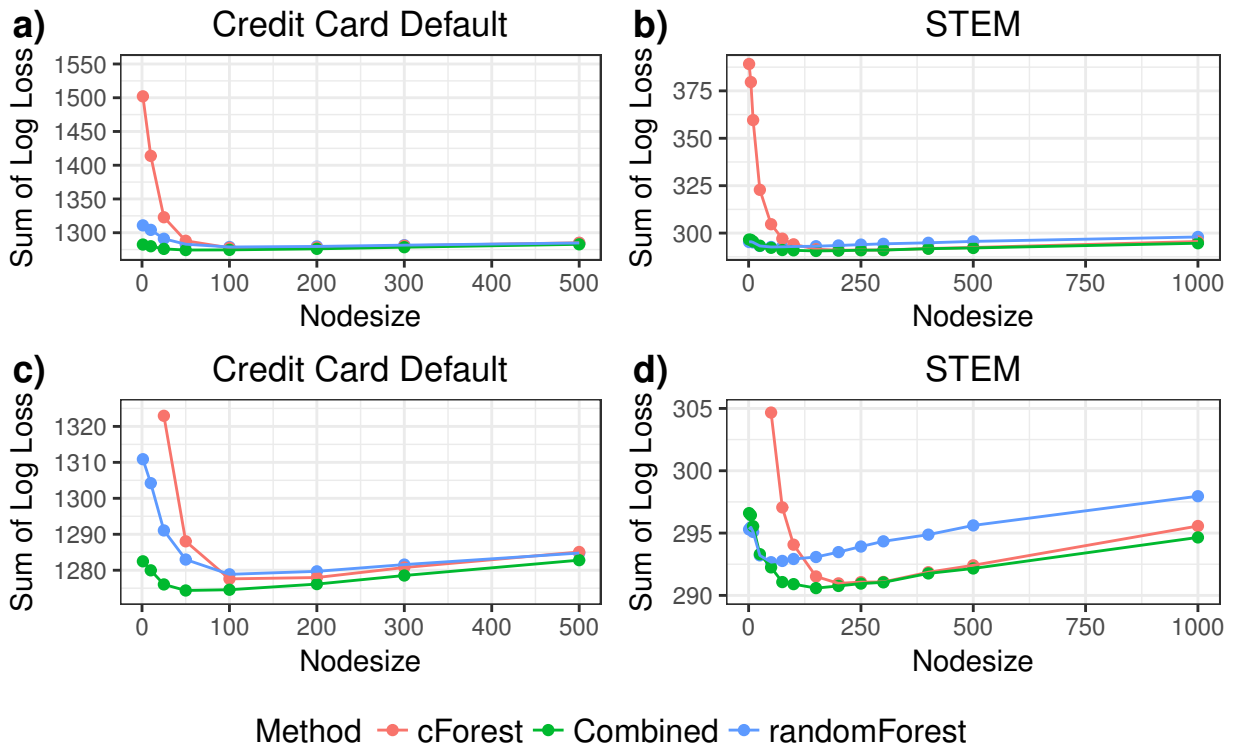

Figure 2.5: The total log loss is plotted against the value of the terminal nodesize parameter. Figures (c) and (d) provide different versions of Figures (a) and (b), focusing the vertical axis on a narrower range.

Through Figures 2.4 and 2.5 we sought to study the performance of each algorithm as a function of nodesize. In practice, however, we are required to determine the appropriate nodesize parameter setting in addition to the *mtry* setting based on the OOB error in the

training data. Accordingly, we determined the best *nodesize*, *mtry* combination for each technique, within each fold, and used those values to obtain predictions for the holdout set. Table 2.5 gives the resulting log loss values on the holdout set for each partition, and each application. Values in boldface indicate the lowest log loss value among the three methods under consideration.

Table 2.5: Total log loss on each partition using cross validation for the CCD and STEM datasets.

	Log Loss-CCD			Log Loss-STEM		
Partition	`cF`	Comb.	`rF`	`cF`	Comb.	`rF`
1	1216.9	**1214.0**	1219.0	**273.7**	276.0	277.3
2	1323.1	**1314.4**	1327.7	296.3	**295.4**	297.1
3	**1257.1**	1259.7	1262.7	306.6	**304.5**	307.8
4	1279.1	**1277.4**	1283.4	299.0	299.9	**298.9**
5	1286.1	1281.8	**1279.8**	322.9	**320.0**	320.5
6	1303.9	**1302.5**	1308.1	**278.2**	279.1	280.0
7	**1302.7**	1302.8	1307.4	**288.5**	289.1	291.7
8	**1306.6**	1307.2	1310.8	**277.2**	279.3	278.9
9	**1240.3**	1246.3	1249.1	**276.0**	276.7	280.5
10	1243.0	**1242.8**	1246.3			
Average	1275.9	1274.9	1279.4	290.9	291.1	292.5

We see that the the permutation test based partitioning methods appear to achieve superior performance to CART, which is consistent with the claim that these techniques better utilize information provided by predictor variables that take on only a few values. Our combined method appears to achieve slightly more favorable performance on the CCD

dataset, while `cForest` performs slightly better in the STEM application. The differences between `cForest` and the combined approach are small.

The similar performance of `cForest` and our combined method is a result of the large terminal nodesize settings that are optimal in this problem. For the CCD dataset, the average terminal nodesizes, as determined using OOB log loss, were 140 for `cForest`, 150 for `randomForest`, and 120 for the combined method. For the STEM dataset, these values were 211.1, 83.3, and 130.6, respectively. However, applications can occur in which requiring large terminal nodes is undesirable.

Requiring large terminal nodes often prevents the separation of cases most likely to result in a credit card holder defaulting on a loan, or a student leaving STEM, from cases in which these outcomes are moderately likely. Suppose that the cost of failing to predict a credit card customer defaulting on a payment, or a student leaving STEM is higher than the cost of wrongly predicting such an occurrence. In these situations, a loss function that heavily penalizes failing to detect defaulting on a payment, or leaving STEM is appropriate. Let

$$L_\alpha(\widehat{\mathbf{p}}, \mathbf{y}) = -\sum_{i=1}^{n} [\alpha \mathbb{I}(Y_i = 1) \log(\widehat{p}_i) + \mathbb{I}(Y_i = 0) \log(1 - \widehat{p}_i)]$$

for $\alpha \geq 1$. When it is especially important to accurately estimate probabilities for the cases carrying the highest risk (high $\mathrm{P}(Y_i = 1)$ in our examples), this can be achieved by setting $\alpha > 1$.

Table 2.6 shows how the average optimal terminal nodesize, determined using OOB error, changes as α increases, for each application. We see that the optimal terminal nodesize decreases substantially for both `randomForest` and the combined approach. Smaller terminal nodesizes allow for splits that separate cases with very high risk, from those with moderately high risk, decreasing bias in probability estimates for the cases of greatest interest. However, optimal nodesizes increase for `cForest`. This results from the `cForest` aggrega-

tion scheme underestimating the probability of customer default or a student leaving STEM when nodesizes are allowed to be small. The cost of this underestimation outweighs the benefit of growing deeper trees, preventing `cForest` from taking advantage of finely partitioned training data.

Table 2.6: Average terminal nodesize for each α, for the CCD and STEM datasets.

		α								
Dataset	Technique	1	1.5	2	3	4	5	10	100	1000
CCD	`cForest`	150.0	220	290	300.0	380.0	400.0	440.0	460.0	460.0
CCD	Combined	120.0	105	105	77.5	60.0	51.0	25.7	10.9	7.9
CCD	`randomForest`	150.0	140	60	14.5	11.5	11.5	8.2	5.5	4.6
STEM	`cForest`	211.1	300.0	377.8	411.1	411.1	411.1	411.1	422.2	422.2
STEM	Combined	130.6	136.1	130.6	130.6	85.1	53.4	4.2	4.2	4.2
STEM	`randomForest`	83.3	38.3	14.4	9.0	9.0	7.3	7.3	6.3	6.3

Figure 2.6 shows the percent increase in the L_α value incurred for each technique compared to that of the optimal technique, for each value of α. As we observed previously, there is little difference between `cForest` and the combined approach when $\alpha = 1$. However, the relative performance of `cForest` begins to deteriorate quickly as α increases, due to its inability to finely partition training data without underestimating the probability of default or leaving STEM. For each dataset, the combined approach achieves the best perfomance for $\alpha = 1.5$, and $\alpha = 2$, and continues to outperform `cForest` for larger α. In these applications, the `randomForest` algorithm performs best for large α. This is due to the fact that `randomForest` tends to overestimate the probability of default or leaving STEM when terminal nodesizes are small, as seen in Figure 2.4. There is no reason to believe that `randomForest` is systematically likely to overestimate or underestimate the probability of an unlikely event in general, and its superior performance for large α is likely just an artifact

of these data. However, it is clear that the combined method is preferable to `cForest` for large α due to its ability to more accurately estimate probabilities for high-risk cases.

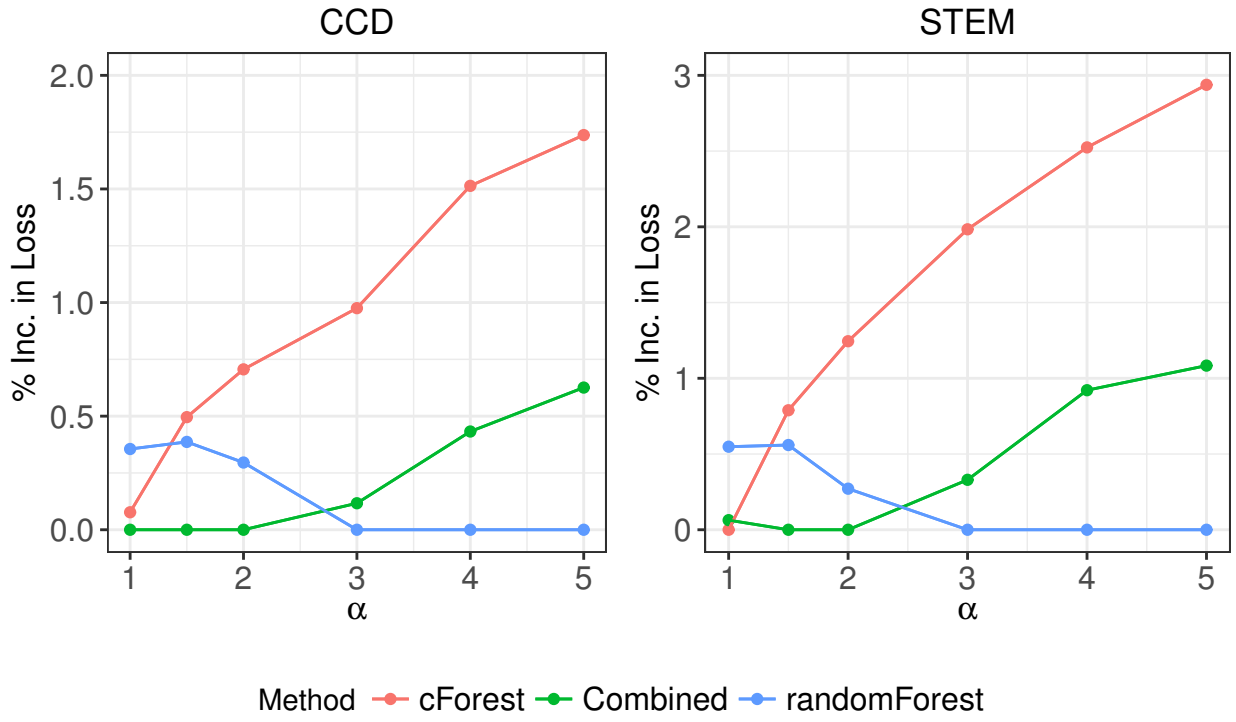

Figure 2.6: The percent of increase in loss for each technique relative to the best technique for a given penalty parameter α is plotted against α.

2.5 Conclusions

We have seen that random forests grown using conditional inference trees, implemented in the `party` package in `R`, are capable of achieving performance superior to those grown using the CART algorithm in `randomForest`, especially in situations involving a mix of continuous and categorical explanatory variables. We have further shown that the aggregation scheme employed by `party` tends to overestimate the probability of the most likely outcome,

a behavior that is especially prominent in unbalanced classification problems, though not restricted to this scenario.

When the depth of the trees is limited by use of a large *minsplit* value, the difference in performance between equally weighted tree aggregation and proportional weighting of trees is mostly negligible. When deep trees with small terminal nodes are desirable, equal weighting achieves superior performance to proportional weighting. The equal weighting aggregation scheme is less sensitive to changes in terminal nodesize settings than proportional weighting. This is a result of large, pure terminal nodes receiving disproportionate weight in proportional tree aggregation when when other nodes are allowed to be small. The appropriate teminal nodesize setting is context dependent, and careful use of cross-validation and OOB error when setting tuning paramters is essential.

In both data applications, trees grown in `cForest` outperform those grown in `randomForest`. While `cForest` employs proportional weighting, it is possible to use equally weighted tree aggregation on random forests grown using `cForest`. In our applications, the performance of this combined approach is at worst on par with `cForest` and is better in some situations. We recommend using equally weighted tree aggregation in situations where deep trees with small terminal nodes are desirable, or when accurately estimating the mean success probability for a large number of cases is important. If a dataset contains a mix of categorical and numerical predictor variables, then combining this aggregation scheme with the `cForest` partitioning approach has the potential to produce better probability estimates than `cForest` or `randomForest` individually. Combining the `cForest` partitioning approach with the regression `randomForest` aggregation technique allows users to take advantage of partitioning based on permutation tests and aggregation that is not disproportionately influenced by large pure terminal nodes, in order to more accurately estimate class probabilities.

Bibliography

Arpaci, A., Malowerschnig, B., Sass, O., and Vacik, H. (2014). Using multi variate data mining techniques for estimating fire susceptibility of tyrolean forests. *Applied Geography*, 53:258–270.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Breiman, L. (2003). *Manual—setting up, and understanding random forests V4.0*.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Buja, A., Stuetzle, W., and Shen, Y. (2005). Loss functions for binary class probability estimation and classification: Structure and applications. *Working draft, November*.

Bylander, T. (2002). Estimating generalization error on two-class datasets using out-of-bag estimates. *Machine Learning*, 48(1-3):287–297.

Hothorn, T., Bühlmann, P., Dudoit, S., Molinaro, A., and Van Der Laan, M. J. (2006a). Survival ensembles. *Biostatistics*, 7(3):355–373.

Hothorn, T., Hornik, K., and Zeileis, A. (2006b). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674.

Hothorn, T., Lausen, B., Benner, A., and Radespiel-Tröger, M. (2004). Bagging survival trees. *Statistics in Medicine*, 23(1):77–91.

Janitza, S. (2017). On the overestimation of random forests out-of-bag error. *Technical Report*.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

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

Meinshausen, N. (2006). Quantile regression forests. *The Journal of Machine Learning Research*, 7:983–999.

Mitchell, M. W. (2011). Bias of the random forest out-of-bag (oob) error for certain input parameters. *Open Journal of Statistics*, 2011.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Sage, A. J. (2017). *Github Repository*.

Scott, S. B., Jackson, B. R., and Bergeman, C. (2011). What contributes to perceived stress in later life? a recursive partitioning approach. *Psychology and Aging*, 26(4):830.

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., and Zeileis, A. (2008). Conditional variable importance for random forests. *BMC Bioinformatics*, 9(1):1.

Strobl, C., Boulesteix, A.-L., Zeileis, A., and Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(1):1.

Yeh, I.-C. and Lien, C.-h. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2):2473–2480.

CHAPTER 3. RANDOM FOREST VARIABLE IMPORTANCE IN THE PRESENCE OF MISSING DATA

A paper in preparation

Andrew J. Sage, Ulrike Genschel, and Dan Nettleton

Abstract

The ability to assess variable importance is a popular feature of random forest methodology. Missing values in predictor variables are often imputed prior to making predictions and measuring variable importance. Numerous random-forest-based imputation techniques have been proposed. We assess the impact of these techniques on a random-forest-based measure of variable importance. After some imputation techniques are applied, the variable importance measure produces inflated estimates of importance for variables with many missing observations. Other imputation techniques lead to deflated measures of importance for such variables. We compare the impact of six random-forest-based imputation techniques on measurements of variable importance, considering various possibilities for the number of missing values, and the correlation between predictor variables. Our work provides guidance on the choice of imputation technique for researchers who are interested in assessing variable importance when missing values are a concern.

3.1 Introduction

Random forest methodology is a nonparametric machine learning approach that uses recursive partitioning to make predictions in classification and regression problems. Random forests routinely handle situations where it is difficult to specify a parametric model because of many predictor variables, unknown nonlinear relationships between the predictors and the response, unknown high-order interactions, or a variety of other model complexities.

An important feature of random forest methodology is its ability to measure the importance of predictor variables, which is often of interest in model selection or interpretation. Measuring variable importance can be challenging when a dataset contains missing values, especially if certain predictor variables are more prone to missingness than others. Complete case analysis, in which cases with missing values are ignored altogether, is widely considered a poor choice as it discards potentially useful information. When data are missing, researchers are likely to be interested in knowing which predictor variables would be most important in predicting a response if the data could have been fully observed. This information might, for example, be used to justify an increased effort to collect additional data on the most important predictors.

Missing values are often imputed prior to making predictions. Multiple imputation, a process by which missing values are imputed repeatedly in order to obtain several imputed datasets, is a popular approach that allows for an assessment of the variability associated with imputation. Numerous imputation approaches that make use of random forests are available (e.g. Breiman (2001); Ishwaran et al. (2008); Stekhoven and Bühlmann (2012); Doove et al. (2014); Shah et al. (2014)). Ideally, the measure of a predictor variable's importance should not be affected by missingness and subsequent imputation. That is, the estimated importance of a predictor variable, when measured on the imputed dataset,

should be approximately the same, relative to other variables, as the importance estimate that would have been obtained if the complete data were available.

Numerous random-forest-based variable importance measures have been discussed in the literature. Most prominant is permutation importance (Breiman, 2003), which measures the loss in predictive accuracy when values for a predictor variable are randomly permuted. A less prominent measure, Gini Importance, (Breiman, 2001) determines the net decrease in impurity resulting from splits on a given predictor variable for classification problems. For datasets with missing values, Hapfelmeier et al. (2014b) developed an alteration to permutation importance that randomly allocates cases to nodes, rather than permuting values whenever a split is performed on the variable whose importance is being measured. Strobl et al. (2008) demonstrate that when permutation importance is used, variables that are correlated with important predictors tend to be considered important themselves, even if they do not add any new information beyond that provided by the correlated variable. Strobl et al. (2008) introduce a conditional variable importance measure that is intended to measure the value of adding a predictor variable when all other predictors are already included.

In the literature, there has been considerable attention given to developing random-forest-based imputation techniques and assessing the quality of the resulting imputed values. Tang and Ishwaran (2017) assess the performance of random-forest-based imputation techniques, finding these approaches to be generally robust. However, little attention has been given to the effect of imputation on measures of variable importance. Hapfelmeier et al. (2014b) introduce a self-contained variable importance measure that results from random forests grown in a way that handles missing values automatically. However, this procedure is intended to reflect the importance of a predictor variable given the data present, and missing observations negatively impact a predictor variable's importance estimate. When the objective is to assess the importance of a predictor variable if the variable had been able to be observed

completely, Hapfelmeier et al. (2014a) recommend using imputation prior to assessing variable importance, and present a case study comparing the results of one popular imputation technique to those obtained using their self-contained measure and a complete case analysis.

In this study, we assess the impact of six random-forest-based imputation techniques on Breiman's popular permutation importance measure of variable importance. Our objective is to assess the impact of imputation on the measure of an explanatory variable's importance, rather than assessing the accuracy of the imputed values themselves, as is done by Tang and Ishwaran (2017). We demonstrate that some imputation techniques lead to a reduction in the estimated importance of variables with many missing values, while other techniques, somewhat paradoxically, result in inflated estimates of importance for such variables. Inflated estimates of variable importance can be especially problematic, as they might lead a researcher to incorrectly conclude that an unimportant variable with many missing values is important.

