Statistics and Its Interface Volume 14 (2021) 389–402

A residual-based approach for robust random forest regression

Andrew J. Sage[*], Ulrike Genschel, and Dan Nettleton

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 a robust fitting procedure first proposed in the context of locally weighted polynomial regression and scatterplot smoothing. We demonstrate that tuning the parameter in the robustness algorithm using a weighted cross-validation approach is advantageous when contamination is suspected in training data responses. 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.

AMS 2000 subject classifications: Primary 62G35; Secondary 62G08.
Keywords and phrases: Data contamination, Robustness, Random forest.

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 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 variety of robust regression techniques are commonly used to reduce the impact of outliers on inference and prediction (c.f. [12, 23]). In this work, we use the term *outlier* to refer to any point that substantially deviates from most other observations, and we consider outlying observations in the response variable.

Random forest methodology, introduced by [1], 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. 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 a large impact on the prediction. This is a potential advantage over ordinary least-squares regression, for example, where as little as one observation can have a considerable influence on the fitted response surface. Still, modifications to splitting rules [3, 8, 18, 24, 15], and aggregation methods [19, 24, 15] have been shown to improve random forest robustness in regression problems. Roy and Larocque (hereafter [RL]), found that the aggregation approach has a stronger impact on robustness than the splitting criterion, and that Meinshausen's quantile regression forest [19] (hereafter [M]) achieves a high degree of robustness. Li and Martin (hereafter [LM]) proposed a method for modifying training case weights in order to improve robustness, but the use of residuals in this context has not been previously studied.

We introduce a new approach for robust random forest regression. When predicting the response value for a new case, our approach modifies initial random forest training case weights so that training cases with large residuals receive smaller weights, reducing their influence on the prediction. This approach is akin to the robust fitting procedure proposed by [5] for locally weighted regression and scatterplot smoothing (LOWESS), so we refer to the proposed method as RF-LOWESS. We generalize Cleveland's LOWESS approach by treating a value previously taken to be constant as a tuning parameter, and introduce a procedure for parameter tuning in situations where contamination is suspected in training data but not test data.

The proposed RF-LOWESS method differs from previous approaches, most notably the approach of [LM], in three important ways.

1. RF-LOWESS calculates case weights using residuals. A benefit of this approach is illustrated in Subsection 3.2.
2. In addition to providing robust predictions, RF-LOWESS provides a straightforward way to detect potential outliers in training data.

[*]Corresponding author.

3. RF-LOWESS can be easily tuned, using a weighted cross-validation procedure that is advantageous when contamination is present in training data.

We perform a thorough investigation of the performance of RF-LOWESS to other robust random forest regression techniques ([RL], [M], and [LM]), some of which have not been directly compared in prior literature. We show that RF-LOWESS outperforms existing robust approaches on noisy datasets with low signal-to-noise ratio, and is consistently competitive with the best existing robust methods.

The remainder of the manuscript is structured in the following manner. In Section 2, we provide an overview of existing approaches for robust random forest regression. In Section 3, we present a detailed description of the newly proposed RF-LOWESS algorithm and give an example to demonstrate its potential benefits. Section 3 further includes a description of the aforementioned parameter tuning procedure. In Section 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 5.

2. 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) [2] is popular.

In CART, an individual decision tree is grown by partitioning training cases into two nodes in a way that minimizes the sum of the squared deviations between each response and the mean response in the corresponding node. All possible binary splits for each explanatory variable are considered. This approach is designed to ensure that the resulting nodes are as homogeneous as possible, with respect to the response variable. As an alternative to minimizing the sum of the squared deviations, Breiman [2] suggested choosing the split that minimizes the sum of the absolute deviations between each response and the median response in the corresponding node, which makes the choice of split less sensitive to outliers. Galimberti et al. [8] generalized this approach, using M-estimators in the splitting procedure. Loh et al. [18] propose splitting based on regression models fit in each node, while Hothorn et al. [11] propose splitting based on permutation tests. Regardless of which splitting criterion is used, splitting continues until nodes are either homogeneous with respect to the response variable, or with respect to all explanatory variables, or until they are smaller than a predetermined size.

Predictions are made by moving a new case through a tree in accordance with its predictor variable values and the splitting rules determined from the training data. Most often the prediction is given by the mean of the response values in the terminal node that contains the new case, although use of the median has also been proposed [24]. Individual tree predictions are often unstable, in the sense that a small change in the training data can have a profound impact on predictions. Breiman [1] suggested growing a forest of many trees, with each tree serving as a low-bias, high-variance predictor. Aggregating predictions across trees, often by averaging, has the effect of reducing variance, while maintaining low bias.

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 all training cases in that node. In Breiman's initial random forest algorithm, [1], an overall random forest prediction is obtained by averaging predictions across trees.

Several modifications to the way random forest predictions are aggregated within and across trees have been proposed. These have been shown to improve random forest robustness when training data contamination is present. [M] introduced a quantile regression forest, which was shown to improve robustness. [RL] showed 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. [LM] introduced a framework for robust random forest regression using general loss functions. The ensuing paragraphs provide detail on these aggregation approaches.

Lin and Jeon [17] 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. These weights play an important role in the quantile regression forest [19], and Huber Forest [15], as well as in our proposed RF-LOWESS technique. 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,

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

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

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

and the predicted value is

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

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

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

The `quantregForest` package [20] in `R` [26] can be used to estimate any quantile of the conditional distribution of Y given $\boldsymbol{X} = \boldsymbol{x}$ in this manner. Each prediction weight, $w_i(\boldsymbol{x})$ can be obtained using the `randomForestSRC` package [13, 14]. [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.

[LM] showed that the ordinary random forest estimator, and the quantile regression forest estimator can be viewed as special cases in a general framework for random forest regression. [LM] implemented the pseudo-Huber [4] and Tukey bi-weight [21] loss functions in order to improve robustness and found that the pseudo-Huber function generally achieved superior performance in their procedure. Given the initial random forest weights $w_i(\boldsymbol{x})$, $i = 1, 2, \ldots, n$, defined in (1), and a predetermined value for the tuning parameter δ, the (pseudo) Huber estimator can be expressed as

$$\widehat{y}^{(pH)}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} w_i^{(pH)}(\boldsymbol{x}) y_i}{\sum_{i=1}^{n} w_i^{(pH)}(\boldsymbol{x})}, \text{ where} \tag{2}$$

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

[LM] provide an algorithm that can be used to obtain this estimate numerically. Keeping with the terminology of [LM], we refer to estimates obtained in this manner as Huber forest estimates.

The Huber forest estimate for a new case $\boldsymbol{x}$ is obtained by modifying the initial random forest weight $w_i(\boldsymbol{x})$ based on the difference between the observed response y_i and the prediction $\widehat{y}^{(pH)}(\boldsymbol{x})$. This approach places heavy weight on training cases with response values close to those of cases with the largest $w_i(\boldsymbol{x})$. It does not consider whether these heavily contributing cases might, themselves, be the result of contamination, and thus potentially unreliable.

3. RF-LOWESS METHOD

3.1 Motivation and algorithm

Like the Huber forest estimator, RF-LOWESS uses the random forest prediction weights $w_i(\boldsymbol{x})$ as a starting point for predicting Y, given $\boldsymbol{x}$. These weights are then modified in accordance with the size of each training case's residual, rather than the proximity of the training value y_i and predicted value $\widehat{y}$. Residuals are calculated using OOB predictions, eliminating the possibility of a training case heavily influencing its own prediction.

We apply Tukey's bi-weight function, which is defined as

$$B(t) = \begin{cases} (1 - t^2)^2 & \text{if } |t| < 1 \\ 0 & \text{if } |t| \ge 1 \end{cases} \tag{3}$$

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 the analogous step of the LOWESS algorithm [5], uses the fixed value $\alpha = 6$. We will demonstrate in Section 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 4.1.

We denote the OOB prediction for training case j as $\widehat{y}_{OOB}(j)$. 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, $i \ne j$ 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 (2).