We structure the remainder of the manuscript in the following way. In Section 3.2, we provide a description of the popular random forest permutation importance measure, an overview of missing value terminology, and summaries of each imputation technique considered. In Section 4.4.1, we present a simulation study that demonstrates the effect of each imputation technique on the permutation variable importance measure, accounting for different proportions of missingness and various correlations between predictor variables. In Section 3.4, we analyze the effect of imputation on the permutation variable importance measure using real datasets in which variable importance is likely to be of interest. Finally, we summarize our conclusions in Section 3.5.

3.2 Variable Importance and Missing Values

3.2.1 Permutation Importance

We begin this section with a discussion of the permutation variable importance measure. In a random forest, each tree is typically grown using a bootstrap sample of the training data, so some cases are not used in the process of growing any given tree. These are known as out-of-bag (OOB) cases and are useful for assessing predictive performance and variable importance.

Let $\mathscr{O}_t$ be the subset of indices corresponding to the OOB cases for tree t, and let $\mathbb{I}(\cdot)$ denote a generic indicator function. Let $|\cdot|$ denote the number of elements in a set. The process of calculating the variable importance for a predictor variable X_j can be summarized in the following steps.

1. Grow a random forest on the set of training data.
2. For $t = 1 \ldots, T$, perform the following steps.
   (a) For each $i \in \mathscr{O}_t$, predict the response for case i using tree t. Call this prediction $\hat{y}_{it}$.
   (b) For a regression problem, calculate $\text{MSE}_t = \frac{1}{|\mathscr{O}_t|} \sum_{i \in \mathscr{O}_t} (y_i - \hat{y}_{it})^2$. For classification problems, calculate $\text{MCR}_t = \frac{1}{|\mathscr{O}_t|} \sum_{i \in \mathscr{O}_t} \mathbb{I}(y_i \neq \hat{y}_{it})$.
   (c) Randomly permute the values of X_j for all OOB cases and predict OOB cases again. Call these predictions $\hat{y}_{it}^{(p)}$.
   (d) For a regression problem, calculate $\text{MSE}_t^{(p)} = \frac{1}{|\mathscr{O}_t|} \sum_{i \in \mathscr{O}_t} (y_i - \hat{y}_{it}^{(p)})^2$. For classification problems, calculate $\text{MCR}_t^{(p)} = \frac{1}{|\mathscr{O}_t|} \sum_{i \in \mathscr{O}_t} \mathbb{I}(y_i \neq \hat{y}_{it}^{(p)})$.
   (e) Calculate the difference in predictive performance, $\text{D}_t = \text{MSE}_t^{(p)} - \text{MSE}_t$ or $\text{D}_t = \text{MCR}_t^{(p)} - \text{MCR}_t$.

3. Average the values obtained in (2e) across all trees to obtain a overall variable importance score for X_j equal to $\frac{1}{T}\sum_{t=1}^{T} \mathrm{D}_t$.

If permuting the values does not harm predictive performance, then X_j is considered unimportant. On the other hand, if prediction accuracy decreases substantially, i.e. mean square error increases substantially, when values of X_j are permuted, this indicates that X_j is an important predictor. In classification problems, mean square error is replaced by misclassification rate.

The process is repeated for all predictor variables. Although an importance score by itself lacks a meaningful interpretation, scores for different variables can be compared in order to identify which variables are most important. This procedure can be used as a variable selection tool for high-dimensional datasets. Breiman (2003) recommends performing these calculations multiple times in order to account for random variability associated with growing random forests and permuting variable values.

3.2.2 Imputation of Missing Values

In the missing data literature, such as Rubin (1976), instances of missingness are typically classified as one of three types. Data are said to be *missing completely at random* (MCAR) if the probability of missingness does not depend on the observed or unobserved data. Data are said to be *missing at random* (MAR) if the probability of missingness depends only on observed data. Finally, data are defined as *missing not at random* (MNAR) if the probability of missingness depends on the missing values themselves. In this study, we focus on data that are missing completely at random. We show that even in this simplest scenario, variable importance measures can be heavily influenced by the choice of imputation technique. Although numerous imputation techniques can be used, we consider only approaches based on random forests. The techniques considered are described in the following subsections.

3.2.2.1 Proximity Weighted Approach (rfImpute)

Breiman suggested a proximity-based approach that imputes a missing value for a predictor variable X_j by taking a weighted average of the observed values for X_j, with the cases most similar to the case with the missing value of X_j receiving the heaviest weights. After an initial rough imputation is performed (e.g., median imputation) (Breiman, 2003), a random forest is used to calculate proximities between all training cases. The proximity between cases i and k is determined by the proportion of trees in which cases i and k land in the same terminal node. Missing values for a numeric predictor variable X_j are imputed using a proximity weighted average of all observed values. If X_j is categorical, a proximity weighted vote is taken using the responses of all observed cases, and the category receiving a plurality is taken as the imputed value. Another forest is then grown using the new imputed values, and the process is repeated iteratively. Breiman (2003) suggests performing at most six iterations. This approach is implemented using the `rfImpute` function in the `randomForest` package (Liaw and Wiener, 2002) in `R` (R Core Team, 2016). Breiman (2003) and Ishwaran et al. (2008) point out that estimates for OOB prediction error, obtained using `rfImpute`, typically overestimate test set error by 10%-20%. Since OOB error rate is used to estimate variable importance, this potentially results in a biased measure of variable importance when the imputed values are used.

3.2.2.2 Adaptive Tree Algorithm (rfsrc)

Ishwaran et al. (2008) propose imputing missing values as a random forest is grown, eliminating the need to perform imputation prior to assessing variable importance. Within a node, missing values for variable X_j are imputed by randomly drawing from the values of X_j among all observed cases in that node. Every time a node is split, missing values are re-imputed. When OOB cases are predicted, missing values are imputed by randomly

drawing from the observed values of training cases within a node. Only in-bag data are used, so out-of-bag variable importance measures are not affected by this approach.

Ishwaran et al. (2008) demonstrate that the accuracy of imputed values can be improved when imputation is performed iteratively. In the second and subsequent iterations, missing values are imputed by randomly drawing from observed values of cases in the same terminal node as the case for which imputation is performed, using the forest grown in the previous iteration. A random draw is made from each tree in the forest and new values are imputed by averaging or using a weighted vote. These techniques are implemented in the R package, `randomForestSRC` (Ishwaran et al., 2008; Ishwaran and Kogalur, 2007, 2016).

3.2.2.3 missForest

The `missForest` algorithm introduced by Stekhoven and Bühlmann (2012) imputes missing values for predictor X_j using random forest predictions for the missing values. First, a rough imputation is performed, and a random forest is grown, with X_j as the response variable, using cases for which X_j was observed. Cases with X_j missing are then predicted and the prediction is used as the imputed value. This is done for each predictor variable, and the process is iterated until the change in imputed values becomes sufficiently small. The `missForest` approach is implemented in the `missForest` package in `R` (Stekhoven and Bühlmann, 2012; Stekhoven, 2015).

3.2.2.4 Multiple Imputation Using Random Forest Predictions and Normal Distribution (CALIBERrfImpute)

One criticism of `missForest` is that it does not adequately account for variability in imputed values. This is discussed by Shah et al. (2014) who propose imputing values by drawing from a normal distribution centered at the value predicted by a random forest, to introduce variability in the imputed values. The variance of this normal distribution is set equal to

the mean square prediction error for out-of-bag cases. While imputed values obtained using `missForest` vary only because of Monte Carlo variability associated with construction of a random forest, the procedure of Shah et al. also incorporates random variability associated with the individual imputed values. Because of this, Shah et al. suggest performing this process multiple times in order to obtain more than one imputed dataset. The resulting imputed datasets can be used to assess the variability associated with imputation. This is known as multiple imputation, which is very popular in the imputation literature. Shah et al.'s technique is implemented in the `CALIBERrfimpute` package (Shah, 2014) in `R`.

3.2.2.5 Random Forest Multivariate Imputation using Chained Equations (RF-mice)

Doove et al. (2014) implement a multiple imputation approach similar to that of Shah et al., but instead of drawing from a normal distribution, after a random forest is grown, a single tree in the forest is selected and a single value of X_j is randomly drawn from the terminal node containing the case being imputed. The process is repeated for each predictor variable and performed iteratively. Again, multiple datasets are generated.

This procedure follows the Multivariate Imputation by Chained Equations (MICE) framework developed by van Buuren and Groothuis-Oudshoorn (2011). In this general framework, after a rough imputation is performed, missing values for X_j are imputed by fitting a model for the conditional distribution of X_j given all other predictor variables, using cases for which X_j was observed. This distribution is often modeled using linear or logistic regression models, although many modeling options are available. New imputed values for X_j are obtained by making a random draw from the estimated conditional distribution of X_j given all other predictors using the fitted model, and the process is performed iteratively. Because it might be difficult to specify a parametric model for the conditional distribution of X_j, Doove et al. suggest using a random forest instead and drawing directly from a terminal node. This algo-

rithm is implemented using the `mice.impute.rf` function in the `mice` package (van Buuren and Groothuis-Oudshoorn, 2011) in `R`.

3.2.2.6 Comparison of Imputation Techniques

The `missForest`, `RF-CALIBER`, and `RF-mice` algorithms all rely on predicting missing observations by growing a random forest on observed cases. While `missForest` simply uses the predicted value in the imputation, `RF-CALIBER` and `RF-mice` account for random variability by drawing from a normal distribution, or from the terminal nodes containing the case being predicted, respectively. These approaches allow for multiple imputation. On the other hand, `missForest` does not account for variability associated with individual values, instead using the estimated conditional mean instead. This is similar to `rfImpute`, which uses a proximity weighted average. When performed using only one iteration, the adaptive tree algorithm, implemented by `rfsrc`, makes random draws within each node, rather than just the terminal node, bringing even more random variability into the imputation process. If the algorithm is performed iteratively, draws are made from terminal nodes.

One major difference in the iterative `rfsrc` algorithm, when compared with `RF-CALIBER` and `RF-mice`, is that `rfsrc` grows random forests using the actual response variable, while the other two routines iteratively grow random forests on the predictor variables whose values are being imputed. While using the response can improve the accuracy of imputed values, it also creates a risk of information from the response variable leaking into the imputed values, thereby artificially strengthening the relationship between that predictor variable and the response variable in the imputed dataset. The `rfImpute` algorithm also uses the response variable to impute missing values. For the other techniques, the response could be included in the data matrix when imputation is performed iteratively, however we do not do this due to the leakage concern described here.

3.3 Simulation Study

3.3.1 Design

We proceed to investigate the impact of each imputation technique on the permutation importance scores for variables with missing data. Data were generated from a linear regression model (3.1). The response variable Y is a function of five predictor variables, drawn from a multivariate normal distribution, and a random error term. For the sake of comparison, a sixth predictor variable having no impact on Y was also generated. The model is

$$Y = 0.5X_1 + 0.4X_2 + 0.3X_3 + 0.2X_4 + 0.1X_5 + 0X_6 + \epsilon, \tag{3.1}$$

where $\boldsymbol{X} = [X_1, X_2, X_3, X_4, X_5, X_6] \sim \mathcal{N}_6(\mathbf{0}, \boldsymbol{\Sigma})$, with

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & \rho & \rho & \rho & \rho & \rho \\ \rho & 1 & \rho & \rho & \rho & \rho \\ \rho & \rho & 1 & \rho & \rho & \rho \\ \rho & \rho & \rho & 1 & \rho & \rho \\ \rho & \rho & \rho & \rho & 1 & \rho \\ \rho & \rho & \rho & \rho & \rho & 1 \end{bmatrix},$$

and $\epsilon \sim \mathcal{N}(0, 1)$. Because X_1 has the largest coefficient, it is reasonable to consider X_1 the most important predictor variable. Each X_j, $j \leq 5$ carries some intuitive importance, with importance decreasing as i increases. Since X_6 has no effect on Y, it can be considered unimportant.

For each value of $\rho = 0$, $\rho = 0.25$, $\rho = 0.5$, and $\rho = 0.75$, 500 different datasets were generated, each consisting of 1,000 observations from model (3.1). Permutation variable importance was calculated for each predictor variable using these completely observed datasets. Subsequently, for each dataset, a proportion of observations of X_1, denoted by p, were deleted

at random and imputed using each of the six imputation techniques described in Section 3.2. Permutation variable importance was calculated again on the imputed dataset. Values of 0, 0.1, 0.25, 0.5, and 0.75 were used for p. For each imputed dataset, the importance score of X_1 was divided by the sum of the importance scores for the six predictor variables to assess the importance of X_1 relative to all other variables. Negative importance scores were set to zero, prior to taking the sum. The same procedure was used to measure the effect of imputation on X_5.

Imputation using the `rfsrc` algorithm was performed using only one iteration and also using five iterations. For all techniques except the two `rfsrc` approaches, imputation was performed and a random forest was subsequently grown to assess variable importance, using the imputed values. This final random forest was grown using the `randomForest` package in `R`. The `rfsrc` approaches were implemented using the `randomForestSRC` package. Default settings were used for each approach. The number of trees was set to 300 for the forests used for imputation and 500 for forests used to measure variable importance. For approaches based on multiple imputation, five imputed datasets were generated for each originally simulated dataset, and and variable importance scores were calculated for each of the imputed datasets. These importance scores were then averaged. The choice of five imputed datasets is consistent with the suggestion of van Buuren and Groothuis-Oudshoorn (2011).

Figure 3.1 displays the mean relative importance scores for X_1, which were calculated based on the 500 different simulated datasets for each setting. Error bars, representing 95% confidence intervals are shown at each value of p, and are slightly staggered for aesthetic reasons. We first note that when $p = 0$, i.e., the data are fully observed, the relative importance of X_1 decreases as ρ increases. This is because the importance of the other predictors increases due to their correlation with X_1. This is consistent with the observations of Strobl et al. (2008).

When $\rho < 0.5$, it is clear that missing values lead to a decrease in the importance scores of X_1, regardless of which imputation technique is used. The lack of correlation between predictor variables makes it difficult or impossible to recover the lost information about X_1. As ρ increases, stronger correlations between X_1 and the other predictor variables allow for better imputation of the values of X_1, thereby restoring the relationship between X_1 and Y to a degree. The `rfImpute`, `missForest`, and iterated `rfsrc` algorithms appear better able to recover the importance of X_1 than the other methods. The performance of the `rfImpute` and iterated `rfsrc` algorithms is aided by the fact that Y is used when performing imputation. Although it appears beneficial in this instance, using the response variable in imputation is often disadvantageous when measuring variable importance, as will be seen in the case of X_5.

Figure 3.2 displays the relative importance scores for X_5 when its values are deleted and imputed. In this simulation, X_5 is the least important of the five predictor variables used to generate Y. The `rfImpute`, `missForest`, and iterated `rfsrc` algorithms result in higher importance estimates for X_5 on the imputed datasets than the importance estimates for X_5 obtained using the completely observed data. This discrepancy increases with the proportion of missing values and can be explained by either the use of the response variable in imputation, or the failure to account for randomness in the imputed values. When the response is used in imputation, as is done in the `rfImpute` and the iterative `rfsrc` algorithms, observations with similar responses become more likely to have similar imputed values for X_5. In this way, the response leaks into the imputed values, artificially increasing the importance of variables with many missing values. In comparison, the `missForest` technique simply uses the estimated conditional mean for X_5 in imputation. Since it does not account for the randomness associated with individual values of X_5, this technique overestimates the relationship between X_5 and Y. Although the variable importance measures from the imputed data might lead an observer to conclude that X_5 is quite important, the increase

in relative importance is really due to the information provided by other predictor variables, as well as the response, which were used in imputation. The problem becomes more severe as the correlation between the predictor variables increases.

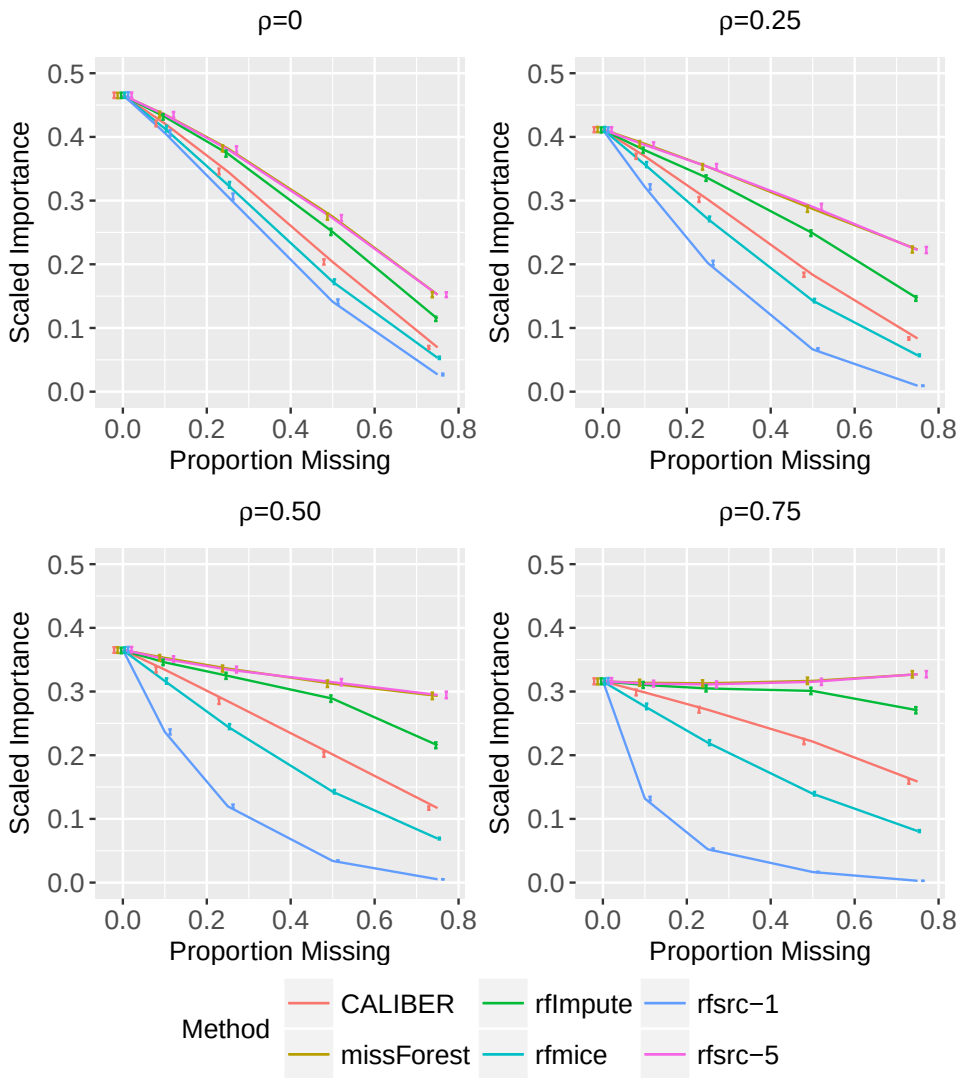

Figure 3.1: Relative importance scores are shown for X_1 as a function of the proportion of missing values for X_1. The importance of X_1 tends to decrease when a large proportion of values are missing. This is especially true when there is low correlation between the predictors variables.