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

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
       $$\widehat{y}^{(1)}_{OOB}(j) = \sum_{i=1}^{n} w_{OOB_i}(j) y_i, \text{ for } i = 1, 2, \ldots, n.$$
2. Set $l = 1$ and perform the following steps:
    i. Calculate residuals $e^{(l)}_j = y_j - \widehat{y}^{(l)}_{OOB}(j)$ for $j = 1, 2, \ldots n$.
    ii. Set $m^{(l)} = \text{Median}\left\{\left|e^{(l)}_j\right|\right\}_{j=1}^{n}$.
    iii. Let
           $$\lambda^{(l)}_j = B\left(\frac{e^{(l)}_j}{\alpha m^{(l)}}\right),$$
         for $j = 1, 2, \ldots, n$, where $\alpha \in \mathbb{R}^+$ is a tuning parameter discussed in Section 3.3 and $B$ denotes Tukey's bi-weight function.
    iv. For $j = 1, 2, \ldots n$, let
           $$\widehat{y}^{(l+1)}_{OOB}(j) = \frac{\sum_{i=1}^{n} \lambda^{(l)}_i w_{OOB_i}(j) y_i}{\sum_{i=1}^{n} \lambda^{(l)}_i w_{OOB_i}(j)}.$$
    v. If
           $$\frac{1}{n}\sum_{j=1}^{n}\left(\widehat{y}^{(l+1)}_{OOB}(j) - \widehat{y}^{(l)}_{OOB}(j)\right)^2 \le \varepsilon_0,$$
         (a non-negative user-specified convergence parameter), stop and return $\lambda_j = \lambda^{(l+1)}_j$ for $j = 1, 2, \ldots, n$. Otherwise, set $l = l + 1$ and repeat steps (i)–(v)
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
       (4)   $$\widehat{y}_{RFL}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} \lambda_i w_i(\boldsymbol{x}) y_i}{\sum_{i=1}^{n} \lambda_i w_i(\boldsymbol{x})}.$$
```

the proposed RF-LOWESS algorithm, given in Algorithm 1. Note that Algorithm 1 requires that each training case be OOB in at least one tree. Since forests with a large number of trees can be grown very quickly using the `randomForest` [16] and `randomForestSRC` [13, 14] R packages, among others, this is not a concern.

The RF-LOWESS prediction in (4) 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 Huber forest approach of [LM] in which local prediction weights are calculated iteratively for each new case.

Step (iii) in the RF-LOWESS algorithm is based on the intuition that training cases with large OOB residuals should be viewed as outliers, and potentially down-weighted. Each training case's OOB residual is compared to the OOB residuals, and cases with large residuals are down-weighted in accordance with the size of the residual. The choice of the Tukey bi-weight function is motivated by its use in the original LOWESS algorithm [5]. An advantage of this function is that it provides flexibility by down-weighting potential outliers with respect to the size of their OOB residuals. For predictions that depend only on training cases with small OOB residuals, no training cases are heavily down-weighted or excluded. Thus, RF-LOWESS does not throw away potentially useful information, as other robust approaches, such as censoring or winsorizing would. On the other hand, when a prediction depends on one or more training cases with large OOB residuals, these training cases are down-weighted to varying degrees, in accordance with the size of their OOB residuals. Training cases with OOB residuals more than α times as large as the median of the OOB residuals are discarded entirely from the prediction. Thus, the Tukey bi-weight function provides a flexible approach for down-weighting potential outliers.

Another advantage of RF-LOWESS, compared to [LM] is the ability of RF-LOWESS to detect possible outliers in training data. The values $\{\lambda_j\}_{j=1}^{n}$ can be used to identify unusual training cases. A value of λ_j close to zero, suggests that y_j is substantially different than expected, given $\boldsymbol{x}_j$, and hence case j can be thought of as a potential outlier. We recommend that practitioners carefully examine cases with $\lambda_j < 0.5$. Doing so might reveal an error in data entry, or point to unexpected structure. This will help the practitioner determine whether a robust prediction approach such as RF-LOWESS is appropriate. While the approach of [LM] down-weights certain training cases in prediction, it does not provide a straightforward method for identifying which cases are being down-weighted, making outlier detection more difficult.

The termination criterion in Algorithm 1 is designed to ensure that changes between OOB predictions are sufficiently small. The squared difference could reasonably be replaced with a different criterion, such as absolute difference, if desired. Cleveland [5] observed that in the original LOWESS algorithm, the greatest changes in weighting occur during the first iteration. We also find this to be true in RF-LOWESS.

In rare instances, Algorithm 1 might fail to achieve convergence. This is usually the result of having cases with

Table 1. Case weights for estimating $E(y|x = 1.4)$, using RF, RF-LOWESS, and Huber forest

Label	x_i	y_i	$\widehat{y}^{(1)}_{RFOOB}$	$e^{(1)}_i$	λ_i	RF weight ($w_i(1.4)$)	RF-LOWESS weight	Huber weight
A	1.442	0.211	0.577	-0.366	0.014	0.393	0.001	0.259
B	1.621	0.512	0.775	-0.264	0.353	0.172	0.114	0.483
	1.518	1.003	0.821	0.182	0.924	0.114	0.229	0.052
	1.659	0.902	1.069	-0.168	0.943	0.050	0.104	0.029
	1.335	0.922	0.832	0.090	0.987	0.049	0.106	0.027

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 [5] suggested with respect to the original LOWESS algorithm, we find it reasonable to stop the RF-LOWESS algorithm after a fixed number of iterations, and ten iterations have been sufficient in all circumstances we have considered. Alternatively, in situations where the algorithm does not converge, users might choose to use the set of values $\{\lambda_i^{(l)}\}_{i=1}^n$ from the iteration l that resulted in the smallest value of $m^{(l)}$.

RF-LOWESS predictions and weights are provided by the R package RFLOWESS [25], which is available at https://github.com/AndrewjSage/RFLOWESS.

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.

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

$$y_i = \sin(x_i) + \varepsilon_i \mathbb{1}(r_i \geq 0.1) + \gamma_i \mathbb{1}(r_i < 0.1), \tag{5}$$

where $\varepsilon_i \sim \mathcal{N}(0, 0.1), \gamma_i \sim \mathcal{N}(0, 0.5)$, $r_i \sim \text{Uniform}(0, 1)$, 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 1 displays the data along with the true expected response curve, and the estimates obtained using an ordinary random forest with default settings and also RF-LOWESS with $\alpha = 6$. We see the random forest estimate

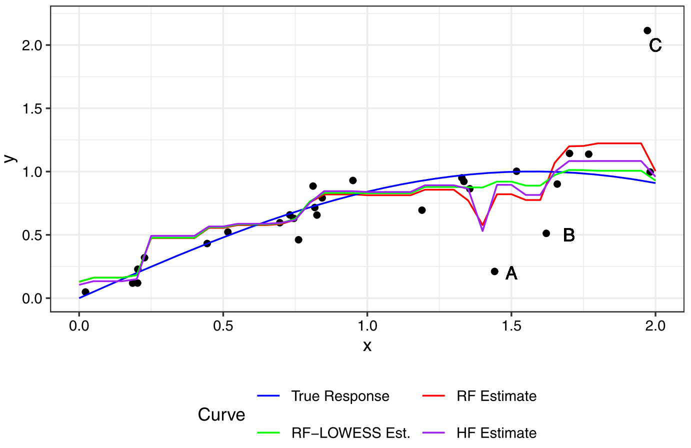

Figure 1. True and estimated response surface using a random forest and RF-LOWESS.

of the expected response curve is pulled below the true response curve near $x = 1.5$, as the result of two potential outliers, labeled A and B. Likewise, the estimated response curve is pulled upward near $x = 2$, as a result of observation C. RF-LOWESS, though not entirely immune from the impact of these potential outliers appears to be more resistant. The Huber forest is fairly resistant to outlier C, but not to outliers A and B, for reasons described below.

Consider estimating $E(Y|X = 1.4)$. We see in Table 1 that cases A and B contribute most heavily to this prediction. As a result, we obtain a random forest estimate of $\widehat{y}_{RF}(1.4) = 0.577$, which is considerably less than $\sin(1.4) \approx 0.985$. Because cases A and B have large residuals, RF-LOWESS reduces their prediction weights from 0.393 and 0.172 to 0.001 and 0.114, respectively. Other training cases with smaller residuals contribute more heavily, yielding an RF-LOWESS estimate of 0.874.

On the other hand, the Huber forest algorithm increases the weight given to case B. This happens because its response value 0.512 is closer to $\widehat{y}_{RF}(1.4) = 0.577$ than any other training case. In this instance, re-weighting based on proximity with respect to the response variable leads to additional weight being placed on a case that appears to be a potential outlier. As a result, the Huber forest estimate of 0.530 is farther from the expected response than the original random forest estimate.

The low λ-values associated with cases A and B identify these as potential outliers. While the outlying nature of these cases is visually apparent in Figure 1, RF-LOWESS can also be used in high-dimensional situations, where outlier detection is more challenging.

In this example, we used the values of $\alpha = 6$ in the RF-LOWESS algorithm and $\delta = 0.10$ in the Huber forest algorithm. The value of α was chosen because it was used in the original LOWESS algorithm [5]. The value of δ was chosen 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.

The example in this section is merely intended to illustrate the RF-LOWESS procedure and shed light on an important difference between RF-LOWESS and the Huber forest approach. Although we have illustrated a scenario in which RF-LOWESS performs favorably, it should not be inferred that this will be true in general. The performances of these and other approaches are investigated more thoroughly in Section 4.

3.3 Parameter tuning

In order to add flexibility to the RF-LOWESS algorithm, 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. A small value of α leads to aggressive down-weighting of cases with large residuals, while a 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 contamination is less of a concern.

Cross-validation, which is commonly used to set tuning parameters in machine learning, 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 of outlying response values in the data withheld for evaluation when choosing α, we use a weighted loss function during cross-validation. Cases in the withheld training data that have large residuals, and hence are believed to have possibly resulted from contamination not present in the test data, are weighted less heavily when calculating loss than cases with small residuals.

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 withheld. Let $\mathcal{I}$ represent a set of candidate values under consideration for α.

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 minimizing value of α to make predictions for new cases. Since Algorithm 2 is used to tune the value of α in Algorithm 1, it is necessary to use a fixed value in place of α in Algorithm 2, in order to avoid an endless cycle of tuning. Cleveland's original LOWESS algorithm [5] fixed α at 6 and used no tuning. Following Cleveland's lead, we adopt $\alpha = 6$ for Algorithm 2.

Algorithm 2 Weighted Cross-Validation Tuning Algorithm (WCV)

```
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 (3).
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 $\alpha$ 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.
```

[LM] state that cross-validation could be used to set the value of the tuning parameter, δ, used in their Huber forest, although the potential impact of training data contamination is not discussed. The weighted cross-validation approach we describe could reasonably be applied in the context of the Huber forest as well. However, because the Huber forest approach requires iteratively calculating weights for each case being predicted, tuning for δ is more computationally expensive than tuning for α in the RF-LOWESS approach. The ability to compute weights once, and apply them to all predictions is an advantage of RF-LOWESS.

The values of α in RF-LOWESS and δ in the Huber forest approach do not affect the tree-growing process. Consequently, 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 [16]. 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. SIMULATION STUDIES AND REAL DATA RESULTS

We now investigate the performance of RF-LOWESS compared to the other robust random forest regression techniques described in Section 2. We apply these techniques to simulation studies and real datasets studied by [RL] and [LM]. In addition to evaluating the effectiveness of RF-LOWESS, we provide the first comparison of the Huber forest method to the median-based aggregation approaches suggested by [RL].

4.1 Simulation studies

In each simulation, we introduced contamination by generating a proportion p of training response observations from a distribution with larger variance than the distribution used to generate the remaining response values. Like [RL], we consider values of p equal to 0, 0.05, 0.10, 0.15, 0.20, and 0.25, and 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 introduced 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

$$\begin{aligned}
Y_i = s \times \Big(&\mathbb{1}((X_{1i} \le 0), (X_{2i} \le 0)) \\
&+ 2\mathbb{1}((X_{1i} \le 0), (X_{2i} > 0), (X_{4i} \le 0)) \\
&+ 3\mathbb{1}((X_{1i} \le 0), (X_{2i} > 0), (X_{4i} > 0), (X_{6i} \le 0)) \\
&+ 4\mathbb{1}((X_{1i} \le 0), (X_{2i} > 0), (X_{4i} > 0), (X_{6i} > 0)) \\
&+ 5\mathbb{1}((X_{1i} \le 0), (X_{3i} \le 0)) \\
&+ 6\mathbb{1}((X_{1i} \le 0), (X_{3i} \le 0), (X_{5i} \le 0)) \\
&+ 7\mathbb{1}((X_{1i} \le 0), (X_{3i} \le 0), (X_{5i} > 0)\Big) \\
&+ \varepsilon_i \mathbb{1}(r_i \ge p) \\
&+ \gamma_i \mathbb{1}(r_i < p)),
\end{aligned}$$

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

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

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

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

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

Simulation 3

In this simulation, taken from [LM], $\boldsymbol{X} \sim \mathcal{N}_{10}(\mathbf{0}, \Sigma)$, where either a) $\Sigma = I_{10}$, or b) Σ is a Toeplitz matrix with $\rho = 0.7$, resulting in correlated predictors. In each setting,

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

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

Each simulation was repeated 500 times for each value of p (and for each value of s in simulations 1 and 2), using training and test sets that each consisted of 1,000 observations.

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

1. Ordinary random forest (RF)
2. Quantile random forest, using 0.5 quantile (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 and Martin's Huber forest (Huber)
6. RF-LOWESS (RFL)

We also considered predictions obtained by using the median when aggregating predictions within terminal nodes, and the mean when aggregating across trees (Med.-Mean), as well as Li and Martin's Tukey forest. These approaches, however, performed considerably worse than the Mean-Med., Med.-Med., and Huber forest techniques, which is consistent with the findings of [RL] and [LM]. For the sake of readability, these results are omitted from Figures 2–5.

QRF predictions were obtained using the `quantregForest` package [19] in `R`, while forests for all other methods were grown using the `randomForestSRC` package [13, 14]. The *nodesize* parameter was set to 7 in simulations 1 and 2, and 5 in simulations 3 and 4. These choices were made to closely mirror the values used by [RL] and [LM], accounting for differences between the way these parameters are defined in `randomForestSRC`, compared to the `randomForest` package used in the previous studies. 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 set the value α to be used in 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 error tolerance was set to $\varepsilon_0 = 10^{-6}$ for RF-LOWESS predictions.

It is conceivable that parameter tuning could also be used for the value of δ in the Huber forest algorithm. In order to account for contaminating outliers in the training data, that are not present in the test data, a method similar to the WCV approach, discussed in Section 3.3 would be needed. This, however, would become computationally expensive, since applying Algorithm 2 to the Huber Forest would require iteratively recalculating case weights for each of the 200 cases in D_2, for each prospective value of α, making this step of the algorithm 200 times as expensive as it is when tuning RF-LOWESS. Therefore, we used the value of $\delta = 0.005$, suggested by [LM], and did not attempt to tune the Huber forest algorithm. The error tolerance was again set to $\varepsilon_0 = 10^{-6}$.