On the other hand, the `rfsrc` algorithm, when performed without iteration, leads to the most cautious estimates of variable importance, as the estimated importance of X_5 is quickly downgraded due to missingness. The variable importance estimates obtained after `RF-mice` and `RF-CALIBER` are used for imputation most closely resemble the estimates obtained when all data are present. When predictor variables are mostly uncorrelated there is little difference in the performance of these two techniques. However, when the correlation is moderate to strong, variable importance estimates resulting from the use of `RF-CALIBER` are slightly higher than those resulting from `RF-mice`. For $\rho = 0.75$, `RF-CALIBER` results in slightly inflated estimates of variable importance, while `RF-mice` results in slightly deflated estimates. It is worth noting that because the data were generated from a normal distribution this represents a best case scenario for the `RF-CALIBER` approach, which assumes that the conditional distributions of predictor variables are normal when performing imputation.

Figure 3.3 displays the estimated relative variable importance for each variable after each imputation technique is applied in the simulation using $\rho = 0.75$. We see that when the proportion of missing values for X_5 approaches or exceeds 50%, `missForest` `rfImpute` and the iterated version of `rfsrc` result in X_5 being ranked ahead of X_3 and X_4, and in extreme situations, even X_2. Only the `CALIBER` approach maintains the proper ordering of the variables for all values of p. When `RF-mice` is used, X_5 slips to being approximately even with or, in the most extreme scenario, slightly behind X_6. When performed without iteration, `rfsrc` results in near zero estimates of importance for X_5, even when the proportion of missing values is small. Although X_6 was not used to generated Y, it is correlated with the other explanatory variables and thus has a nonzero importance estimate.

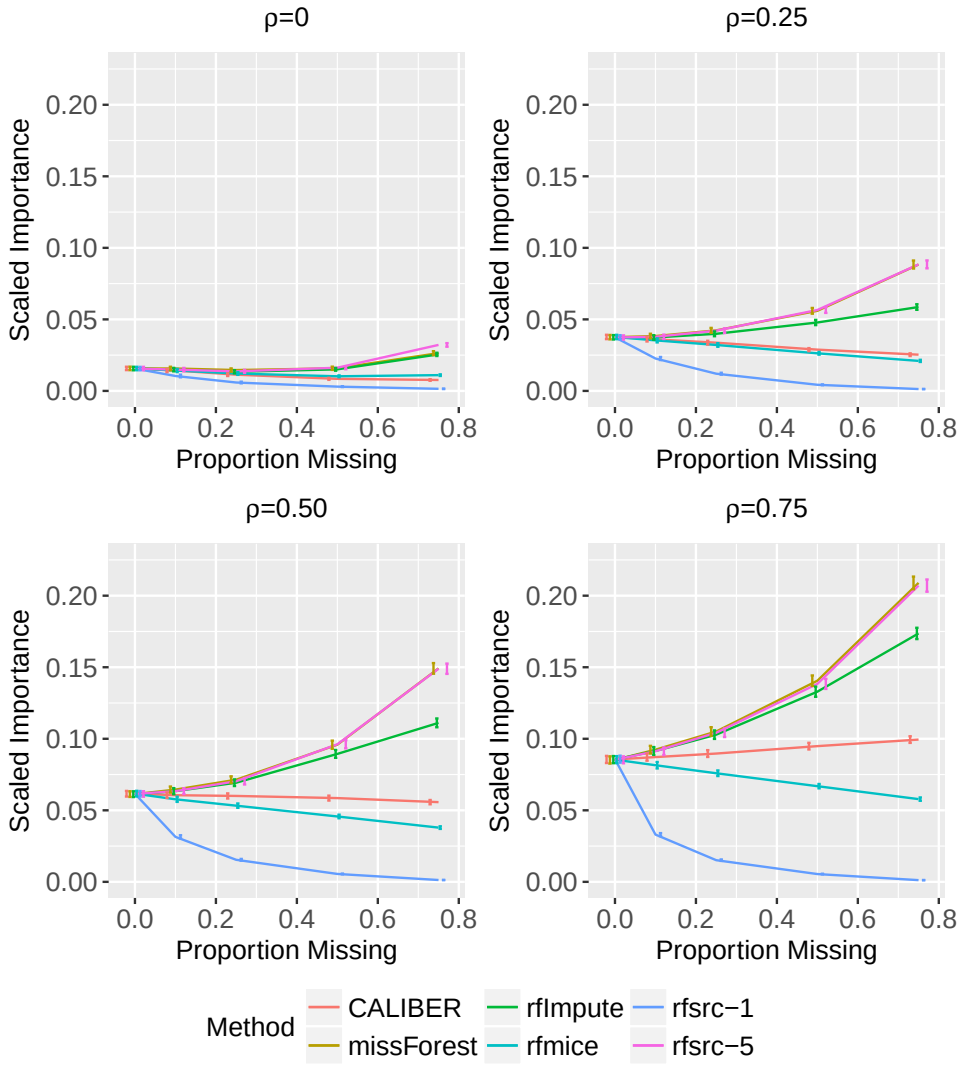

Figure 3.2: Relative importance scores are shown for X_5 as a function of the proportion of missing values for X_5.

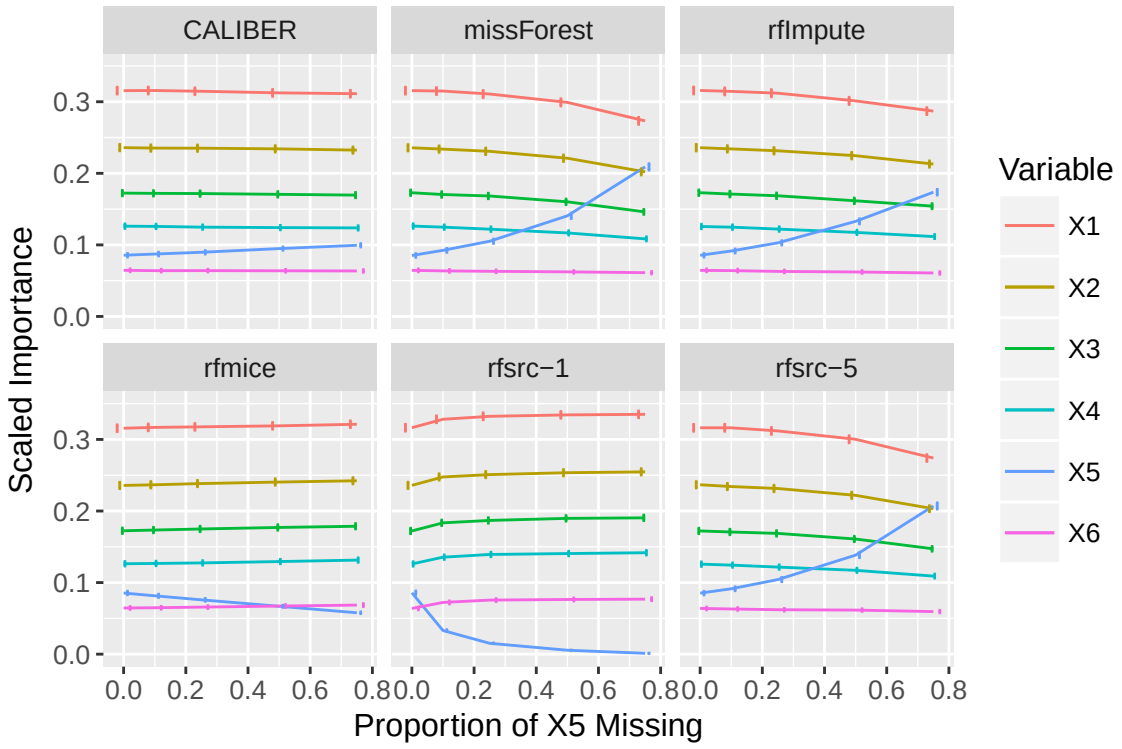

Figure 3.3: The estimated importance of each variable is shown when $\rho = 0.75$. Only `CALIBER` maintains the proper ordering. When a large proportion of X_5 values are imputed, three of the techniques lead to importance measures for X_5 that exceed those of X_3 and X_4.

3.4 Data Applications

We now consider the performance of each imputation technique using real datesets that pertain to classification and regression problems in which variable importance might be of interest. The first dataset contains information on students who declared a major in science, technology, engineering, or mathematics (STEM) during their first year at a large public university. In this classification problem, we seek to predict whether a student will switch to a non-STEM major during his or her first year in college. We consider 15 predictor variables, relating to a student's demographic information, academic background, and first

semester courses. Although the complete dataset is larger, we consider only 4,899 students for which complete data on these variables are available. The second dataset is the public Boston Housing dataset, which is available at the UCI machine learning repository (Lichman, 2013). This dataset pertains to a regression problem, where the response is the median value of owner-occupied homes in the suburbs of Boston. There are 506 observations and 13 predictor variables in the dataset.

In the STEM problem, we moniter the effects of imputation on estimated variable importance for three numerical variables, namely ACT score, number of high school science units, and Regent's Admissions Index (RAI) and one binary variable, participation in a learning community during the first semester at the university. Because RAI is calculated using information such as high school grades, ACT score, and high school courses, all of which are other predictor variables in the dataset, it is highly correlated with many other predictors. Multicollinearity involving other predictor variables in the STEM dataset is likely as well.

Variable importance for the complete STEM dataset was calculated using 100 different random forests. Unlike the simulation described in Section 4.4.1, each random forest was grown using the same training data so the only differences in importance scores are due to random variability associated with growing the forests. Using the complete data, RAI score was the most important predictor, while ACT score, learning community participation, and number of high school science units rank fourth, eight and twelfth, respectively.

Separately for each variable, we randomly deleted 10%, 25%, 50% and 75% of the values and imputed these values using each of the six random-forest-based imputation techniques. This was repeated 100 times for each proportion of missingness. Relative importance was calculated as described in Section 4.4.1. Figure 3.4 displays the relative importance scores for each of the four predictor variables as a function of the proportion of missing values.

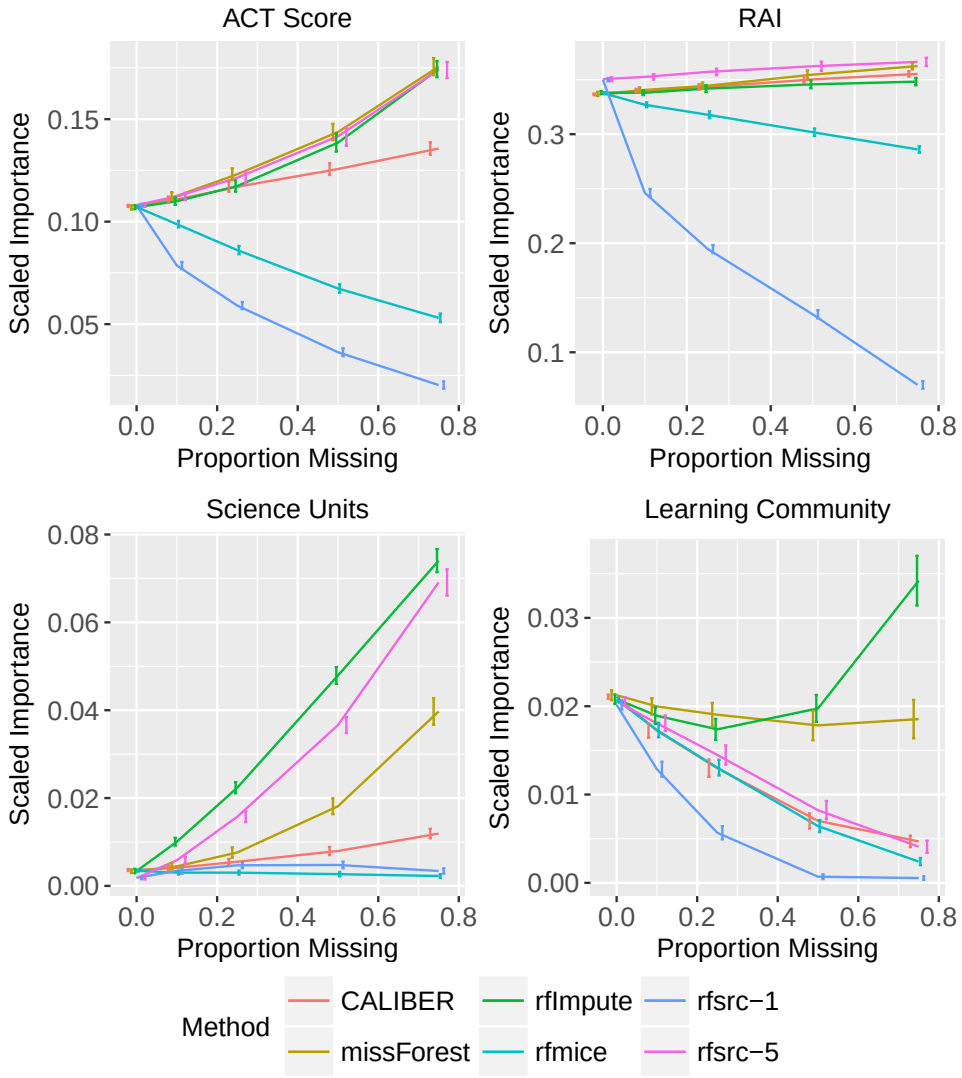

Figure 3.4: Variable importance ratings for ACT score, learning community participation, number of high school science units, and Regent's Admissions Index (RAI) score for the STEM dataset.

The `rfImpute`, `missForest`, iterated `rfsrc`, and `RF-CALIBER` algorithms result in increasingly inflated estimates of the importance of ACT score and high school science units as the proportion of missing values grows. The inflation resulting from the use of `RF-CALIBER` is not as severe as for the other approaches. This inflation is a result of information from

other important predictor variables, such as RAI score, or the response variable, leaking into the imputed values. Using the complete data, the number of high school science units ranks as the twelfth most important out of fifteen explanatory variables. However, this variable climbs to eighth in importance when 25% of values are deleted and imputed using `rfImpute` and fifth when 50% of values are deleted and imputed. The estimated importance of RAI score is only slightly inflated after imputation using these methods as it was already the most important variable and has less to gain from other variables being used in imputation. Again, the `rfsrc` algorithm, when performed without iteration, leads to diminished estimates of variable importance for variables with missing values. This is especially apparent for RAI and ACT scores, whose estimated importance deteriorates quickly when this imputation technique is used. The importance of the categorical learning community variable is deflated by most methods, with the exception of `rfImpute` when 75% of the values are missing.

As was seen in the simulation, the multiple imputation approaches come the closest to maintaining the variable importance estimates obtained when no data were missing. The `RF-CALIBER` approach leads to moderately inflated importance estimates for ACT score and slight inflation for science units. On the other hand, the `RF-mice` technique results deflated estimates of importance for ACT and RAI scores. The degree to which the estimates are deflated is substantially less severe than when the uniterated `rfsrc` algorithm is used.

The inflated measures of variable importance for ACT score and high school science units present a risk to those trying to draw conclusions from these data, as imputation of a large number of missing test scores might cause a university to conclude that ACT scores are of greater importance than GPA, when the complete data tell a different story. High schools, seeing the increased importance in high school science units, might be deceived into over-investing in this area.

In the regression example, based on the Boston Housing dataset, we again moniter estimated importance after imputation for three numeric variables, namely number of rooms, crime rate, and age of the home, along with a categorical variable indicating whether or not the property borders the Charles River. These variables ranked second, fourth, ninth, and twelfth in respective estimated importance when the complete data are used.

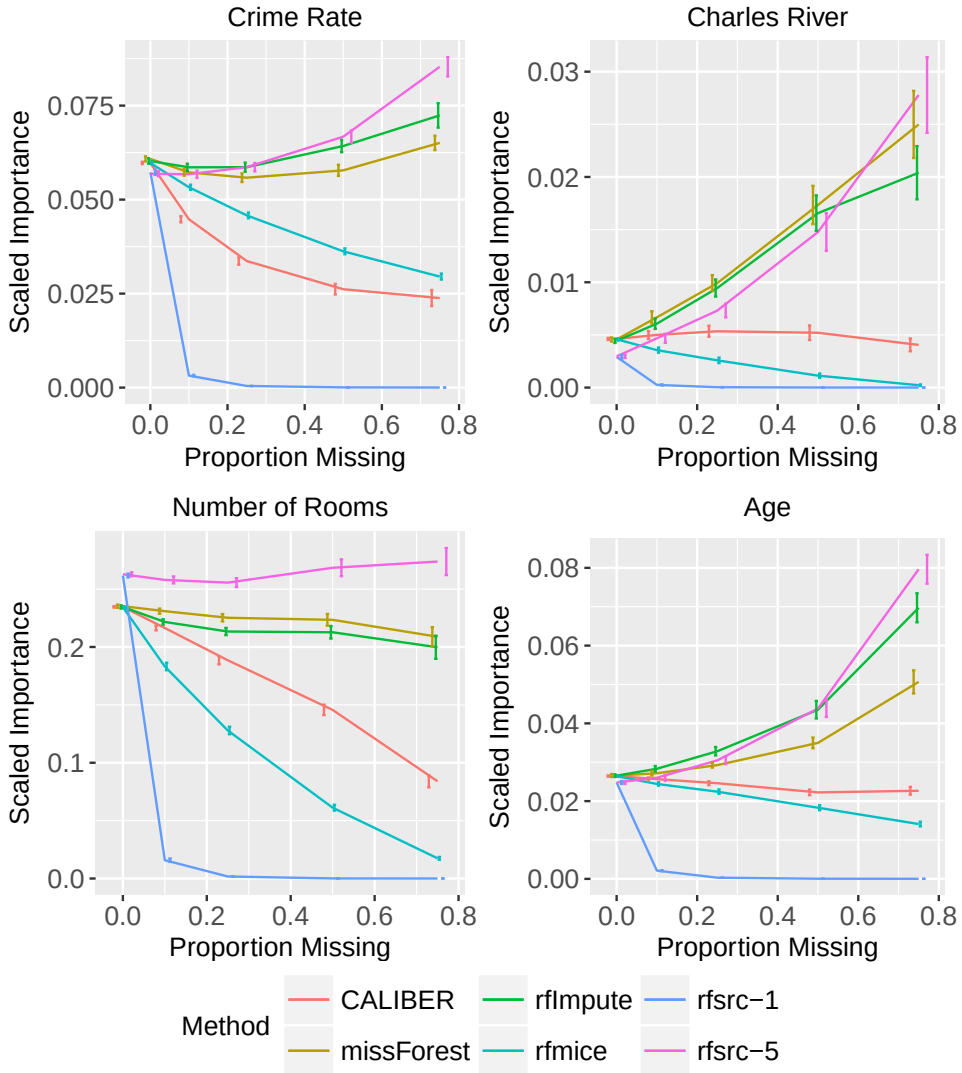

Figure 3.5: Variable importance ratings for crime rate, bordering the Charles River, number of rooms, and age in Boston Housing dataset, as the proportion of missing values increases.

Figure 3.5 illustrates the impact that each imputation technique has on the estimated importance of predictor variables when predicting house values. It is again apparent that the `rfImpute`, `missForest`, and iterated `rfsrc` algorithms lead to inflation of the estimated importance of the less important variables, namely crime rate, age, and bordering the Charles River. When the `rfsrc` algorithm is used without iteration, the perceived importance of variables contining missing values is decimated. Both multiple imputation techniques deflate the importance of all three numerical variables though not as drastically as the uniterated `rfsrc` algorithm. The `RF-CALIBER` imputation technique is largely able to achieve an accurate representation of the importance of bordering the Charles River, even when there are many missing data. The most important variable, number of rooms, sees its estimated importance diminished by the multiple imputation approaches and `rfsrc`, performed once. Because this variable is not strongly correlated with other predictor variables, information lost through missing data is difficult to recover, as was the case when values of X_1 were imputed in the simulation.

3.5 Conclusions

Ideally, variable importance measures obtained after imputation should closely resemble those resulting from the fully observed data. However, as we have shown, imputation often leads to either inflated or deflated measures of variable importance, depending on the technique used, the proportion of missing values, and the strength of the correlation between the variable being imputed and other predictor variables. No single imputation technique performs uniformly better than all others, with respect to assessing variable importance.

Overestimation of variable importance is common when imputation techniques use the response variable in the imputation process (`rfImpute`, iterated `rfsrc`) or fail to adequately account for variability in missing observations (`missForest`). This overestimation can be

severe for a variable with many missing values, especially when other predictor variables are highly correlated with the one whose values are missing. This phenomenon could cause users to erroneously conclude that a variable with many missing values is more important than a completely observed variable, potentially resulting in misappropriation of valuable resources. On the other hand, the `rfsrc` algorithm, when performed without iteration, consistently results in diminished estimates of the importance of variables with missing data.

In most situations, the multiple imputation techniques, `RF-CALIBER` and `RF-mice`, come closest to preserving the importance measures that would have been obtained if data could be completely observed. The `RF-mice` approach typically results in slightly deflated estimates of variable importance, while `RF-CALIBER` often results in slightly inflated estimates.

It is reasonable to expect that a lack of information should never lead us to conclude that a variable is more important than it would be when completely observed. This is consistent with the view articulated by Hapfelmeier et al. (2014b). Only the `RF-mice` algorithm, and the `rfsrc` algorithm, when performed without iteration, satisfy this criterion in all of the examples and simulations we considered. By using an iterated, multiple imputation approach, `RF-mice` is better able to recover at least some of the importance of a predictor with missing values than the uniterated `rfsrc` approach. The `RF-mice` algorithm falls within the respected multivariate imputation using chained equations multiple imputation framework, and our mostly encouraging results when applied to variable importance measures further illustrate the benefits of this approach.

Although we find `RF-mice` generally preferable to the other techniques for the purpose of assessing variable importance, `RF-mice` is not able to completely recover lost information relating to variable importance and typically underestimates the importance of variables with large amounts of missing data. This is especially true when the variable with missing data is truly very important, and is not highly correlated with other variables in the dataset. Such a situation presents a substantial challenge when one wishes to assess variable importance.

Future research focused on the development of variable importance measures that account for missing values is warranted.

Our analysis pertains only to the the impact of imputation on measures of variable importance. Imputation techniques that lead to an inflation or deflation of a variable's estimated importance might perform well for other purposes, such as generating a forest that provides accurate predictions of the response. It is important to consider a researcher's objective when deciding which imputation technique is most appropriate.

In this paper, only imputation techniques based on random forests were considered. These are but a small subset of all imputation techniques available. It would be reasonable to perform imputation using some other technique and then use random forests to assess variable importance. Other imputation techniques will likely lead to inflated or deflated measures of variable importance, just as we have seen for random-forest-based approaches. Before drawing conclusions, users should investigate the effect of the imputation technique on variable importance in a manner similar to the one described in this paper.

Bibliography

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Breiman, L. (2003). *Manual—setting up, and understanding random forests V4.0*.

Doove, L., Van Buuren, S., and Dusseldorp, E. (2014). Recursive partitioning for missing data imputation in the presence of interaction effects. *Computational Statistics & Data Analysis*, 72:92–104.

Hapfelmeier, A., Hothorn, T., Riediger, C., and Ulm, K. (2014a). Estimation of a predictor's importance by random forests when there is missing data: risk prediction in liver surgery using laboratory data. *The International Journal of Biostatistics*, 10(2):165–183.

Hapfelmeier, A., Hothorn, T., Ulm, K., and Strobl, C. (2014b). A new variable importance measure for random forests with missing data. *Statistics and Computing*, 24(1):21–34.

Ishwaran, H. and Kogalur, U. (2007). Random survival forests for R. *R News*, 7(2):25–31.

Ishwaran, H. and Kogalur, U. (2016). *Random Forests for Survival, Regression and Classification (RF-SRC)*. R package version 2.2.0.

Ishwaran, H., Kogalur, U., Blackstone, E., and Lauer, M. (2008). Random survival forests. *Annals of Applied Statistics*, 2(3):841–860.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

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

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, 63(3):581–592.

Shah, A. (2014). *CALIBERrfimpute: Imputation in MICE using Random Forest*. R package version 0.1-6.

Shah, A. D., Bartlett, J. W., Carpenter, J., Nicholas, O., and Hemingway, H. (2014). Comparison of random forest and parametric imputation models for imputing missing data using mice: a caliber study. *American Journal of Epidemiology*, 179(6):764–774.

Stekhoven, D. J. (2015). Missforest: nonparametric missing value imputation using random forest. *Astrophysics Source Code Library*.

Stekhoven, D. J. and Bühlmann, P. (2012). Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118.

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., and Zeileis, A. (2008). Conditional variable importance for random forests. *BMC Bioinformatics*, 9(1):1.

Tang, F. and Ishwaran, H. (2017). Random forest missing data algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 10(6):363–377.

van Buuren, S. and Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 45(3):1–67.

CHAPTER 4. A ROBUST RESIDUAL-BASED APPROACH TO RANDOM FOREST REGRESSION

A paper in preparation

Andrew J. Sage, Ulrike Genschel, and Dan Nettleton

Abstract

We introduce a novel robust approach for random forest regression that is useful when the conditional distribution of the response variable, given predictor values, is contaminated. Residual analysis is used to identify unusual response values in training data, and the contributions of these values are down-weighted accordingly. This approach is motivated by the robust fitting procedure first proposed in the context of locally weighted polynomial regression and scatterplot smoothing (Cleveland, 1979). We further demonstrate that tuning the parameter in the robustness algorithm using a weighted cross-validation approach is advantageous when contamination is suspected in training data. We conduct extensive simulations, comparing our method to existing robust approaches, some of which have not been compared to one another in prior studies. Our approach outperforms existing techniques on noisy training datasets with response contamination. While no approach is uniformly optimal, ours is consistently competitive with the best existing approaches for robust random forest regression.

4.1 Introduction

Data contamination occurs when some observations are the result of a data generating process different from the one under which the majority of the observations were obtained. Contamination might result, for instance, from an unknown change in machine settings, unidentified extraneous factors, or or mistakes in recording data. Contamination often produces outlying observations of the response variable in training data that negatively affect the predictions of new, uncontaminated cases. A large variety of robust regression techniques are commonly used to reduce the impact of outliers on inference and prediction (c.f. Huber (2011),Rousseeuw and Leroy (2005)).

Random forest methodology, introduced by Breiman (2001), uses ensembles of decision trees to make predictions in classification and regression settings. Random forests automatically handle nonlinear relationships and interactions among predictor variables, making them a popular approach for regression on large and complex datasets. In this work, we extend existing work on the impact of data contamination on random forest regression and propose an improvement that performs particularly well with noisy data containing substantial contamination in training data response values.

The localized nature of random forest predictions provides an automatic layer of robustness. Unless a new case has features similar to those of a training case with an outlying response, the outlier is not likely to have much impact on the prediction. This is a potential advantage over ordinary least-squares regression, for example, where as little as one observation can have an considerable influence on the fitted response surface.

Hamza and Larocque (2005) and Folleco et al. (2008) demonstrated that random forests are generally more robust than many other methods in classification problems. Still, modifications to splitting rules and aggregation methods have been shown to improve random forest robustness in regression problems (Brence and Brown, 2006; Galimberti et al., 2007;

Roy and Larocque, 2012). For example, Roy and Larocque (2012), (hereafter [RL]) found that robust aggregation approaches usually have a stronger impact on mitigating the effect of outliers than robust splitting.

Meinshausen (2006) (hereafter [M]) showed that in regression problems, a random forest predicted value $\widehat{y}$ for a given vector of covariates $\boldsymbol{x}$ can be expressed as a weighted average of the response values of all training cases where each weight itself is an average of the weights associated with the individual trees. [M] further demonstrated that these weights can be used to estimate the conditional distribution function of Y for a vector of covariates $\boldsymbol{x}$, thus allowing the estimation of any quantile of the conditional distribution function of Y given $\boldsymbol{x}$ in addition to the conditional mean. [RL] showed that the weighted median, i.e. the estimate for the 0.5 quantile of this conditional distribution, is more robust than the ordinary random forest estimate of $E(Y|\boldsymbol{x})$.

Li and Martin (2017) (hereafter [LM]) introduced a general loss function framework for random forest regression. They showed that Breiman's original random forest approach and Meinshausen's quantile regression forest can be viewed as special cases of this generalized approach, using squared error and quantile loss, respectively. In an effort to improve random forest robustness, [LM] utilized Huber (Huber, 2011) and Tukey bisquare (Mosteller and Tukey, 1977) loss functions in a general loss random forest framework. [LM] found that a pseudo Huber loss function (Charbonnier et al., 1997) performs favorably when there is contamination in the training data.

Our proposal to improve the robustness of random forest regression differs from the aforementioned approaches as follows. Rather than determining the weights used in the random forest prediction depending on the closeness of training cases' responses to the predicted value for a new case, as done by [LM], we establish the individual case weights according to the size of the corresponding training cases' residuals. This approach is akin to the robust fitting procedure proposed by Cleveland (1979) for locally weighted regression

and scatterplot smoothing (LOWESS). Training cases with large residuals receive smaller weights to reduce their influence on the prediction. We refer to the proposed method as RF-LOWESS.

Because the RF-LOWESS weighting adjustment is determined using only residuals from training data, scaling factors are calculated just once and then can be applied in the predictions of all new test cases. This potentially improves efficiency when compared with Li & Martin's approach, in which the training case weights are adjusted differently for each new case being predicted.

We further generalize the LOWESS method by treating a value typically taken to be constant as a tuning parameter. We introduce a procedure for setting this and potentially other random forest tuning parameters, via cross-validation, when contamination is suspected in training data but not test data. This makes RF-LOWESS a highly flexible procedure, capable of achieving strong performance on noisy, contaminated datasets, while also self-correcting to ordinary random forest predictions when contamination is not a concern. Our study includes a thorough comparison of the performance of RF-LOWESS and several existing approaches for robust random forest regression ([RL], [M], [LM]) some of which have not been directly compared in prior literature.

The remainder of the manuscript is structured in the following manner. In Section 4.2, we provide an overview of existing approaches for robust random forest regression. In Section 4.3, we present a detailed description of the newly proposed RF-LOWESS algorithm and give an example to demonstrate its potential benefits. Section 4.3 further includes a description of the aforementioned parameter tuning procedure. In Section 4.4, we compare the performance of RF-LOWESS and other robust random forest regression techniques, using simulated and real data. Finally, we summarize our findings in Section 4.5.

4.2 Background

4.2.1 Random Forest Background

Random forests are ensembles of decision trees, which are grown by recursively performing binary splits on training data. Among numerous splitting techniques that have been proposed, the Classification and Regression Tree Algorithm (CART) (Breiman et al., 1984) is popular. In regression problems, this approach determines the best split for a node by minimizing the sum of the squared deviations between each response and the mean response in the corresponding node.

The trees in a random forest differ due to randomness that is injected in two ways. First, each tree is grown using a different bootstrap sample of the training data. Training cases that are not part of the bootstrap sample used to grow a tree are referred to as *out-of-bag* (OOB) cases. These cases can be used to assess performance in a manner similar to cross-validation. Second, a different randomly selected subset of predictor variables is considered for each possible split. When a prediction is made for a new case, the case is moved through each tree in accordance with the values of its explanatory variables and the splitting rules determined by the training data. Once a new case reaches a terminal node, a prediction is made by taking the mean of response values for training cases in that node. An overall random forest prediction is obtained by averaging predictions across trees.

4.2.2 Prior Robust Aggregation Approaches

Several modifications to the way random forest predictions are aggregated within and across trees have been proposed ([RL], [M], [LM]). These have been shown to improve random forest robustness when training data contamination is present. In this section, we give a brief overview of these techniques.

4.2.2.1 Aggregation via Median

[RL] show that using the median, rather than the mean, when aggregating predictions from individual trees leads to more robust predictions when training data response contamination is prevalent. Further improvement is possible if individual tree predictions are determined using median values within terminal nodes, instead of mean values.

4.2.2.2 Regression Based on Ranks

[RL] consider replacing the true response values for training cases with their ranks when growing the forest. Each tree is used to predict the rank of the new case. The training response value associated with the median predicted rank over all trees serves as the predicted value. Although an improvement over ordinary random forest regression in situations involving response contamination, [RL] do not find this approach to yield optimal prediction error in any of their simulations and mention that it might be "too robust." Therefore, we do not consider this approach in our investigation.

4.2.2.3 Quantile Regression Forest (QRF)

[M] showed that a random forest prediction for a new case $(\boldsymbol{x}, Y)$ can be written as a weighted average of the response values of all training cases. Let $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$ denote a set of training data of size n, and let T be the number of trees grown in a forest. For each tree $t = 1, \ldots, T$ and new case $\boldsymbol{x}$, let $N_t(\boldsymbol{x})$ denote the collection of indices of training cases lying in the same terminal node as $\boldsymbol{x}$. That is,

$$N_t(\boldsymbol{x}) = \{i : (\boldsymbol{x}_i, y_i) \text{ is in the same terminal node as } \boldsymbol{x} \text{ for tree } t \ , i = 1, 2, \ldots n\}. \quad (4.1)$$

Let $b_t(i)$ denote the number of times training case i occurs in the bootstrap sample used to grow tree t, and let $\mathbb{1}(\cdot)$ represent a generic indicator function. Then for $i = 1, 2, \ldots, n$, the weight of training case i in the prediction of Y given $\boldsymbol{x}$ is given by

$$w_i(\boldsymbol{x}) = \frac{1}{T}\sum_{t=1}^{T}\frac{b_t(i)\mathbb{1}(i \in N_t(\boldsymbol{x}))}{\sum_{j=1}^{n} b_t(j)\mathbb{1}(j \in N_t(\boldsymbol{x}))}, \tag{4.2}$$

and the predicted value is

$$\widehat{y}_{RF}(\boldsymbol{x}) = \sum_{i=1}^{n} w_i(\boldsymbol{x}) y_i. \tag{4.3}$$

[M], further showed that the α-quantile of the conditional cumulative distribution function $F(y|\boldsymbol{X} = \boldsymbol{x})$ can be estimated by

$$\widehat{Q}_\alpha(\boldsymbol{x}) = \inf\{y : \sum_{i=1}^{n} w_i(\boldsymbol{x})\mathbb{1}(Y_i \leq y) \geq \alpha\}.$$

The `quantregForest` package (Meinshausen, 2016) in `R` (R Core Team, 2016) can be used to estimate any quantile of the conditional distribution of Y given $\boldsymbol{X} = \boldsymbol{x}$ in this manner. [RL] showed that the 0.5 quantile of this conditional distribution(i.e., a weighted median) is a robust predictor that achieves strong performance when training data responses are contaminated.

4.2.2.4 Huber-Loss-Based Forest

[LM] showed that for a new case $\boldsymbol{x}$, with random forest weights $w_i(\boldsymbol{x}), i = 1, \ldots, n$, the random forest prediction given in (4.3) satisfies

$$\widehat{y}_{RF}(\boldsymbol{x}) = \underset{\lambda \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} w_i(\boldsymbol{x})(y_i - \lambda)^2,$$

and the quantile regression forest estimator [M] satisfies

$$\widehat{y}_{QRF}(\boldsymbol{x}) = \underset{\lambda \in \mathbb{R}}{\operatorname{argmin}} \sum_{i=1}^{n} w_i(\boldsymbol{x}) \rho_\tau(y_i - \lambda),$$

where $\rho_\tau(z) = z(\tau - \mathbb{1}_{\{z<0\}})$ corresponds to the τ-th quantile loss function.

[LM] introduced an extension to general loss functions (GL), resulting in a locally weighted estimator of the form

$$\widehat{y}_{GL}(\boldsymbol{x}) = \underset{s \in \mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n} w_i(\boldsymbol{x}) \phi(y_i, s(\boldsymbol{x}_i)), \tag{4.4}$$

where $\mathscr{F}$ is a family of functions and $\phi(\cdot)$ is a general loss function.

In order to improve robustness, [LM] implemented the pseudo-Huber loss function (Charbonnier et al., 1997) given by

$$L_\delta(y) = \delta^2 \left(\sqrt{1 + \left(\frac{y}{\delta}\right)^2} - 1 \right).$$

The estimating equation

$$\sum_{i=1}^{n} w_i^{pH}(x) \left(\hat{y}^{pH} - y_i \right) = 0, \tag{4.5}$$

with

$$w_i^{pH}(\boldsymbol{x}) = \frac{w_i(\boldsymbol{x})}{\sqrt{1 + \left(\frac{\hat{y}_{pH}(\boldsymbol{x}) - y_i}{\delta} \right)^2}},$$

has a solution given by the (psuedo) Huber estimator

$$\widehat{y}^{(pH)}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} w_i^{pH}(\boldsymbol{x}) y_i}{\sum_{i=1}^{n} w_i^{pH}(\boldsymbol{x})}. \tag{4.6}$$

Algorithm 3, which is given in the appendix gives Li & Martin's procedure for interatively approximating the solution to equation 4.5.

In addition to the pseudo Huber estimator, [LM] considered a similar estimator based on Tukey's bisquare function (Mosteller and Tukey, 1977), but found in their simulations that the pseudo Huber estimator achieves stronger predictive performance. Therefore, we use the pseudo Huber function when applying Li & Martin's approach. Li & Martin refer to their approach as a *Huber forest*, and we will use that name for the remainder of the paper.

4.2.3 Prior Robust Splitting

In addition to modifying aggregation approaches, random forest robustness can also be improved through changes to the procedure for partitioning training cases into nodes. Breiman et al. (1984) suggested performing splits that minimize the sum of the absolute deviations from the median in each resulting node (LAD), rather than minimizing the sum of the squared deviations from the mean (LS). In addition to LAD and LS, Galimberti et al. (2007) implemented the Huber (Huber, 2011) and Tukey bisquare (Mosteller and Tukey, 1977) functions in the splitting process, and Bhat et al. (2015) considered splits using quantile loss. These robust splitting criteria are computationally more expensive than LS splitting (Bhat et al., 2015; Torgo, 1999). [RL] found that while LAD splitting leads to improvements in robustness, robust aggregation usually has a stronger impact than robust splitting. Still, any of the aggregation approaches we discuss can be applied using either LAD or LS splitting, and we investigate improvement obtained by using LAD splitting with each aggregation approach in Section 4.4.1.

4.3 RF-LOWESS Method

4.3.1 Motivation and Algorithm

We propose a new method for modifying random forest training case weights $w_i(\boldsymbol{x})$, in the prediction of Y, given $\boldsymbol{x}$. The approach is based on the size of the residual associated

with each training case, rather than the proximity of the training value y_i and predicted value of y. In our approach, training cases with large residuals are down-weighted, and the weights of the other training cases are adjusted accordingly.

Instead of using predictions resulting from a weighted least square approach, as is done in LOWESS (Cleveland, 1979), RF-LOWESS uses the random forest prediction weights as a starting point. These weights are then modified in accordance with the size of each training case's residual. Residuals are calculated using OOB predictions, eliminating the possibility of a training case heaviliy influencing its own prediction. We apply Tukey's bisquare function, which is defined as

$$B(t) = \begin{cases} (1-t^2)^2 & \text{if } |t| < 1 \\ 0 & \text{if } |t| \geq 1 \end{cases} \tag{4.7}$$

to the ratio of the residual of case i and αm, where m denotes the median of all absolute residual values and α denotes a constant. Note that in the analogous step of the LOWESS algorithm (Cleveland, 1979), the value of α is automatically set to 6. We will demonstrate in Section 4.3.3 that α can be considered a tuning parameter, thereby making RF-LOWESS more flexible, and we will illustrate the benefits of this added flexibility in Section 4.4.2.