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 only in situations where there were notable differences. Additional results are provided in the supplementary material, http://intlpress.com/site/pub/files/_supp/sii/2021/0014/0004/SII-2021-0014-0004-s001.pdf.

Figures 2 and 3 display average MSPE as a function of p for each of the four values of s that we considered. Error bars, representing 95% confidence intervals are shown at each value of p, and are slightly staggered for aesthetic reasons. We see that RF-LOWESS consistently outperforms the other techniques in Simulation 1. Differences are especially notable for lower signal-to-noise ratios ($s = 0.20$, $s = 0.40$) and large values of p. In Simulation 2, RF-LOWESS

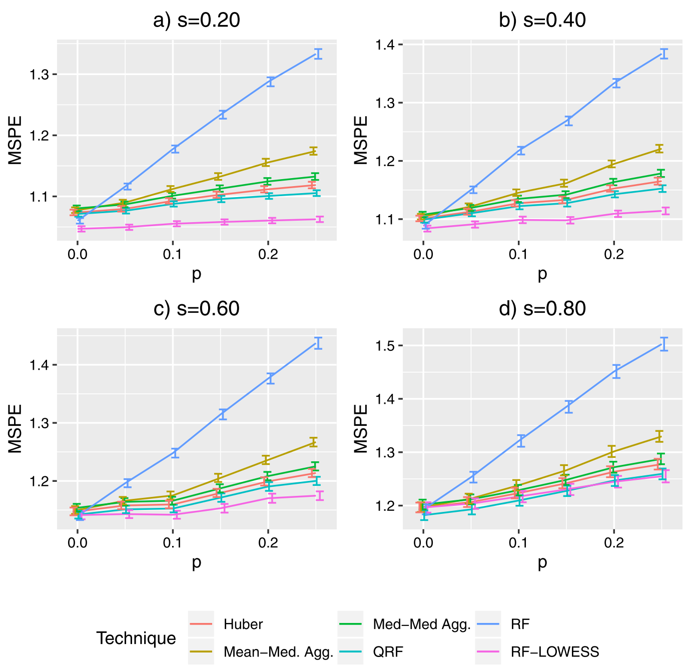

Figure 2. In Simulation 1, RF-LOWESS achieves the lowest MSPE when contamination is present, for $s = 0.20$, $s = 0.40$, and $s = 0.60$. When the signal-to-noise ratio is very large (s=0.80), RF-LOWESS and QRF achieve similar performance for large p.

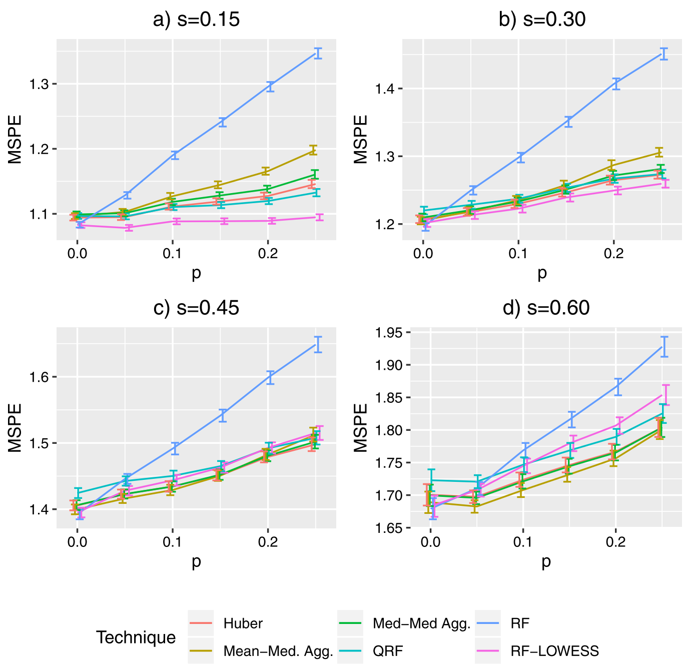

Figure 3. In Simulation 2, RF-LOWESS achieves the lowest MSPE for moderate signal-to-noise ratios, when contamination is present. Other approaches, including Huber forest, and median-based aggregation, perform best when the signal-to-noise ratio is strong.

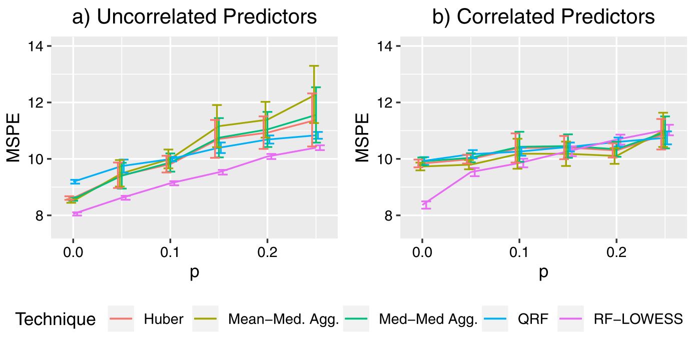

Figure 4. In Simulation 3(a), RF-LOWESS consistently achieves the lowest MSPE. In Simulation 3(b), RF-LOWESS performs best for small p, but other approaches achieve similar or better performance for large p.

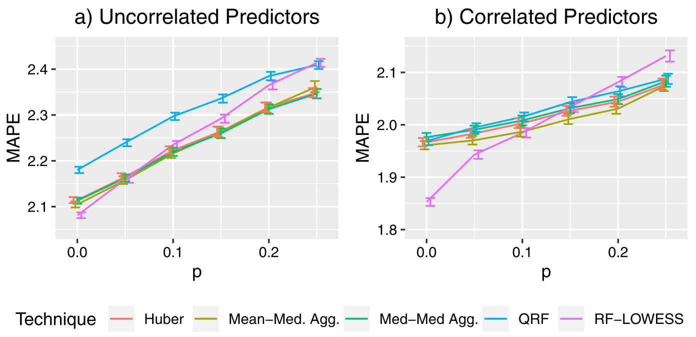

Figure 5. In Simulation 3(a), the Huber forest and median-based approaches achieve the best MAPE for large p. In simulation 3(b), RF-LOWESS achieves the best MAPE for small p, but is surpassed by the other approaches for large p.

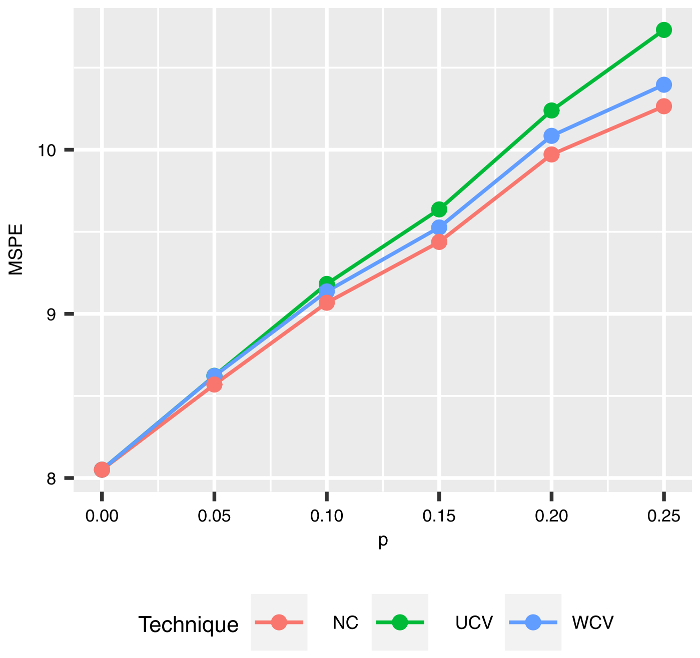

Figure 6. In Simulation 3(a), the weighted cross-validation tuning procedure yields results closer to the ideal results that would be obtained using only noncontaminated cases than ordinary cross-validation, when contamination is present.

again achieves the best performance for low signal-to-noise ratios, but is surpassed by other robust approaches for datasets with stronger signal. The median-based and Huber forest aggregation approaches perform best in these situations. When contamination is present, all of the robust approaches substantially outperform the ordinary random forest approach.

In order to test the performance of each technique in a high-dimensional setting, we performed a second version of Simulation 2, by adding 94 explanatory variables that were unrelated Y_i, while maintaining the same expected response function. The results of this simulation were largely consistent with those shown in Figure 3, and are included in the supplementary material.

Figure 4 displays the results for Simulations 3(a) and 3(b), using MSPE as the evaluation criterion. In this simulation, results differ when MAPE is used, as seen in Figure 5. MSPE and MAPE are extremely high for RF, relative to the other methods, making RF difficult to visualize. Hence, the MSPE and MAPE for RF are excluded from these plots.

We see that RF-LOWESS achieves the best performance when MSPE is used to evaluate predictions in Simulation 3(a), but that it is suboptimal for large p when MAPE is applied. This is explained by the nature of the contamination in this simulation. Because contamination cases were generated from a heavy tailed t-distribution with two degrees of freedom, some extreme outliers occur in the training data. Although these extreme cases only have substantial impact on predicting a few test cases, predictions that are influenced are impacted very heavily. RF-LOWESS, due to its use of residuals, is able to recognize and down-weight these extreme outliers more effectively than the other methods, leading to very large improvements on a small number of test cases. Hence, the impact of RF-LOWESS is much heavier for MSPE than for MAPE. 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, is context dependent.

When correlation exists among the explanatory variables, as in Simulation 3(b), prediction error is typically lower, and differences between RF-LOWESS and the other methods are not as pronounced. Other methods overtake RF-LOWESS for large p, with the median-based approaches and Huber forest achieving the best MAPE in both for large p in these simulations.

We now turn our attention to the importance of the choice of the tuning parameter α, and the impact of the weighted cross-validation approach described in Algorithm 2. Figure 6 shows the MSPE values obtained in Simulation

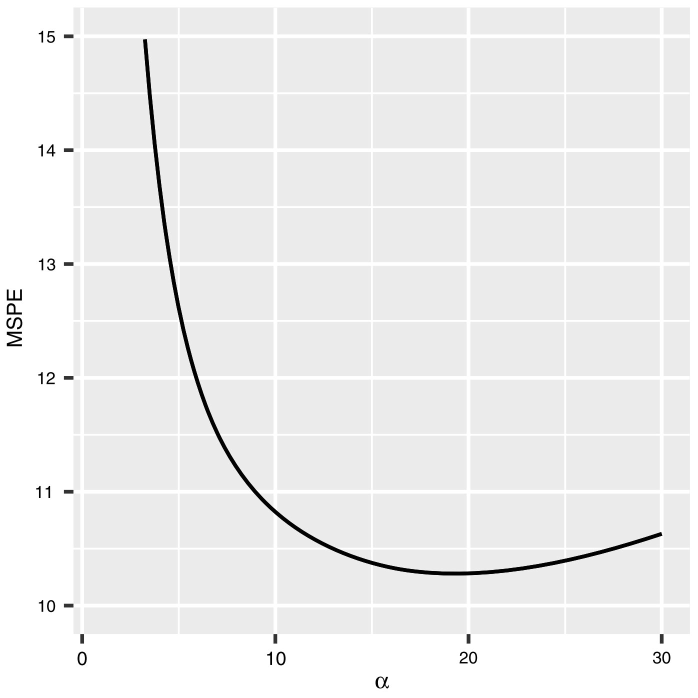

Figure 7. The optimal choice of α in Simulation 3(a) is between 15 and 20.

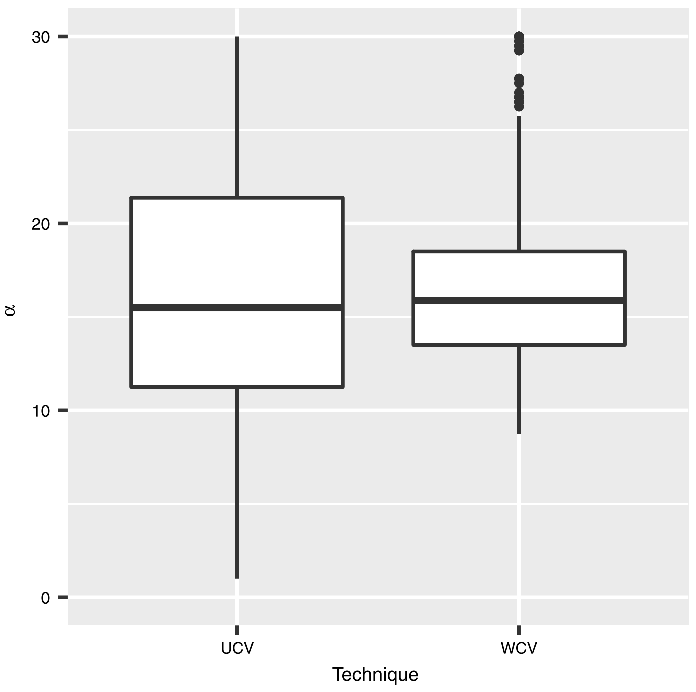

Figure 8. WCV is able to more consistently select values within this range than UCV.