We denote the OOB prediction for training case j as $\widehat{y}_{OOB}(\boldsymbol{x}_j)$. Let $(\boldsymbol{x}_j, y_j)$ be a training case and let $\mathscr{T}_j$ represent the index set for trees grown from bootstrap samples not including case j;

$$\mathscr{T}_j = \{t : b_t(j) = 0, t = 1, 2, \ldots, T\}.$$

We use $|\mathscr{T}_j|$ to denote the cardinality of set $\mathscr{T}_j$. Then the weight of training case i, $i \neq j$ on the OOB prediction for case j is given by

$$w_{OOB_i}(j) = \frac{1}{|\mathscr{T}_j|} \sum_{t \in \mathscr{T}_j} \frac{b_t(i)\mathbb{1}(i \in N_t(\boldsymbol{x}_j))}{\sum_{i=1}^{n} b_t(i)\mathbb{1}(i \in N_t(\boldsymbol{x}_j))},$$

where N_t is as defined in 4.1.

Note that it is necessary to define these weights and the corresponding out of bag predictions as functions of j, rather than $\boldsymbol{x}_j$, because two training cases with equivalent covariate vectors are unlikely to have precisely the same OOB weights. For example, even if $\boldsymbol{x}_1 = \boldsymbol{x}_2$, $\mathscr{T}_1$ is unlikely to equal $\mathscr{T}_2$, so different subforests are used to determine the OOB weights for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. This is not an issue when discussing weights associated with new test cases, where training case weights are written as functions of $\boldsymbol{x}$. We will use these OOB weights to make initial predictions in Step 1 of the proposed RF-LOWESS algorithm, given in Algorithm (1).

In rare instances, Algorithm 1 might fail to achieve convergence. This is usually the result of having cases with large positive and large negative residuals in close proximity to one-another, causing the λ-values associated with these and possibly other cases to alternate between different numbers without converging. In these instances, the λ-values associated with most training cases do converge, and we have found that this phenomenon has very little impact on predictions. Like Cleveland (1979) suggested with respect to his original LOWESS algorithm, we find it reasonable to stop the RF-LOWESS algorithm after a fixed number of iterations, and ten iterations appear to be sufficient. Alternatively, in situations where the algorithm does not converge, users might choose to use the set of values $\left\{\lambda_i^{(l)}\right\}_{i=1}^{n}$ from the iteration l that resulted in the smallest value of $m^{(l)}$.

Algorithm 1 RF-LOWESS Algorithm

1. Grow a random forest and for all $j = 1, 2, \ldots, n$, calculate OOB prediction weights $w_{OOB_i}(j)$. Make initial OOB predictions

$$\widehat{y}_{OOB}^{(1)}(j) = \sum_{i=1}^{n} w_{OOB_i}(j) y_i, \text{ for } i = 1, 2 \ldots, n.$$

2. Set $l = 1$ and perform the following steps:

   i. Calculate residuals $e_j^{(l)} = y_j - \widehat{y}_{OOB}^{(l)}(j)$ for $j = 1 \ldots n$.

   ii. Set $m^{(l)} = \text{Median} \left\{ \left| e_j^{(l)} \right| \right\}_{j=1}^{n}$.

   iii. Let

$$\lambda_j^{(l)} = B\left(\frac{e_j^{(l)}}{\alpha m^{(l)}} \right),$$

   for $j = 1, 2, \ldots, n$, where $\alpha \in \mathbb{R}^+$ is a tuning parameter discussed in Section 4.3.3 and B denotes Tukey's bisquare function.

   iv. For $j = 1, 2, \ldots n$, let

$$\widehat{y}_{OOB}^{(l+1)}(j) = \frac{\sum_{i=1}^{n} \lambda_i^{(l)} w_{OOB_i}(j) y_i}{\sum_{i=1}^{n} \lambda_i^{(l)} w_{OOB_i}(j)}.$$

   v. If

$$\frac{1}{n} \sum_{j=1}^{n} \left(\widehat{y}_{OOB}^{(l+1)}(j) - \widehat{y}_{OOB}^{(l)}(j) \right)^2 \leq \epsilon_0,$$

   (a non-negative user-specified convergence parameter), stop and return $\lambda_j = \lambda_j^{(l+1)}$ for $j = 1, 2, \ldots, n$. Otherwise, set $l = l + 1$ and repeat steps (i)-(v)

Algorithm 1 (continued)

3. For a new case $\boldsymbol{x}$, with random forest prediction weights $\{w_i(\boldsymbol{x})\}_{i=1}^n$ the RF-LOWESS predicted value $\widehat{y}_{RFL}(\boldsymbol{x})$ is then given by

$$\widehat{y}_{RFL}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} \lambda_i w_i(\boldsymbol{x}) y_i}{\sum_{i=1}^{n} \lambda_i w_i(\boldsymbol{x})}. \tag{4.8}$$

The RF-LOWESS prediction in (4.8) can be interpreted as a weighted average of all training case response values, where the weights are determined by two factors. The factor $w_i(\boldsymbol{x})$ captures the proximity between test case $\boldsymbol{x}$ and training case i. The factor λ_i captures the degree to which training case i is down-weighted due to the size of its OOB residual. While $w_i(\boldsymbol{x})$ depends on both the training and test cases, λ_i is determined entirely by OOB residuals for training cases. Therefore, $\lambda_1, \ldots, \lambda_n$ are calculated just once regardless of the number of test cases for which predictions are sought. This is in contrast to the approach of [LM] in which local prediction weights are calculated iteratively for each new case.

4.3.2 Illustrative Example

In this section, we will consider a basic example that is intended to illustrate an important difference between the RF-LOWESS and Huber forest approaches. Although random forests are rarely used in situations as simple as the example in this section, the issues we will discuss here are relevant for larger, more complex datasets, such as the ones we will consider in Section 4.4.

In our example, the expected response is a nonlinear function of a single explanatory variable. A random error term is added to each expected response, with 90% of the errors

coming from a normal distribution with mean 0 and standard deviation 0.1, and 10% of the errors coming from a contaminating distribution with a larger standard deviation of $\sigma = 0.5$.

The model is

$$y_i = \sin(x_i) + \epsilon_i \mathbb{1}(r_i \geq 0.1) + \gamma_i \mathbb{1}(r_i < 0.1), \tag{4.9}$$

where $\epsilon_i \sim \mathcal{N}(0, 0.1), \gamma_i \sim \mathcal{N}(0, 0.5)$, $r_i \sim$ Uniform$(0, 1)$, and all ϵ_i, γ_i, and r_i values are independent. A total of 300 datapoints were generated with the $x_1, x_2, \ldots, x_n$ drawn independently from a uniform distribution on the interval (-3,3).

Figure 4.1 displays the data along with the true expected response curve, and the expected response curve estimated using a random forest with default settings. We see that traning cases 79 and 81, which appear to be potential outliers, pull the random forest estimate below the true response curve near $x = 1.5$. Likewise, case 85 appears to pull the estimated response curve upward near $x = 2$.

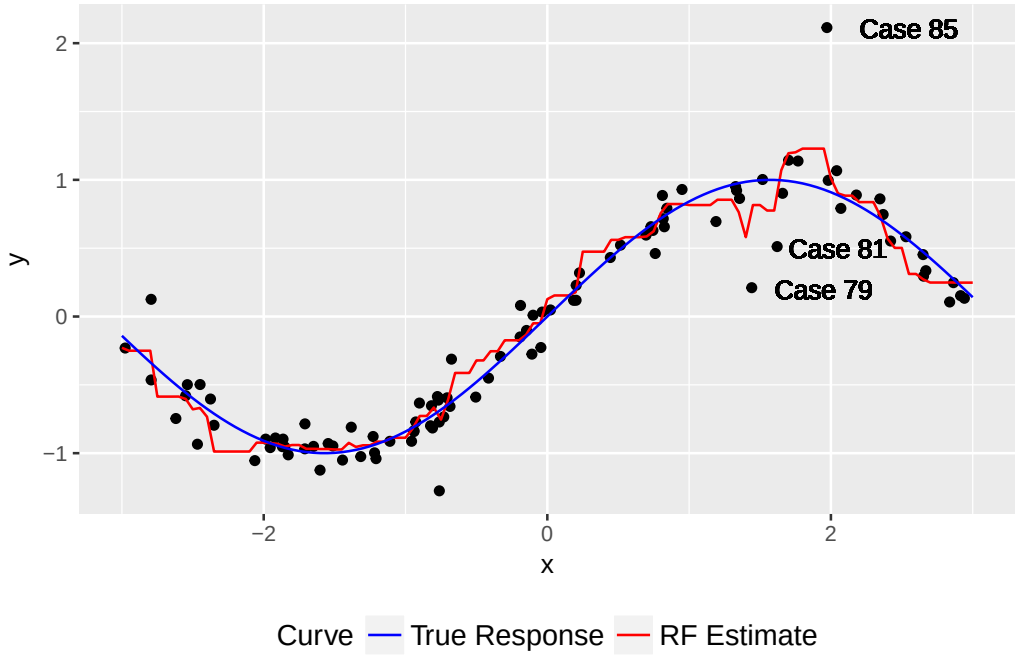

Figure 4.1: True and estimated response surface using random forest.

Suppose we are interested in estimating the conditional expection of y when $x = 1.4$. Table 4.1 gives the five observations with the largest random forest weights for a prediction when $x = 1.4$. Cases 79 and 81, account for more than half of the random forest prediction weight. These two cases have OOB residuals of -0.370 and -0.264, respectively, which are considerably larger in absolute value than the median absolute value of the OOB residuals, $m^{(1)} = 0.065$.

Table 4.1: Random forest weights and residuals for cases contributing most heavily to estimation of $E(y|x = 1.4)$.

i	x_i	y_i	$w_i(1.4)$	$e_i^{(1)}$
79	1.442	0.211	0.388	-0.370
81	1.621	0.512	0.176	-0.264
80	1.518	1.003	0.111	0.187
82	1.659	0.902	0.055	-0.168
77	1.335	0.922	0.048	0.088

Using RF-LOWESS, with $\alpha = 6$,

$$\lambda_{79}^{(1)} = B\left(\frac{e_{79}^{(1)}}{\alpha m^{(1)}}\right) = B\left(\frac{-0.370}{6(0.065)}\right) = B(-0.949) \approx 0.010.$$

After recalculating OOB predictions and residuals iteratively, we obtain $\lambda_{79} \approx 0.014$. Then, the RF-LOWESS weight for training case 79 is given by

$$\frac{\lambda_{79} w_{79}(1.4)}{\sum_{i=1}^{n} \lambda_i w_i(1.4)} \approx \frac{0.014(0.388)}{0.475} \approx 0.012.$$

Due to the size of its OOB residual, RF-LOWESS has decreased the weight of training case 79 from 0.388 to 0.012. Similarly, the weight of training case 81 is decreased from

0.176 to 0.131. Other training cases with smaller OOB residuals contribute more heavily to make up for the weight lost by down-weighting the potential outliers. As a result, the RF-LOWESS estimate of 0.860 is closer to the true expected response of 0.985 than the random forest estimate of 0.581.

The Huber forest estimator, given in (4.6) and calculated using Algorithm 3, recalculates predictions based on the difference between y_i and $\hat{y}_{pH}(1.4)$. Using the initial random forest prediction of $\hat{y}_{pH}(1.4) \approx 0.581$, as a starting value in Algorithm 3, the training case for which the quantity

$$\sqrt{1 + \left(\frac{0.581 - y_i}{\delta}\right)^2}$$

is minimized is $i = 81$. Thus, the Huber forest algorithm increases the weight of training case 81, instead of decreasing it, as RF-LOWESS does. Table 4.2 gives RF-LOWESS and Huber forest weights for the training cases in Table 4.1. The value of $\alpha = 6$ was used in RF-LOWESS because it is the used in the the original LOWESS algorithm Cleveland (1979). The value of $\delta = 0.10$ was used in the Huber forest algorithm because it minimizes the sum of the squared differences between the true expected response and Huber forest estimate when predictions are made for all values of x between -3 and 3, using increments of 0.05.

While RF-LOWESS down-weights cases 79 and 81 due to their large residuals, the Huber forest increases the weight placed on case 81, while down-weighting other training cases with smaller residuals. As a result, the Huber forest estimate of 0.532 is farther from the expected response than the original random forest estimate. Figure 4.2 illustrates the estimated curve for $E(Y|X = 1.4)$ using the ordinary random forest (RF), RF-LOWESS, and the Huber forest (HF) methods.

Table 4.2: Case weights for estimating $E(y|x = 1.4)$, using RF, RF-LOWESS, and Huber forest.

i	x_i	y_i	$w_i(1.4)$	$e_i^{(1)}$	λ_i	RF-LOWESS weight	Huber weight
79	1.442	0.211	0.388	-0.370	0.014	0.012	0.252
81	1.621	0.512	0.176	-0.264	0.353	0.131	0.489
80	1.518	1.003	0.111	0.187	0.924	0.215	0.050
82	1.659	0.902	0.055	-0.168	0.943	0.109	0.031
77	1.335	0.922	0.048	0.088	0.987	0.099	0.026

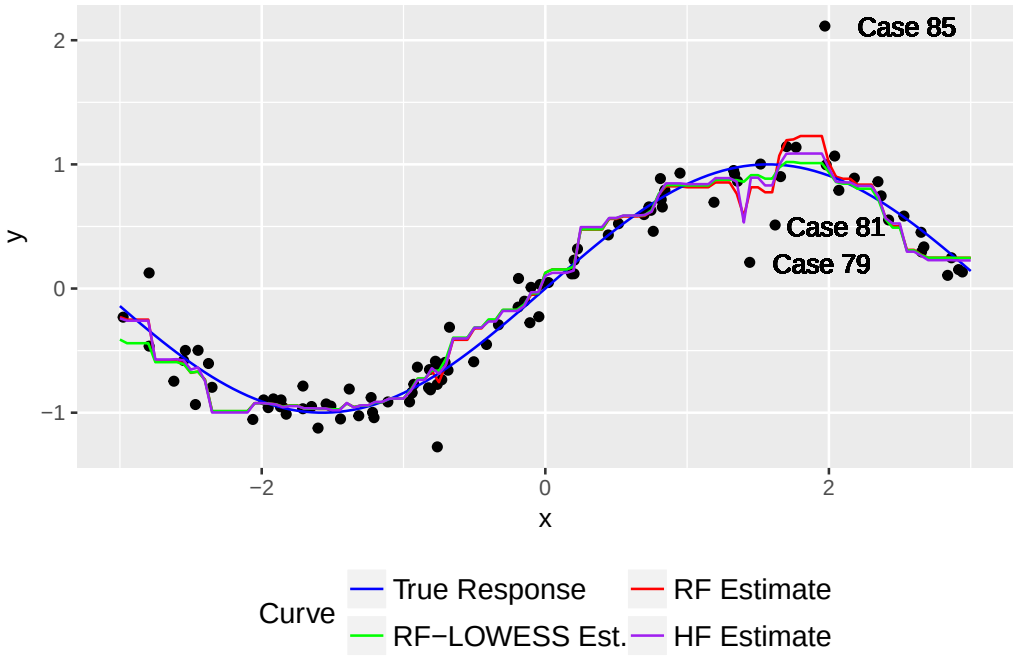

Figure 4.2: True and estimated response surfaces using random forest, RF-LOWESS, and Huber forest.

The example in this section is merely intended to illustrate the RF-LOWESS method and shed light on an important difference between the ways that RF-LOWESS and Huber forest calculate prediction weights. Although we have illustrated a scenario in which RF-LOWESS performs favorably, we do not claim, based on this example, that RF-LOWESS is always preferable to the Huber forest. The performances of these and other approaches are investigated thoroughly in Section 4.4.

4.3.3 Parameter Tuning

We now turn our attention to the parameter α. Cleveland (1979) used the value $\alpha = 6$ in the LOWESS algorithm. Rather than assigning a predetermined value, we treat α as a tuning parameter. Intuitively, the optimal value of α should vary depending on the amount of contamination in the training data. Small values of α lead to aggressive down-weighting of suspected outliers, while large α results in little change to ordinary random forest prediction weights. As $\alpha \to \infty$, $\lambda_i \to 1$ for $i = 1, 2, \ldots, n$, making RF-LOWESS predictions equivalent to ordinary random forest predictions. Therefore, small values of α are desirable in situations with substantial training data contamination, while large α values are preferable when robustness is of little concern.

In machine learning, optimal values for tuning parameters are usually determined using cross-validation. Training data are partitioned into k sets, referred to as folds, and each fold is withheld, one at a time. A method is trained on the remaining $k - 1$ folds and is used to predict cases in the fold that was withheld. Performance is evaluated by minimizing a loss function, such as mean square prediction error. This procedure can be performed repeatedly if desired.

Cross-validation is based on the assumption that training and test data come from the same distribution, making it unreliable when contamination is present in training data but not in the test data. In order to minimize the impact that outlying response values in the

data withheld for evaluation have on the choice of α, we use a weighted loss function. Cases in the withheld set with large residuals are weighted less heavily when calculating loss than cases with small residuals.

Let $\mathcal{I}$ represent a set of candidate values under consideration for α. The steps to be performed within each fold of a cross-validation procedure are given in Algorithm 2. We assume that the training data have been partitioned into two disjoint sets D_1 and D_2. In k-fold cross validation, D_1 is made up of $k-1$ subsets of the training data, and D_2 consists of the remaining subset that was witheld.

Algorithm 2 Weighted Cross-Validation Tuning Algorithm

For disjoint subsets of training data, D_1 and D_2:

1. Grow a random forest, RF_1, using only data in D_1.

2. Grow a second random forest, RF_2, using only data in D_2.

3. For $j \in D_2$, calculate OOB residuals $e_j = y_j - \widehat{y}_{OOB}(\boldsymbol{x}_j)$, using OOB predictions from RF_2.

4. For $j \in D_2$, calculate weights $\nu_j = B\left(\frac{e_j}{6m}\right)$, where $m = \text{Median}\left\{|e_j|\right\}_{j \in D_2}$, and B is as defined in (4.7).

5. For $\alpha \in \mathcal{I}$, use RF_1 to calculate RF-LOWESS predictions for cases $j \in D_2$. Denote the prediction for case j using tuning parameter candidate α as $\widehat{y}^{(\alpha)}(\boldsymbol{x}_j)$.

6. For $\alpha \in \mathcal{I}$, calculate a weighted mean square error

$$\text{WMSE}_\alpha = \sum_{j \in D_2} \nu_j \left(y_j - \widehat{y}^{(\alpha)}(\boldsymbol{x}_j)\right)^2.$$

A different loss function could be substituted in place of mean square error if desired.

This procedure is repeated for each fold in the cross-validation, and the resulting values of WMSE_α are averaged. Then the choice of α is made by minimizing the average WMSE_α value over all folds and repetitions. A random forest is grown on the full set of training data, and the RF-LOWESS algorithm is used with this value of α to make predictions for new cases.

[LM] state that cross-validation could be used to set the value of the tuning parameter, δ, used in their Huber forest, although they do not discuss the potential impact of training data contamination on the tuning process. The procedure we suggest can reasonably be applied in the context of the Huber forest. This would, however, require iteratively calculating weights for each OOB training case individually, making the procedure more computationally expensive than for RF-LOWESS.

The values of α in RF-LOWESS and δ in the Huber forest approach do not affect the tree-growing process. Consqeuently, all proposed values for these parameters can be evaluated without having to regrow a forest. Other random forest tuning parameters include the size of a node below which no further splitting is allowed, and the number of explanatory variables randomly selected for consideration for each split. These are denoted *nodesize* and *mtry*, respectively, in the `randomForest` package (Liaw and Wiener, 2002). When data contamination is suspected, these could also be set using a procedure similar to the one described in Algorithm 2, but different random forests would need to be grown for each parameter value under consideration.

4.4 Simulations & Real Data Results

4.4.1 Simulation Study

In order to investigate the effectiveness of RF-LOWESS, we carry out two simulations used by [RL], and two others used by [LM] and compare the performance of RF-LOWESS

to the prior approaches described in those papers. Our study also provides a comparison of the Huber forest of [LM] to the median-based aggregation approaches suggested by [RL]. Such a comparison was not previously available in the literature, as [LM] only compare their Huber forest to an ordinary random forest and the quantile regression forest introduced by [M].