3(a) for RF-LOWESS predictions on test data, after tuning the method using ordinary, unweighted cross-validation (UCV), weighted cross-validation (WCV), and the results that would have been obtained in the idealized scenario where we could tune the algorithm using only noncontaminated training cases and ordinary cross-validation (NC). This last strategy is, of course, impossible in practice.

We see that the weighted cross-validation approach comes closer to matching the ideal performance than ordinary CV. By down-weighting outlying training cases when evaluating predictions, the weighted approach yields a more robust choice of α, resulting in better performance on the uncontaminated test data.

The impact of weighted cross-validation is further illustrated in Figures 7 and 8. Shown are the average MSPE on test data as a function of α, and the distribution of choices of α, selected by each cross-validation approach across the 500 repetitions of the simulation. We see that the optimal performance is obtained for values of α between 15 and 20. Performance suffers substantially when α is chosen to be too small, and also declines for large α. Weighted cross-validation more consistently leads to values of α close to this optimal range than unweighted cross-validation. Omitted from the boxplots are nine instances in which UCV leads to a choice of $\alpha > 30$.

We conclude this section with a discussion of the efficiency of the RF-LOWESS algorithm. When the algorithm is applied without parameter tuning, RF-LOWESS resulted in an approximate time increase of only about 3%, compared to the standard random forest algorithm. Thus, users could implement the algorithm, using $\alpha = 6$ as in the original LOWESS algorithm, with very little loss in efficiency. Because RF-LOWESS recalculates weights once, using OOB predictions, the size of the test set does not impact the relative increase in time required. This is an advantage over the Huber forest algorithm, which recalculates training case weights for each new prediction. In our implementation of the Huber forest, we observed an approximate time increase of 4% when making predictions on a test set of size 10, but this grew to approximately 30% and 400% for test sets of 100, and 1,000, respectively.

When parameter tuning was performed, using Algorithm 2 with five-fold cross validation, the entire process took about 15 times as long as the ordinary random forest algorithm with no parameter tuning. This is explained by the need to grow two additional random forests and iteratively recalculate weights within each fold of the cross-validation process. While this process can be time consuming, it has the advantage of not needing to regrow a forest for each proposed value of α, as would need to be done when tuning standard random forest parameters, such as maximum allowable terminal nodesize. Regardless of how many values of α are tested, only two new forests need to be grown within each fold of the cross-validation. We did not attempt to test the Huber forest, using cross validation, due to the issues described previously, but any attempt to do so would incur the challenges of growing additional trees in each fold, as well as recalculating weights for each new test case.

Table 2. Ratio of MSPE to that of ordinary RF for real datasets. When a robust approach improves on RF, the best robust approach is shown in bold

Dataset	N	p	k	Cont.	QRF	Huber	Mean-Med.	Med-Med	RFL
Airfoil	1,503	5	9	No	1.833	**0.955**	0.989	1.029	1.000
Ames	2,930	80	10	No	1.083	1.038	1.032	1.040	1.100
Auto	392	7	8	No	1.034	1.013	1.001	1.018	1.058
Birthwt	189	9	9	No	**0.966**	1.008	1.019	1.008	0.972
Boston	506	13	11	No	**0.948**	1.094	1.068	1.086	1.254
CCRI	1994	102	10	No	1.026	1.002	**0.999**	1.002	1.071
Comp	209	6	11	No	1.033	1.001	**0.936**	1.031	1.991
Conc	1030	8	10	No	1.010	0.992	**0.985**	1.012	1.000
CSLU	103	7	4	No	**0.885**	0.969	0.930	0.960	1.000
Serv	167	4	5	No	1.495	1.147	**0.938**	1.121	1.111
Average				No	1.091	1.008	**0.995**	1.020	1.156
Airfoil	1,503	5	9	Yes	0.526	0.330	0.540	0.357	**0.317**
Ames	2,930	80	10	Yes	0.449	**0.434**	0.464	0.444	0.442
Auto	392	7	8	Yes	0.280	0.317	0.447	0.328	**0.266**
Birthwt	189	9	9	Yes	0.717	0.767	0.838	0.776	**0.697**
Boston	506	13	11	Yes	**0.378**	0.413	0.461	0.422	0.383
CCRI	1994	102	10	Yes	0.704	0.695	0.700	0.697	**0.690**
Comp	209	6	11	Yes	0.522	0.371	0.547	**0.360**	0.557
Conc	1030	8	10	Yes	0.505	0.333	0.552	0.352	**0.220**
CSLU	103	7	4	Yes	0.590	0.580	0.639	0.605	**0.566**
Serv	167	4	5	Yes	0.546	**0.402**	0.523	0.410	0.440
Average				Yes	0.522	0.464	0.571	0.475	**0.458**

4.2 Data applications

Next, we apply the robust regression techniques under consideration to ten real datasets. These include the Airfoil, Auto, Communities and Crime (CCRI) [22], Computer Hardware (Comp), Concrete Compressive Strength (CCS) [27], Concrete Slump Test (CSLU) [28], and Servo datasets available in the UCI Machine Learning Repository [7], along with the Ames Housing (Ames) [6], Boston Housing (Boston) [9], and Low Birthweight (Birthwt) [10] datasets.