In each simulation, contamination was introduced by generating a proportion p of training response observations from a distribution with larger variance than the distribution used to generate the rest of the response values. We consider values of p equal to 0, 0.05, 0.10, 0.15, 0.20, and 0.25. We set $p = 0$ when generating test data, because our focus is on predicting uncontaminated responses for new target cases.

Simulation 1

In the first simulation, data are generated through a tree-like mechanism used by [RL]. A six-dimensional vector of independent, normally distributed explanatory variables is used, i.e. $\boldsymbol{X} \sim \mathcal{N}_6(\mathbf{0}, I_6)$, where I_6 is a six-by-six identity matrix. The response variable is defined as

$$\begin{aligned} Y_i = m \times \Bigg(& \mathbb{1}((X_{1i} \leq 0), (X_{2i} \leq 0)) \\ & + 2\mathbb{1}((X_{1i} \leq 0), (X_{2i} > 0), (X_{4i} \leq 0)) \\ & + 3\mathbb{1}((X_{1i} \leq 0), (X_{2i} > 0), (X_{4i} > 0), (X_{6i} \leq 0)) \\ & + 4\mathbb{1}((X_{1i} \leq 0), (X_{2i} > 0), (X_{4i} > 0), (X_{6i} > 0)) \\ & + 5\mathbb{1}((X_{1i} \leq 0), (X_{3i} \leq 0)) \\ & + 6\mathbb{1}((X_{1i} \leq 0), (X_{3i} \leq 0), (X_{5i} \leq 0)) \\ & + 7\mathbb{1}((X_{1i} \leq 0), (X_{3i} \leq 0), (X_{5i} > 0)) \Bigg) \\ & + \epsilon_i \mathbb{1}(r_i \geq p) \\ & + \gamma_i \mathbb{1}(r_i < p)), \end{aligned}$$

where, ϵ_i are normally distributed with mean 0 and standard deviation 1, γ_i are normally distributed with mean 0 and standard deviation 5, and r_i follow uniform distributions on the unit interval.

The constant m describes the signal-to-noise ratio in the data. As m increases, the signal-to-noise ratio becomes stronger. [RL] use values of $m = 0.20$ and $m = 0.80$ to represent moderate and high signal-to- noise ratios. We use each of these values and also $m = 0.40$ and $m = 0.60$.

Simulation 2

In the second simulation, also from [RL], $\boldsymbol{X} \sim \mathcal{N}_6(\mathbf{0}, I_6)$, and the expected response is a nonlinear function of the predictor variables defined by

$$Y_i = m \times \Big(X_{1i} + 0.707X_{2i}^2 + \mathbb{1}(X_{3i} > 0) + 0.873\log(|X_{1i}|)X_{3i} + 0.894X_{2i}X_{4i} + 2\mathbb{1}(X_{5i} > 0) + 0.464\exp(X_{6i}) \Big) + \epsilon_i \mathbb{1}(r_i > p) + \gamma_i \mathbb{1}(r_i < p),$$

where ϵ_i, γ_i, r_i, and m are defined as in Simulation 1. Like [RL], we used $m = 0.15$ and $m = 0.60$ to signify moderate and high signal-to-noise ratios, and also considered $m = 0.30$ and $m = 0.45$.

Simulation 3

In this simulation, taken from [LM], $\boldsymbol{X} \sim \mathcal{N}_{10}(\mathbf{0}, I_{10})$, and

$$Y_i = \sum_{j=1}^{10} X_{ji}^2 + \epsilon_i \mathbb{1}(r_i \geq p) + 15\gamma_i \mathbb{1}(r_i < p),$$

where $\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, 1)$, and γ_i are independent observations from t-distributions with two degrees of freedom.

Simulation 4

This simulation, also from [LM], differs from Simulation 3 by using correlated predictors. Here, $\boldsymbol{X} \sim \mathcal{N}_{10}(\mathbf{0}, \Sigma)$, where Σ is a Toeplitz matrix with $\rho = 0.7$,

$$Y_i = \sum_{j=1}^{10} X_{ji}^2 + \epsilon_i \mathbb{1}(r_i \geq p) + 15\gamma_i \mathbb{1}(r_i < p),$$

where ϵ and γ are defined as in Simulation 3.

In simulations 1 and 2, training and test sets consisted of 500 and 1,000 observations, respectively, which is consistent with the simulations of [RL]. In simulations 3 and 4, training and test sets each consisted of 1,000 observations, consistent with [LM]. Each simulation was repeated 500 times for each value of p (and for each value of m in simulations 1 and 2).

We consider the following random-forest-based prediction approaches:

1. Ordinary random forest (RF)
2. Quantile random forest (QRF)
3. Use of median when aggregating predictions across trees (Mean-Med.)
4. Use of median when aggregating predictions within terminal nodes and across trees (Med.-Med.)
5. Li & Martin's Huber forest (Huber)
6. RF-LOWESS

Random forest tuning parameters were set in accordance with the settings used by [RL] in simulations 1 and 2, and by [LM] in simulations 3 and 4. The *nodesize* parameter was set to 15 in simulations 1 and 2, and 10 in simulations 3 and 4. In each simulation, the *mtry* parameter was set to its default value of the floor of one-third times the number of explanatory variables. Forests of 500 trees were grown.

Within each simulation, the tuning procedure given in Algorithm 2 was used to determine the optimal value of α to be used for RF-LOWESS. Values of α ranging from 1 to 30 by increments of 0.25 were considered, along with $\alpha = 100$ and $\alpha = 1,000$. Training data were partitioned into five equal sized sets. One at a time, each set was withheld and used in the place of D_2 in Algorithm 2, with the other four sets comprising D_1. Once the optimal value of α was determined from the training data, it was used to make RF-LOWESS predictions on the test data. The random forests RF_1 and RF_2, used for parameter tuning, consisted of 100 trees. The test data played no part in the choice of α. The value of $\delta = 0.005$, suggested by [LM], was used for the Huber forest. We did not attempt to tune this procedure due to the computational complexity associated with recalculating weights for each individual case within each step of the cross-validation. The error tolerance was set to $\epsilon_0 = 10^{-4}$ for RF-LOWESS, and $\epsilon_0 = 10^{-6}$ for Huber forest predictions.

We evaluate the performance of each technique using mean square prediction error (MSPE), defined as

$$\text{MSPE} = \frac{1}{n_t}\sum_{i=1}^{n_t}(y_i - \widehat{y}_i)^2,$$

and mean absolute prediction error (MAPE) defined as

$$\text{MAPE} = \frac{1}{n_t}\sum_{i=1}^{n_t}|y_i - \widehat{y}_i|,$$

where n_t represents the number of test cases. In most situations, the techniques that performed best with respect to MSPE also performed best with respect to MAPE. We present our results using MSPE and include MAPE results in situations where there were notable differences.

Figure 4.3 displays the results for simulation 1. We see that all five of the robust approaches achieve considerably lower MSPE than ordinary RF predictions when training data contamination is present. The RF-LOWESS algorithm outperforms all other approaches for low to moderate signal-to-noise ratios indicated by $m = 0.20$ and $m = 0.40$, with the magni-

tude of the differences increasing as the proportion of contaminated observations increases. QRF achieves the next strongest performance in these situations. As the signal-to-noise ratio increases, other approaches gain on and surpass RF-LOWESS. For $m = 0.60$, the QRF, median aggregation, Huber forest and RF-LOWESS techniques all yield similar results, with QRF slightly edging the others when contamination is most prevalent. For $m = 0.80$, median aggregation, Huber, and QRF achieve lower MSPE than RF-LOWESS. Although it might be surprising to see MSPE increase as the signal-to-noise ratio increases, this is explained by larger values of m resulting in more variability in the marginal distribution of y-values in the training data.

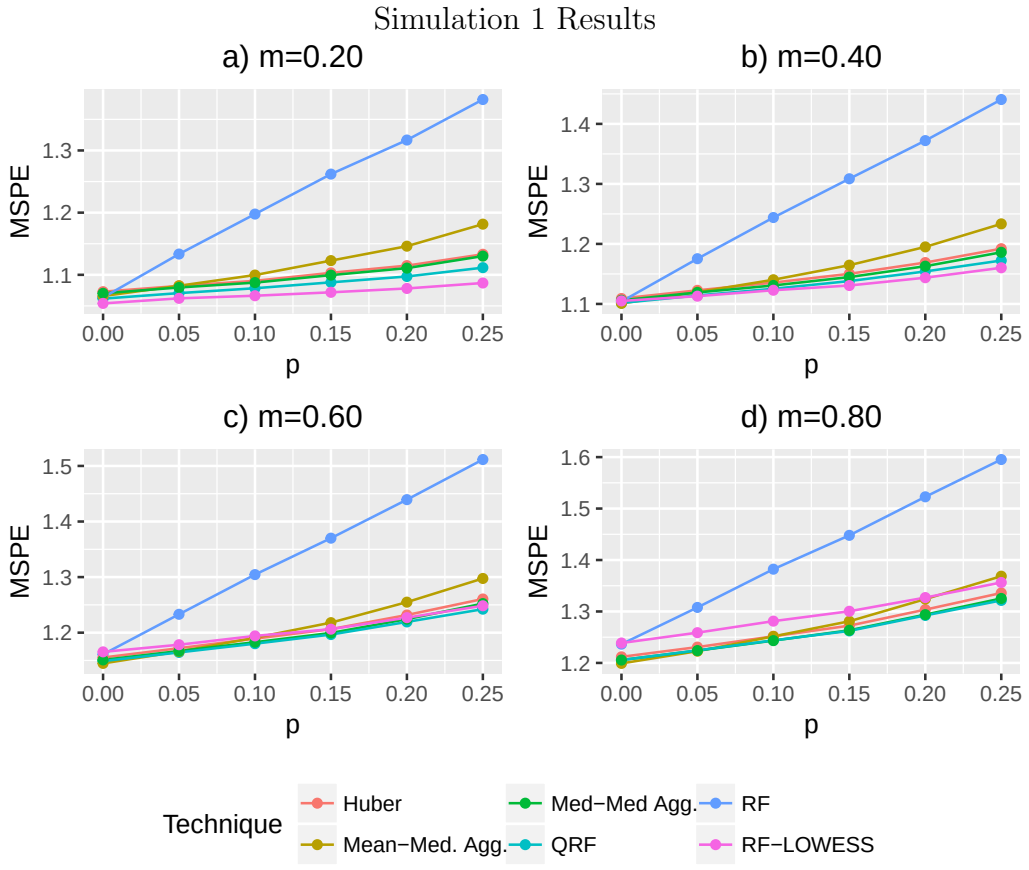

Figure 4.3: RF-LOWESS outperforms other approaches for moderate signal-to-noise ratio. Other approaches perform better when the signal-to-noise ratio is high.

Figure 4.4 displays results for simulation 2. These results are similar to those for simulation 1, with RF-LOWESS again achieving the best performance for low and moderate signal-to-noise ratios. Li & Martin's Huber forest performs best when the signal-to-noise ratio is high. While QRF is competitive for $m = 0.15$ and $m = 0.30$, it is surpassed by other techniques for the larger two values of m. In most situations, there is little difference between the techniques when $p = 0$. For $m = 0.60$, the original random forest outperforms the robust approaches when no contamination is present. In this situation, the RF-LOWESS tuning approach is usually able to correctly revert to RF predictions by setting $\alpha = 1,000$, causing RF-LOWESS to outperform the other robust approaches. In both simulations 1 and 2, RF-LOWESS outperforms the other robust techniques in situations where the data are noisiest.

Figure 4.5 shows the results for simulations 3 and 4. We see in Figures 4.5 (a) and (b) that the robust approaches again outperform RF when contamination is present. Figures 4.5 (c) and (d) omit the RF results, focusing on results for the five robust approaches. Like [LM], we find that the Huber forest achieves lower MSPE than RF and QRF in this simulation. However, median aggregation and RF-LOWESS outperform all of these techniques, with respect to MSPE. RF-LOWESS achieves the lowest MSPE, except when $p = 0.25$. When explanatory variables are correlated, as in Simulation 4, the median aggregation approaches perform best, followed by the Huber forest. While RF-LOWESS performs well for low values of p, it becomes suboptimal when p exceeds 0.10. QRF performs poorly, with respect to MSPE, in both simulations 3 and 4.

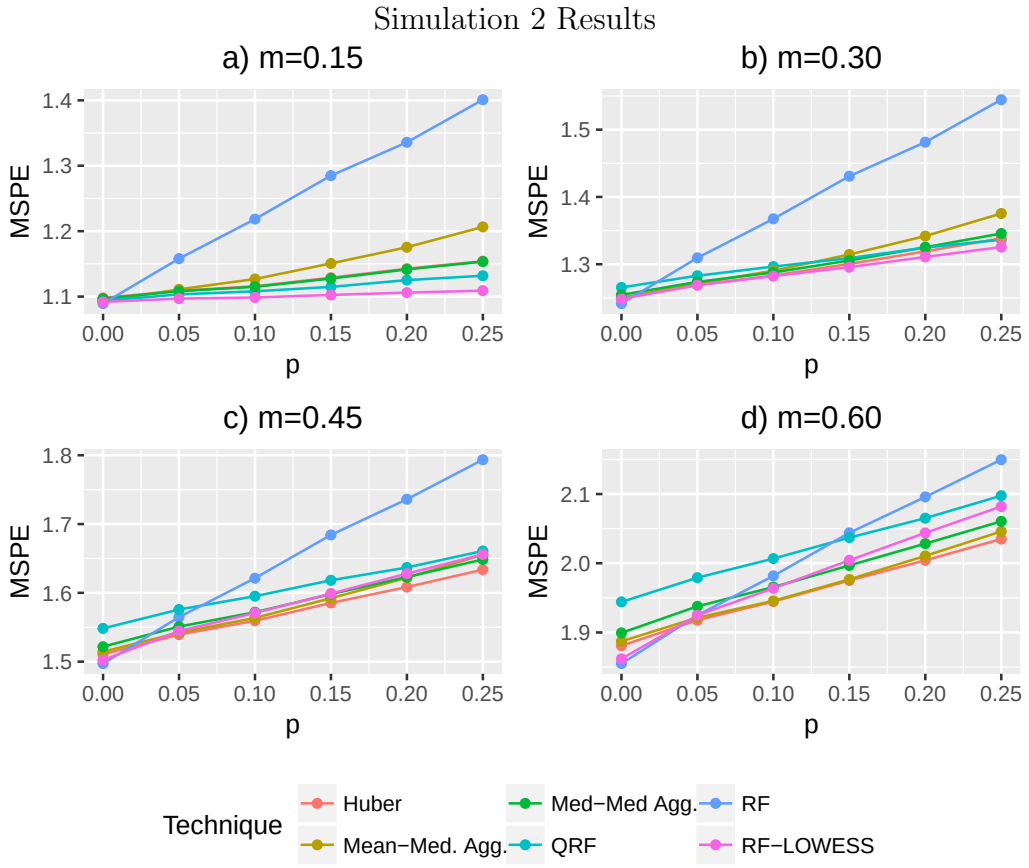

Figure 4.4: RF-LOWESS outperforms other approaches for moderate signal-to-noise ratio. The Huber approach achieves the strongest performance when the signal-to-noise ratio is high.

In simulations 1 and 2, the results obtained using the MAPE evaluation criteria largely resemble those obtained using MSPE and are thus omitted. In simulations 3 and 4, however, results differ considerably when predictions are evaluated using MAPE. These results are shown in Figure 4.6.

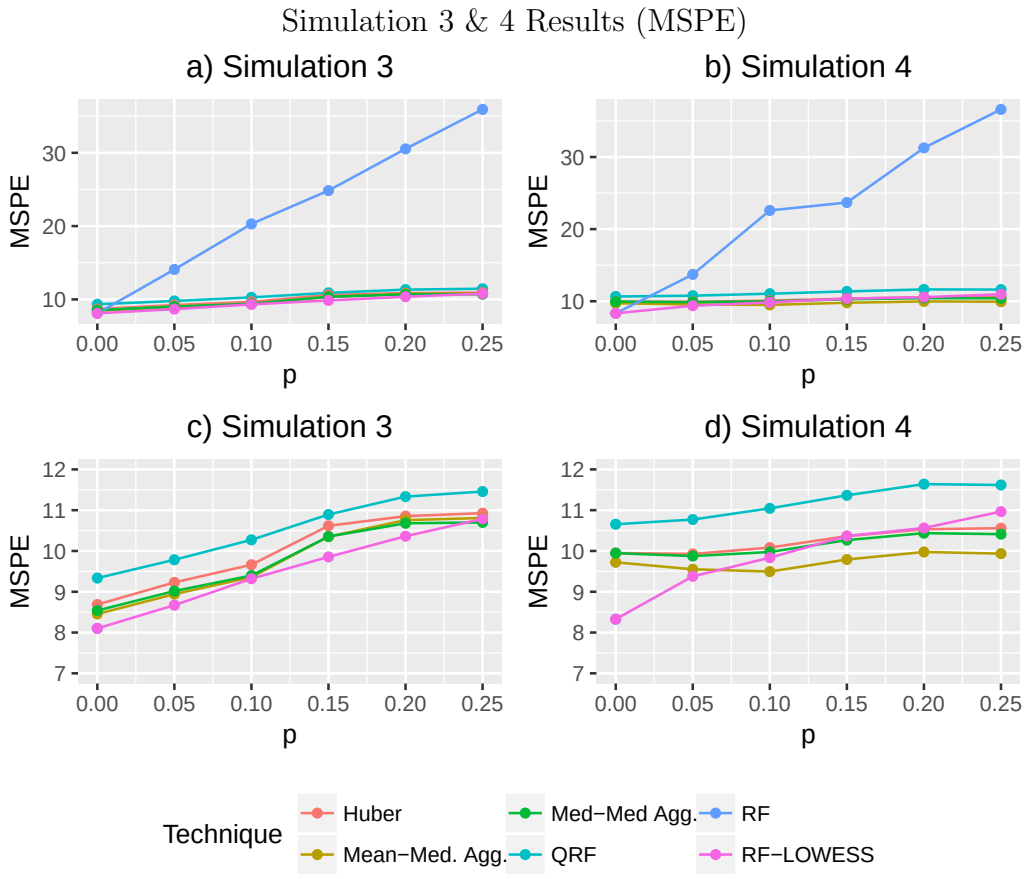

Figure 4.5: In simulation 3, RF-LOWESS achieves the lowest MSPE for each setting except $p = 0.25$. In simulation 4, the median aggregation approaches perform best for large p.

The median aggregation approaches perform best when predictions are evaluated using MAPE. The fact that RF-LOWESS achieves stronger performance using MSPE than MAPE is due to the fact that it is less likely to miss on a prediction by a very large amount. In situations where a considerable amount of the original random forest weight comes from outlying observations, other robust procedures are prone to not only failing to down-weight these outliers, but actually increase their prediction weight, as we saw with the Huber forest in Section 4.3.2. As a result, the other robust approaches miss severely on these predictions, heavily affecting MSPE. Because MAPE does not reward such large improvements on a small

number of test cases as heavily, other techniques gain on RF-LOWESS, when MAPE is used. In practice, whether it is more costly to badly mispredict a few cases, or to make smaller prediction errors on a larger number of cases, will be context dependent.