Thirty repetitions of k-fold cross validation, were performed on each dataset, except the Ames Housing data, on which fifteen repetitions were performed, for computational reasons. Whenever possible, a value of k close to 10 was selected so that the entire training set could be divided into k equal-sized folds. Predictions were assessed using the initial datasets, without adding any training data contamination, 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 15% of training cases, chosen randomly, 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]. Table 2 provides the number of observations (N), the number of explanatory variables p, and number of folds k used in the evaluation, as well as whether contamination (Cont.) was introduced. Results are expressed as a ratio of the MSPE achieved by each robust approach to that achieved by the ordinary RF method. Table 3 displays analogous results using the MAPE evaluation criterion.

When contamination is present, the robust approaches consistently outperform the ordinary RF by wide margins. RF-LOWESS achieves the strongest performance on six of the ten datasets, with respect to MSPE and on four with respect to MAPE. The stronger performance with respect to MSPE is due in large part to the ability of RF-LOWESS to substantially improve predictions on a small number of cases, as was seen previously in Simulation 3. On average, RF-LOWESS, Huber forest, and median-based aggregation perform best when contamination is present, with RF-LOWEST achieving the smallest MSPE, and Huber forest achieving the smallest MAPE.

It is unsurprising that the robust approaches achieve little improvement over ordinary random forest predictions when there is no contamination. Huber forest and median-based aggregation do achieve small gains on average, using MAPE. RF-LOWESS, rarely improves on ordinary random forest predictions, as it usually resorts to using large values of α, thereby keeping the original predictions intact.

The poor performance of RF-LOWESS on the Comp dataset, especially with respect to MSPE, is explained by the presence of two responses that are considerably larger than the rest. In situations where one of these observations was placed in the training set, and the other in the test set, RF-LOWESS heavily down-weights the contribution of the large training response, causing it to miss badly on the

Table 3. Ratio of MAPE to that of ordinary RF for real datasets. When a robust approach improves on RF, the best robust approach is shown in bold

Dataset	N	p	k	Cont.	QRF	Huber	Mean-Med.	Med-Med	RFL
Airfoil	1,503	5	9	No	1.411	**0.970**	0.986	1.010	1.000
Ames	2,930	80	10	No	1.036	1.010	1.009	1.011	1.016
Auto	392	7	8	No	1.000	0.994	**0.992**	0.995	1.012
Birthwt	189	9	9	No	**0.975**	0.990	1.002	0.989	0.991
Boston	506	13	11	No	**0.937**	0.984	0.981	0.983	1.036
CCRI	1994	102	10	No	0.944	**0.942**	0.943	0.943	0.975
Comp	209	6	11	No	0.986	0.962	**0.932**	0.959	1.137
Conc	1030	8	10	No	**0.901**	0.944	0.952	0.958	1.000
CSLU	103	7	4	No	**0.895**	0.961	0.945	0.958	1.000
Serv	167	4	5	No	1.030	0.825	**0.814**	0.830	1.009
Average				No	1.011	0.958	**0.958**	0.964	1.018
Airfoil	1,503	5	9	Yes	0.792	0.593	0.721	0.619	**0.591**
Ames	2,930	80	10	Yes	0.605	**0.591**	0.600	0.593	0.601
Auto	392	7	8	Yes	0.523	0.536	0.606	0.543	**0.517**
Birthwt	189	9	9	Yes	0.860	0.883	0.920	0.887	**0.857**
Boston	506	13	11	Yes	**0.544**	0.561	0.593	0.565	0.572
CCRI	1994	102	10	Yes	**0.758**	0.759	0.765	0.760	0.784
Comp	209	6	11	Yes	0.422	**0.404**	0.556	0.408	0.456
Conc	1030	8	10	Yes	0.568	0.542	0.677	0.561	**0.528**
CSLU	103	7	4	Yes	0.**727**	0.761	0.782	0.772	0.779
Serv	167	4	5	Yes	0.523	**0.432**	0.567	0.444	0.565
Average				Yes	0.632	**0.606**	0.679	0.615	0.625

prediction of the new case. The impact of these cases on MAPE is not nearly as strong, so RF-LOWESS performs better using this metric.

5. DISCUSSION

We have introduced a new residual-based approach for random forest regression, which improves robustness by down-weighting the predictive weight of training cases with large residuals. Through simulations and analysis of real data, we have shown that when contamination is present, RF-LOWESS is competitive with the best robust approaches, often outperforming them on noisy, contaminated datasets. RF-LOWESS is flexible enough to revert to ordinary random forest predictions when there is no evidence of training data contamination by using very large value of α. Therefore, the ordinary random forest approach can be seen as a special case of RF-LOWESS, with $\alpha = \infty$.

Like [RL], we did not find any one robust approach that uniformly outperformed 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 perform well in various situations. While RF-LOWESS and Huber forest both achieve strong predictive performance, RF-LOWESS has the advantage of only needing to recalculate prediction weights once, using residuals, as opposed to calculating weights each time a prediction is made.

In addition to the algorithm itself, we have described a weighted cross-validation procedure for setting tuning parameters when training data contamination is present. Ordinary cross-validation is unreliable in this situation, because of the impact of predictions made on training data outliers on the tuning parameter selection. Our weighted approach results in more consistent tuning parameter choices, closer to those that would be attained if we could evaluate only predictions on noncontaminated training cases during cross-validation. A similar weighted approach would likely be useful in parameter tuning for other robust approaches, such as the Huber forest of [LM].