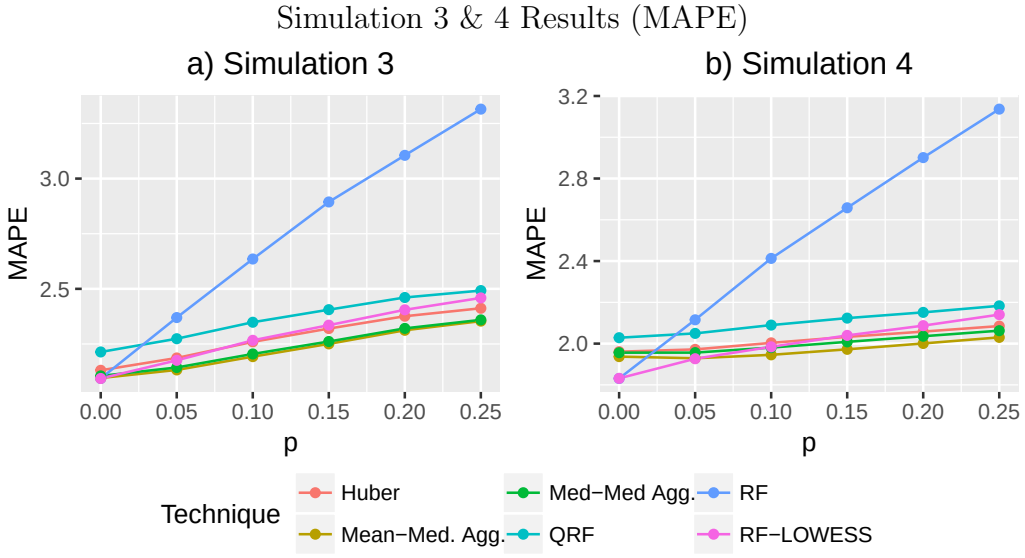

Figure 4.6: The median aggregation approaches usually achieve the best MAPE in both simulations.

We also tested the impact of LAD splitting on RF-LOWESS, finding that it typically results in small improvements (usually a decrease in MSPE or MAPE of less than 2%) when contamination is prevalent. These results are consistent with those observed by [RL] for other robust approaches. The techniques that perform best using LS splitting continue to do so when LAD splitting is used. Additional details on the results of LAD splitting are given in the appendix.

4.4.2 Impact of Weighted Cross-Validation

Algorithm 2 provides a method for down-weighting training cases believed to be outliers, when setting tuning parameters via cross-validation. Figure 4.12 shows the MSPE values

obtained for RF-LOWESS predictions on test data, after tuning the method on training data, using ordinary cross-validation, and the weighted cross-validation approach given in Algorithm 2. Also shown are the results that would have been obtained on test data, in the idealized scenario where we could tune the algorithm using only noncontaminated training cases and ordinary cross-validation. This is, of course, impossible to do in practice. Figure 4.12 and the other figures appearing in this section, are based on results from simulation 2. Results for the other simulations are similar and are provided in the appendix.

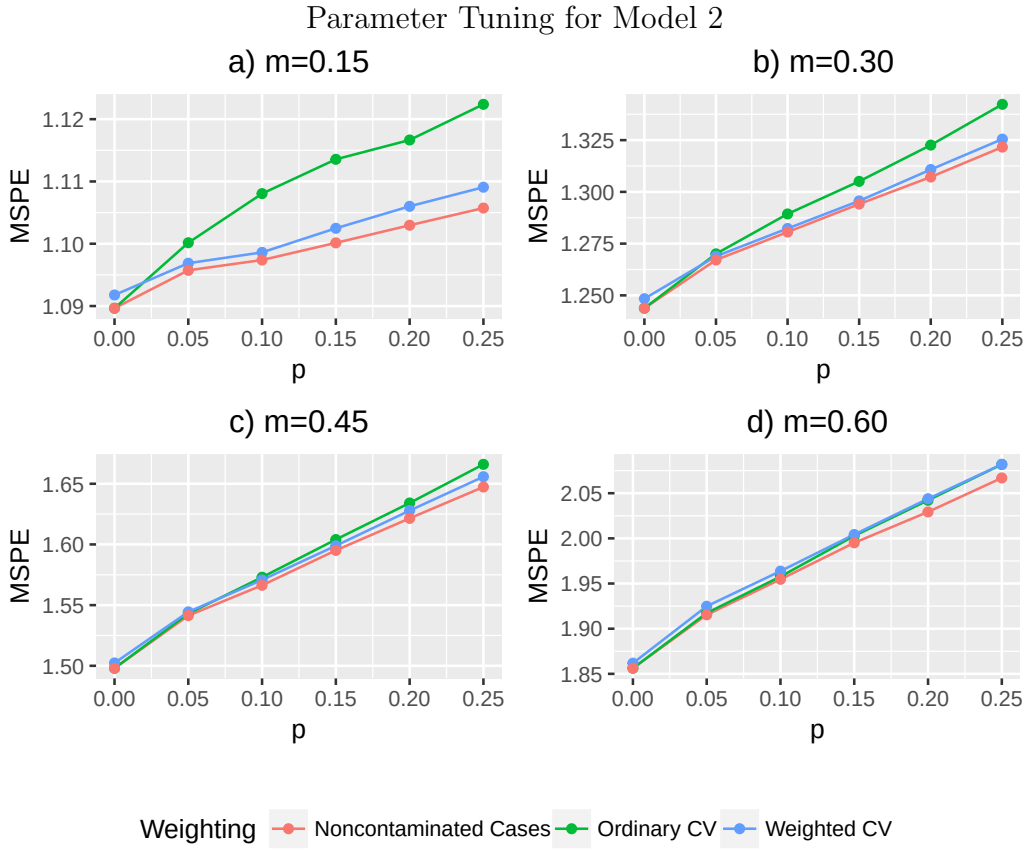

Figure 4.7: The weighted cross-validation tuning procedure yields results closer to the ideal results that would be obtained using only noncontaminated cases than ordinary cross-validation.

We see that the weighted cross-validation approach comes much closer to matching the ideal performance than ordinary CV. The impact of down-weighting outlying training cases is especially large for noisier datasets ($m = 0.15$, $m = 0.30$). When the signal-to-noise ratio is high, weighting training cases makes little difference in the choice of α.

Figures 4.8 (a) & (b) illustrate the RF-LOWESS algorithm's sensitivity to the choice of α for $m = 0.15$ and $m = 0.60$, when $p = 0.25$. Specifically, these figures display the MSPE that would have been obtained on test data, averaging across the 500 repetitions, if a given value of α had been used for all 500 repetitions. When $m = 0.15$, the optimal choice is $\alpha = 4.25$. Performance deteriorates considerably for $\alpha > 10$. When $m = 0.6$, the optimal value of α is 11.75. In this situation, performance suffers drastically if α is chosen to be too small, but not as dramatically if a large α is selected.

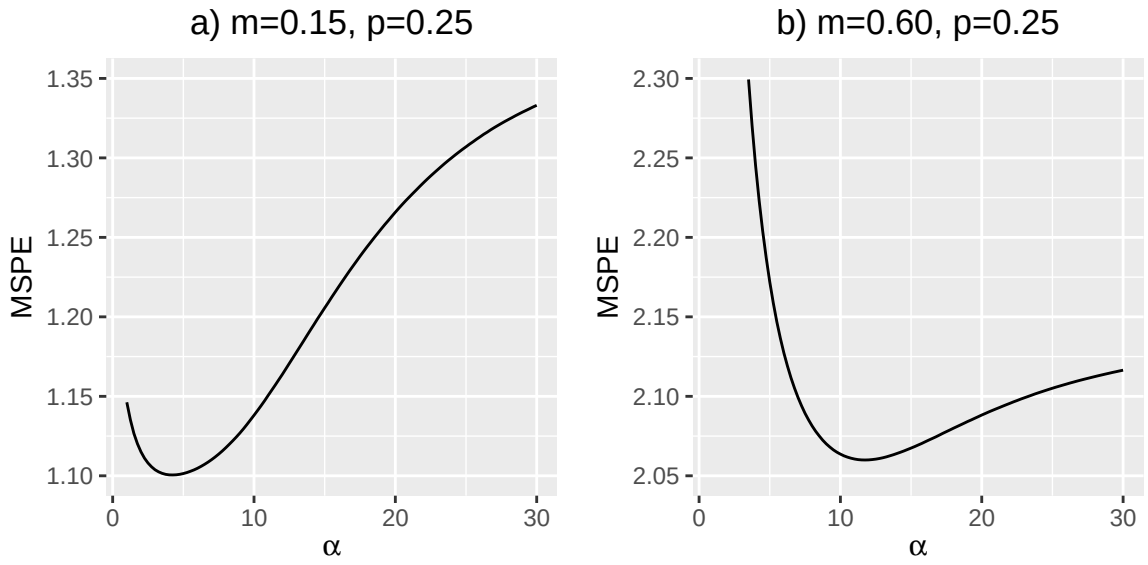

Figure 4.8: For $m = 0.15$, performance suffers drastically for $\alpha > 5$. When the signal-to-noise ratio is high (m=0.6), the procedure is not as sensitive to the choice of α, as long as $\alpha > 7$.

Figures 4.9 (a) & (b) display the distribution of the values of α selected using the training data in each of the 500 repetitions of our simulation. Excluded from Figure 4.9 (b) are 24 instances where ordinary cross-validation resulted in a choice of $\alpha > 30$, and 2 such instances for weighted cross-validation. We see that ordinary cross-validation is prone to choosing values of α that are larger than optimal. Such choices are driven by predictions on training data outliers during the tuning process. The weighted cross-validation approach yields more precision, typically choosing values of α in a narrower range, closer to the optimal values seen in Figures 4.8 (a) & (b). Undesirably large values of α are rarely chosen via weighted cross-validation. Figures 4.8 and 4.9 explain why weighted cross-validation outperforms ordinary cross-validation substantially for $m = 0.15$, but not for $m = 0.60$. In the former case, small α are highly desired and the ordinary cross-validation procedure's tendency to select larger α results in inferior performance. For $m = 0.60$, ordinary cross-validation is still prone to selecting larger than optimal values of α, but doing so has little impact on predictions, due to the algorithm's lack of sensitivity in this situation.

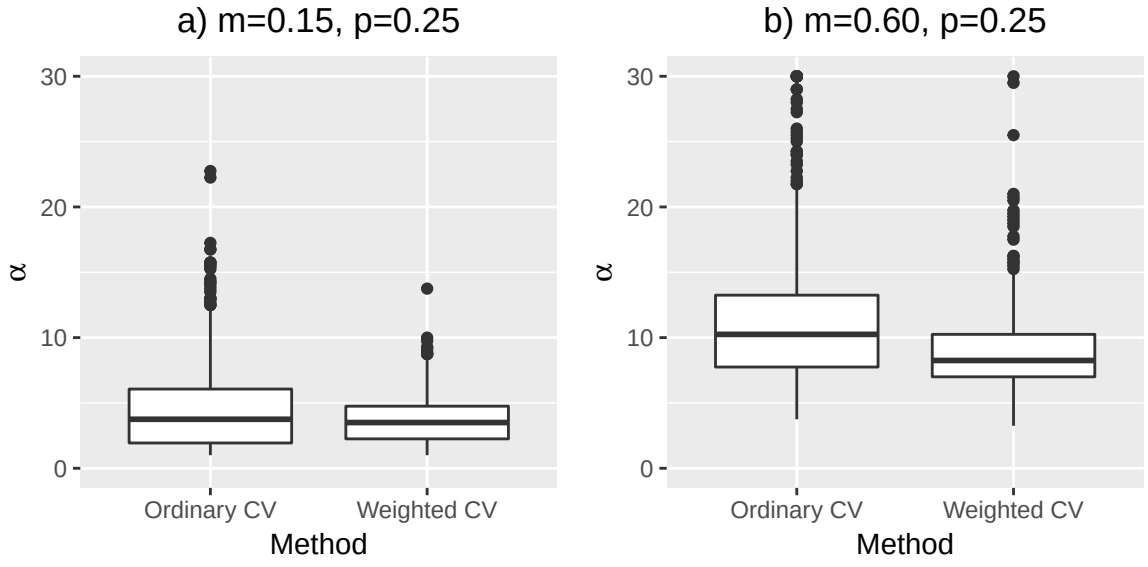

Figure 4.9: For both $m = 0.15$, and $m = 0.60$, there is more variability in the values of α chosen using ordinary cross-validation than the values chosen by weighted cross-validation.

Table 4.3 gives the optimal choice of α for each setting of m and p, which result from calculations on test data, averaging over the 500 repetitions. For a given m, we see that the optimal value of α decreases as the proportion of contaminated training cases grows. When contamination is prevelant, small α are chosen, resulting in an aggressive weighting adjustment. When there is little or no contamination, large values of α are chosen, yielding predictions similar those obtained using ordinary random forest weights. When there is no contamination, we see that $\alpha = 1,000$ is chosen in all except the noisiest setting, effectively reverting to ordinary random forest predictions in these scenarios. For a given p, small α are desirable when m is small, indicating noisy training data, while larger α are desirable when the signal-to-noise ratio is stronger.

Table 4.3: Optimal choices of α. Small values, corresponding to aggressively down-weighting outliers, are desirable for noisy datasets with many outliers. Large α are preferable when data contain strong signal and fewer outliers.

	$p = 0$	$p = 0.05$	$p = 0.10$	$p = 0.15$	$p = 0.20$	$p = 0.25$
$m = 0.15$	5.00	5.25	5.00	5.00	4.50	4.25
$m = 0.30$	1000	10.00	9.00	8.00	7.50	6.75
$m = 0.45$	1000	14.75	12.25	10.75	9.75	9.00
$m = 0.60$	1000	20.25	16.00	13.75	13	11.75

4.4.3 Data Applications

We now test each approach using three real datasets, previously considered in the literature. These include the airfoil and concrete compressive strength (Yeh, 1998) datasets from the UCI Machine Learning Repository (Lichman, 2013) and the low birthweight dataset referenced in Hosmer Jr. et al. (2013). The airfoil dataset is based on aerodynamic and acoustic tests on airfoil blade sections and involves predicting sound pressure of airfoils. The

dataset includes 1,503 observations and 5 explanatory variables. The concrete dataset can be used to predict the strength of concrete specimens using information on eight different ingredients. The dataset contains 1,030 observations. The low birthweight dataset contains information on 189 mothers and their babies, collected at Baystate Medical Center in Springfield, Massachusetts in 1986. The dataset contains 8 predictor variables. We use the actual birthweight as the response variable for our regression task. The dataset also includes an indicator variable for whether the birthweight was less that 2.5 kg. We did not use this variable in our analysis. The birthweight dataset exhibits a high degree of variability in weights, making it particularly interesting in our study.

For each dataset, each robustness approach was tested using 30 repetitions of k-fold cross validation. Values of $k = 9$, $k = 3$, and $k = 5$ were used for the airfoil, birthweight, and concrete datasets respectively. These values were selected in order to evenly partition the data into folds with test sets large enough to adequately assess performance. Forests of 500 trees were grown using default settings for the *mtry* parameter and by setting the *nodesize* parameter to 10, as was done by [RL]. In the airfoil and concrete datasets, the RF-LOWESS algorithm was tuned using a single four-fold cross-validation, with Algorithm 2 implemented within each fold. For the birthweight dataset, three repetitions of three-fold cross-validation were performed, again implementing Algorithm 2 in each fold. Forests consisting of 100 trees were used for tuning. Test data remained untouched throughout the tuning process and average MSPE and MAPE were calculated across test sets after predictions were made.

We tested each technique on the actual datasets and also datasets created by adding contamination to the training data. In each fold, contamination was introduced by adding the value resulting from a draw from a normal distribution with mean 0 and standard deviation equal to $5\sigma_Y$, to 20% of all training cases (where σ_Y is the standard deviation of the empirical marginal distribution for Y). This method for adding contamination is consistent with the approach used by [RL].

Tables 4.4 and 4.5 give the ratio of MSPE and MAPE for each technique relative to those achieved by the ordinary RF approach. When there is no training data contamination, the QRF, Huber, and median aggregation approaches are able to achieve moderate improvements over the original RF predictions, while RF-LOWESS does not. The Huber approach shows particular promise when predictions are evaluated using MAPE. When contamination is present, the RF-LOWESS approach results in considerable improvement over RF, and achieves the best performance on the birthweight and concrete datasets.

Table 4.4: Ratio of MSPE to that of ordinary RF for real datasets

Dataset	Contamination	QRF	Huber	Mean-Med.	Med-Med	RFL
Airfoil	No	0.8535	**0.8345**	0.9881	1.0109	1.0000
Birthweight	No	1.0036	1.0058	1.0007	0.9993	**0.9974**
Concrete	No	0.9352	0.9140	**0.9114**	0.9317	1.0000
Airfoil	Yes	0.7326	0.7417	0.8868	0.7809	**0.7287**
Birthweight	Yes	0.7572	0.7732	0.8767	0.7660	**0.7229**
Concrete	Yes	0.4821	0.6668	0.9368	0.6976	**0.2366**

Table 4.5: Ratio of MAPE to that of ordinary RF for real datasets

Dataset	Contamination	QRF	Huber	Mean-Med.	Med-Med	RFL
Airfoil	No	0.9063	**0.8979**	0.9845	0.9890	1.0000
Birthweight	No	0.9910	0.9905	0.9958	**0.9879**	1.0039
Concrete	No	0.9041	**0.8973**	0.9150	0.9143	1.0000
Airfoil	Yes	0.8665	**0.8660**	0.9562	0.9194	0.9148
Birthweight	Yes	0.8911	0.8951	0.9471	0.8909	**0.8810**
Concrete	Yes	0.7042	0.7642	0.9210	0.7876	**0.6374**

4.5 Conclusions

We have introduced a new residual-based approach for random forest regression. This method is motivated by the popular LOWESS approach for polynomial regression and extends this method's intuition to random forests. We have shown that down-weighting the contributions of training cases with large OOB residuals increases robustness and improves predictive performance when contamination is present in the training data responses. Through simulations and analysis of real data, we have shown that RF-LOWESS outperforms the best existing techniques when working with noisy datasets where the proportion of contaminated cases is high. Even when it is not optimal, RF-LOWESS improves on ordinary random forest predictions, and it is consistently competitive with the best robust approaches.

RF-LOWESS is intended for situations in which users have reason to believe that contamination is present in training data but not in the data on which they seek to make predictions. When no contamination is present, the RF-LOWESS algorithm, properly tuned, will revert back to ordinary random forest predictions by using very large α. Therefore, the ordinary random forest approach can be seen as a special case of RF-LOWESS, making RF-LOWESS flexible enough to improve predictions on noisy datasets, without hurting performance on uncontaminated data. This is not always true of other robustness techniques, as is evidenced by the results of simulation 2, when $m = 0.60$ and $p = 0$. However, in the real data examples, the Huber, QRF, and median aggregation approaches were sometimes able to improve on ordinary random forest predictions when no contamination was present, while RF-LOWESS was not. Still, RF-LOWESS typically performed strongest when contamination was introduced in these datasets.

Like [RL] we did not find one robust approach to uniformly outperform the others. In addition to RF-LOWESS, the Huber forest algorithm of [LM] shows promise, especially in

situations where the signal-to-noise ratio is strong. The median aggregation approaches suggested by [RL] and the quantile regression forest, introduced by [M] also show promise in various situations.

We have also introduced a weighted cross-validation approach for setting tuning parameters when training data contamination is present. This approach results in tuning parameter selections closer to those that would be attained in the idealized setting where we would actually know which cases resulted from contamination. Ordinary cross-validation is unreliable in this situation, because of the impact of predictions made on training data outliers. A weighted cross-validation approach similar to ours would likely be useful in parameter tuning for other robust approaches, such as the Huber forest of [LM] as well.

Since the optimal robust approach varies, depending on factors such as signal-to-noise ratio and the number of outliers, it would be helpful to be able to predict which robust approach will work best on a given dataset. Our weighted cross-validation approach could be used in this context. Future research might further explore how to choose an optimal robust approach, or an appropriate weighted average of such approaches, when training response contamination is a concern.

Bibliography

Bhat, H. S., Kumar, N., and Vaz, G. J. (2015). Towards scalable quantile regression trees. In *Big Data (Big Data), 2015 IEEE International Conference on*, pages 53–60. IEEE.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Brence, M. J. R. and Brown, D. E. (2006). Improving the robust random forest regression algorithm. *Systems and Information Engineering Technical Papers, Department of Systems and Information Engineering, University of Virginia*.

Charbonnier, P., Blanc-Féraud, L., Aubert, G., and Barlaud, M. (1997). Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on Image Processing*, 6(2):298–311.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836.

Folleco, A., Khoshgoftaar, T. M., Van Hulse, J., and Bullard, L. (2008). Software quality modeling: The impact of class noise on the random forest classifier. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3853–3859. IEEE.

Galimberti, G., Pillati, M., and Soffritti, G. (2007). Robust regression trees based on m-estimators. *Statistica*, 67(2):173–190.

Hamza, M. and Larocque, D. (2005). An empirical comparison of ensemble methods based on classification trees. *Journal of Statistical Computation and Simulation*, 75(8):629–643.

Hosmer Jr., D. W., Lemeshow, S., and Sturdivant, R. X. (2013). *Applied logistic regression*, volume 398. John Wiley & Sons.

Huber, P. J. (2011). Robust statistics. In *International Encyclopedia of Statistical Science*, pages 1248–1251. Springer.

Li, A. H. and Martin, A. (2017). Forest-type regression with general losses and robust forest. In *International Conference on Machine Learning*, pages 2091–2100.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

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

Meinshausen, N. (2006). Quantile regression forests. *The Journal of Machine Learning Research*, 7:983–999.

Meinshausen, N. (2016). *quantregForest: Quantile Regression Forests*. R package version 1.3-5.

Mosteller, F. and Tukey, J. W. (1977). Data analysis and regression: a second course in statistics. *Addison-Wesley Series in Behavioral Science: Quantitative Methods*.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rousseeuw, P. J. and Leroy, A. M. (2005). *Robust regression and outlier detection*, volume 589. John wiley & sons.

Roy, M.-H. and Larocque, D. (2012). Robustness of random forests for regression. *Journal of Nonparametric Statistics*, 24(4):993–1006.

Torgo, L. F. R. A. (1999). Inductive learning of tree-based regression models.

Yeh, I.-C. (1998). Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808.

4.6 APPENDIX: Additional Details and Results

4.6.1 Li & Martin's Huber Forest Algorithm

Algorithm 3 Huber Forest Algorithm (Li and Martin, 2017)

For test points $\{\boldsymbol{x}_k\}_{k=1}^m$, initial guess $\left\{\widehat{y}^{(0)}(\boldsymbol{x}_k)\right\}$, local weights $w_{i,k}$, training responses $\{y_i\}_{i=1}^n$, and error tolerance ϵ_0,

1. While $\epsilon > \epsilon_0$:

   i. Update weights
   $$w_{i,k}^{(l)} = \frac{w_{i,k}}{\sqrt{1 + \left(\frac{\widehat{Y}^{(l-1)}(\boldsymbol{x}_k) - y_i}{\delta}\right)^2}}.$$

   ii. Update the estimator
   $$\widehat{y}^{(pH)}(\boldsymbol{x}_k) = \frac{\sum_{i=1}^n w_{i,k}^{(l)} y_i}{\sum_{i=1}^n w_{i,k}^{(l)}}.$$

   iii. Calculate error
   $$\epsilon = \frac{1}{m}\sum_{k=1}^m \left(\widehat{y}^{(l)}(\boldsymbol{x}_k) - \widehat{y}^{(l-1)}(\boldsymbol{x}_k)\right)^2.$$

   iv. set $l = l + 1$.

2. Output the Huber forest estimator $\widehat{y}_H(\boldsymbol{x}_k) = \widehat{y}^{(l)}(\boldsymbol{x}_k)$.

4.6.2 Additional Simulation Results

Impact of LAD Splitting

Table 4.6: Ratio of MSPE using LAD splitting to MSPE using LS splitting for Simulation 2 ($m = 0.15$).

	$p = 0$	$p = 0.05$	$p = 0.1$	$p = 0.15$	$p = 0.2$	$p = 0.25$
RF	0.997	0.982	0.970	0.956	0.952	0.971
QRF	1.002	1.001	1.002	1.002	1.005	1.001
Li-Martin(Huber)	1.002	1.004	1.005	1.002	1.005	1.003
Mean-Med.	0.999	0.996	0.992	0.983	0.982	0.992
Med.-Med.	1.004	1.004	1.005	1.001	1.003	1.003
RF-LOWESS	1.001	1.002	1.002	1.000	1.000	1.001

Table 4.7: Ratio of MAPE using LAD splitting to MAPE using LS splitting for Simulation 2 ($m = 0.15$)

	$p = 0$	$p = 0.05$	$p = 0.1$	$p = 0.15$	$p = 0.2$	$p = 0.25$
RF	0.999	0.992	0.985	0.978	0.976	0.970
QRF	1.001	1.001	1.001	1.001	1.002	1.002
Li-Martin(Huber)	1.001	1.002	1.002	1.002	1.003	1.004
Mean-Med.	0.999	0.998	0.996	0.994	0.993	0.990
Med.-Med.	1.002	1.002	1.002	1.001	1.002	1.003
RF-LOWESS	1.001	1.001	1.001	1.000	1.001	1.000

Table 4.8: Ratio of MSPE using LAD splitting to MSPE using LS splitting for Simulation 2 ($m = 0.60$)

	$p = 0$	$p = 0.05$	$p = 0.1$	$p = 0.15$	$p = 0.2$	$p = 0.25$
RF	1.016	1.001	0.990	0.979	0.973	0.973
QRF	1.017	1.008	1.000	0.990	0.987	0.987
Li-Martin(Huber)	1.018	1.009	0.999	0.991	0.986	0.986
Mean-Med.	1.023	1.015	1.006	0.998	0.992	0.992
Med.-Med.	1.022	1.013	1.003	0.994	0.989	0.989
RF-LOWESS	1.014	1.002	0.994	0.988	0.985	0.985

Table 4.9: Ratio of MAPE using LAD splitting to MAPE using LS splitting for Simulation 2 ($m = 0.60$)

	$p = 0$	$p = 0.05$	$p = 0.1$	$p = 0.15$	$p = 0.2$	$p = 0.25$
RF	1.008	1.000	0.995	0.990	0.986	0.983
QRF	1.009	1.004	1.000	0.996	0.994	0.992
Li-Martin(Huber)	1.009	1.004	1.000	0.996	0.994	0.992
Mean-Med.	1.012	1.008	1.003	0.999	0.996	0.994
Med.-Med.	1.012	1.007	1.002	0.998	0.996	0.993
LOWESS-RF	1.007	1.001	0.997	0.994	0.992	0.991

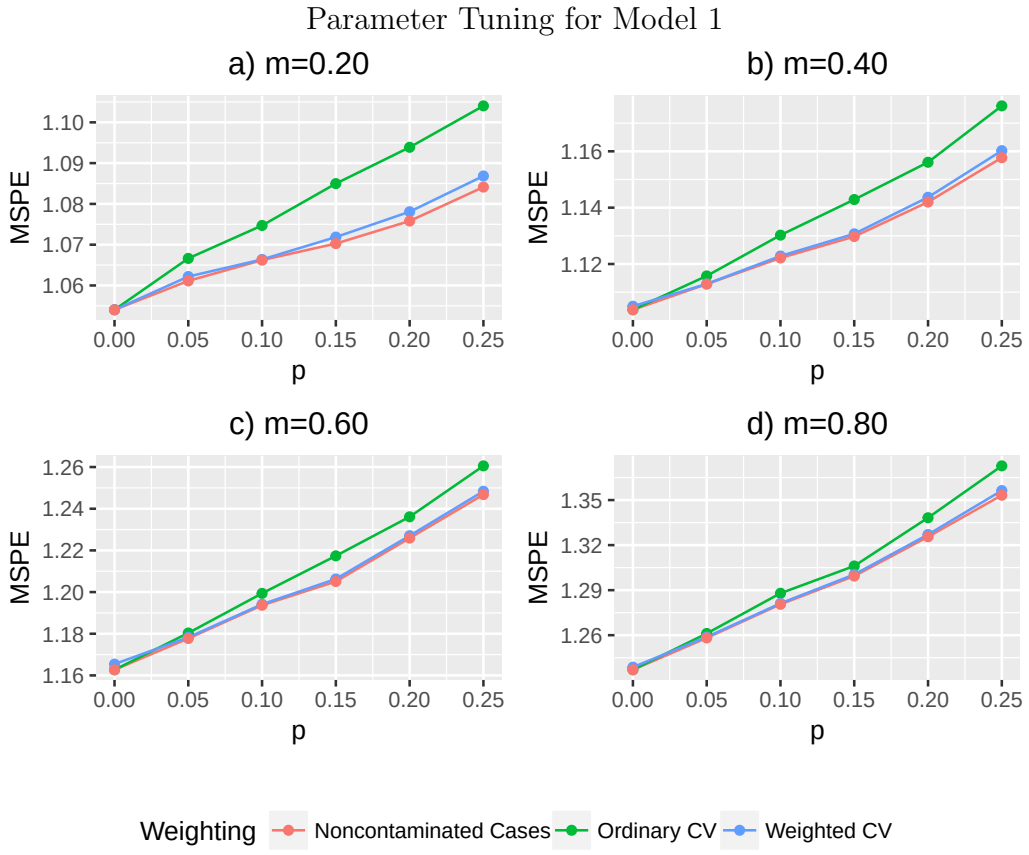

Figure 4.10: A comparison of the MSPE attained on test data, using ordinary and weighted cross-validation. We see that weighted cross-validation comes closer to attaining the ideal tuning parameter choices that would result from using only non-contaminated training cases.

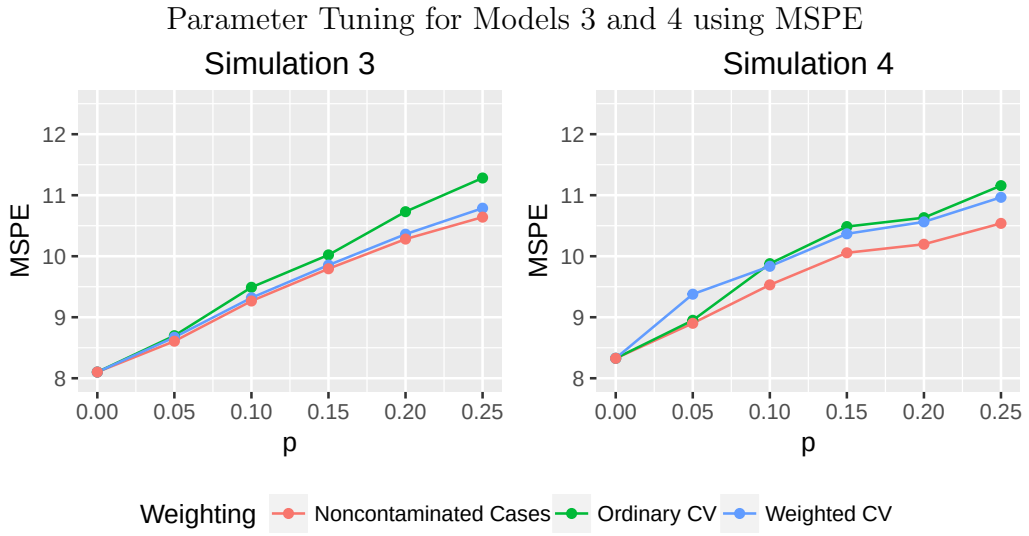

Figure 4.11: A comparison of the cross-validation approaches in simulations 3 and 4, when predictions are evaluated using MSPE. Weighting is very beneficial in Simulation 3, but less impactful in simulation 4.

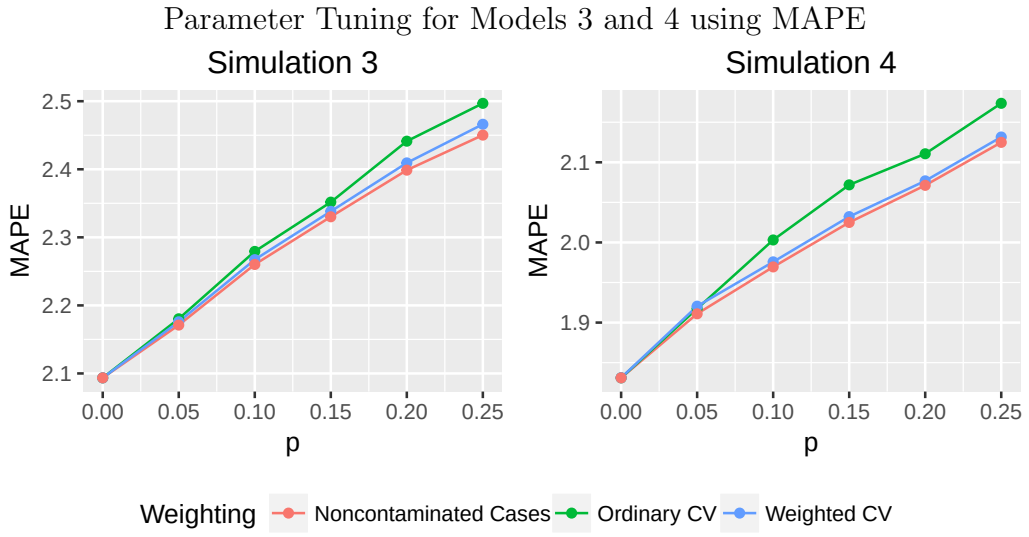

Figure 4.12: A comparison of the cross-validation approaches in simulations 3 and 4, when predictions are evaluated using MAPE. Weighted cross-validation outperforms ordinary cross-validation in both situations.

CHAPTER 5. GENERAL CONCLUSION

Random forest methodology has emerged as a powerful nonparametric, machine learning approach, capable of both making predictions and providing insight on the predictive importance of explanatory variables. Random forests have have gained popularity in a wide range of applications, but improving the methodology and better understanding and interpreting random forest results remain open areas of research. In this dissertation, we have explored three important topics related to random forest methodology, shedding light on the impacts of 1) aggregation on class probability estimation, 2) the imputation of missing values on measures of variable importance, and 3) training data response contamination on predictions. We have provided recommenations for each situation, and introduced a new robust method capaple of strong performance in the third scenario.

In Chapter 2, we compared the partitioning and aggregation approaches implemented in the `randomForest` (Liaw and Wiener, 2002) and `party` (Hothorn et al., 2006a; Strobl et al., 2007, 2008) `R` packages. We showed that while the partitioning algorithm implemented by `party` can be advantageous, this method's weighted tree aggregation approach is prone to underestimating probabilities of unlikely outcomes in classification problems. We further showed that combining the permutation-test-based partitioning algorithm implemented in `party` with the aggregation approach used by `randomForest` often yields results that are superior to those obtained using either method on its own.

In Chapter 3, we considered the impact of six random-forest-based imputation techniques on estimates of variable importance. We showed that some commonly used approaches, such as the `rfimpute` function in `randomForest`, often lead to inflated measures of importance for variables with a large percentage of missing values. On the other hand, some imputation

techniques produce results that suggest important variables are not important at all if a substantial percentage of values are missing. While no imputation approach was able to reproduce variable importance results obtained on complete data, in our simulation studies, the multiple imputation approach implemented in `RF-mice` (Doove et al., 2014) typically comes the closest, without leading to artificially inflated importance estimates for variables with missing values. Future research might focus on developing imputation methods, or new measures of variable importance, that provide a more informative assessment of variable importance when data are missing.

In Chapter 4, we proposed a new residual-based approach, RF-LOWESS, for improving random forest robustness in the regression setting. This method shows strong potential for applications involving training data contamination. In simulation studies, our approach outperforms existing techniques when data contamination is present and the signal-to-noise ratio is not very strong. We also proposed a new method for weighting training cases when performing cross-validation in situations where contamination is suspected in the training data, but not test data. We tested this algorithm in the context of parameter tuning associated with our RF-LOWESS algorithm, but it could be applied more broadly. Our method also enjoys the advantage of being self-correcting in the sense that it is capable of automatically reverting back to ordinary random forest predictions in situations where our robust enhancement is unhelpful. This allows our approach to improve performance on noisy and contaminated data, without damaging predictions in situations where contamination is not a concern.

We used a number of criteria for assessing the performance of the methods under consideration. These criteria were selected based upon the primary objective in each situation. In Chapters 2 and 4, we evaluated methods primarly by using loss functions, such as log loss or mean square prediction error, as is common in situations where predictive ability is of primary interest. In Chapter 2, we also considered the calibration of probability estimates

which was of interest in the application considered. In Chapter 3, the primary concern was not predictive performance, but rather the change in estimated variable importance before and after values were deleted and imputed. Therefore, we compared methods by assessing the change in estimated importance before and after imputation.

As random forests and other machine learning algorithms continue to evolve to meet the needs of researchers seeking to analyze increasingly complex data, established statistical procedures can provide valuable insight. By generalizing an earlier statistical technique and applying it to random forests, we were able to improve robustness in situations where training data responses contain contamination. Future research might focus on how statistical procedures might be extended to improve machine learning algorithms and increase their interpretability. Understanding and quantifying the uncertaintly associated with random forest estimates remains a challenge and advances in this area would further enhance the interpretability of results obtained via random forest methodology.

BIBLIOGRAPHY

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836.

Doove, L., Van Buuren, S., and Dusseldorp, E. (2014). Recursive partitioning for missing data imputation in the presence of interaction effects. *Computational Statistics & Data Analysis*, 72:92–104.

Hapfelmeier, A., Hothorn, T., Riediger, C., and Ulm, K. (2014a). Estimation of a predictor's importance by random forests when there is missing data: risk prediction in liver surgery using laboratory data. *The International Journal of Biostatistics*, 10(2):165–183.

Hapfelmeier, A., Hothorn, T., and Ulm, K. (2012). Recursive partitioning on incomplete data using surrogate decisions and multiple imputation. *Computational Statistics & Data Analysis*, 56(6):1552–1565.

Hapfelmeier, A., Hothorn, T., Ulm, K., and Strobl, C. (2014b). A new variable importance measure for random forests with missing data. *Statistics and Computing*, 24(1):21–34.

Hothorn, T., Bühlmann, P., Dudoit, S., Molinaro, A., and Van Der Laan, M. J. (2006a). Survival ensembles. *Biostatistics*, 7(3):355–373.

Hothorn, T., Hornik, K., and Zeileis, A. (2006b). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674.

Hothorn, T., Lausen, B., Benner, A., and Radespiel-Tröger, M. (2004). Bagging survival trees. *Statistics in Medicine*, 23(1):77–91.

Ishwaran, H. and Kogalur, U. (2007). Random survival forests for R. *R News*, 7(2):25–31.

Ishwaran, H. and Kogalur, U. (2016). *Random Forests for Survival, Regression and Classification (RF-SRC)*. R package version 2.2.0.

Ishwaran, H., Kogalur, U., Blackstone, E., and Lauer, M. (2008). Random survival forests. *Annals of Applied Statistics*, 2(3):841–860.

Li, A. H. and Martin, A. (2017). Forest-type regression with general losses and robust forest. In *International Conference on Machine Learning*, pages 2091–2100.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

Meinshausen, N. (2006). Quantile regression forests. *The Journal of Machine Learning Research*, 7:983–999.

Meinshausen, N. (2016). *quantregForest: Quantile Regression Forests*. R package version 1.3-5.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Roy, M.-H. and Larocque, D. (2012). Robustness of random forests for regression. *Journal of Nonparametric Statistics*, 24(4):993–1006.

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., and Zeileis, A. (2008). Conditional variable importance for random forests. *BMC Bioinformatics*, 9(1):1.

Strobl, C., Boulesteix, A.-L., Zeileis, A., and Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(1):1.