Our analysis is based on the standard assumption that both training and test observations are identically distributed according to some unknown distribution. Training observations that come from a different distribution than those cases we seek to predict are problematic. In practice, it is possible that outliers might be indicative of unknown signal in data and thus should not necessarily be ignored. The ability of RF-LOWESS to identify training cases being down-weighted allows users to further investigate these cases and determine whether they might in fact provide relevant information.

RF-LOWESS is intended to address situations where contamination is present in the training data, but not in the test data. We assume that the response variable has been contaminated for a certain percentage of training cases, and that this contamination occurs completely at random. That is, there is no way to identify contaminated cases using information from the explanatory variables. A method that also handles test data contamination would be ideal, but this would require distinguishing test cases that are likely contaminated from those which are not. Because we are assuming that contamination cannot be detected based on information contained in the explanatory variables, no method would be able to do this. While the assumption that contamination occurs only in the training data might limit the scope of the method, this assumption is consistent with prior research in random forest robustness. [RL] explain, "we see the contamination as errors or anomalies and we really want to predict the clean data that we see as the true population. If the test data is also contaminated, this becomes a characteristic of the population itself and cannot really be considered a contamination" [24]. Nevertheless, when contamination is present in the test data, our method can be expected to improve predictions on the test cases that do not come from the contaminating distribution.

We have considered only the impact of modifying aggregation approaches on the robustness of random forests. Additional gains in predictive performance might be possible by using RF-LOWESS along with more robust splitting rules, (e.g. [3, 8, 18, 24]). Our preliminary investigations were consistent with [RL]'s finding that robust aggregation has a larger impact than robust splitting, and our paper therefore focuses on aggregation approaches. The impact of combining RF-LOWESS with robust splitting algorithms is an area for potential future research, though we expect that in most situations, the gains in predictive performance would be relatively small.

While the flexibility and historical popularity of the Tukey bi-weight function make it a strong choice for step (iii) in the RF-LOWESS algorithm, other functions could conceivably be used in its place. Future research might explore the impact of various robustness functions on RF-LOWESS prediction accuracy.

Because 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 explore ways to efficiently employ weighted cross-validation algorithms, such as Algorithm 2 to tune the Huber forest algorithm. 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.

Received 27 January 2020

REFERENCES

[1] Breiman, L. (2001). Random forests. *Machine Learning* **45** 5–32. MR3874153

[2] Breiman, L., Friedman, J., Stone, C. J. and Olshen, R. A. (1984). *Classification and regression trees.* CRC press. MR0726392

[3] 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.*

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

[5] Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* **74** 829–836. MR0556476

[6] De Cock, D. (2011). Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project. *Journal of Statistics Education* **19**.

[7] Dheeru, D. and Karra Taniskidou, E. (2019). UCI Machine Learning Repository.

[8] Galimberti, G., Pillati, M. and Soffritti, G. (2007). Robust regression trees based on M-estimators. *Statistica* **67** 173–190. MR2654613

[9] Harrison Jr, D. and Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management* **5** 81–102.

[10] Hosmer Jr., D. W., Lemeshow, S. and Sturdivant, R. X. (2013). *Applied logistic regression* **398**. John Wiley & Sons. MR3287463

[11] Hothorn, T., Hornik, K. and Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics* **15** 651–674. MR2291267

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

[13] Ishwaran, H. and Kogalur, U. B. (2007). Random survival forests for R. *R News* **7** 25–31.

[14] Ishwaran, H. and Kogalur, U. B. (2019). Fast Unified Random Forests for Survival, Regression, and Classification (RF-SRC) R package version 2.9.0.

[15] Li, A. H. and Martin, A. (2017). Forest-type Regression with General Losses and Robust Forest. In *International Conference on Machine Learning* 2091–2100.

[16] Liaw, A. and Wiener, M. (2002). Classification and Regression by randomForest. *R News* **2** 18–22.

[17] Lin, Y. and Jeon, Y. (2006). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association* **101** 578–590. MR2256176

[18] Loh, W.-Y. (2008). Regression by parts: Fitting visually interpretable models with GUIDE. In *Handbook of Data Visualization* 447–469. Springer.

[19] Meinshausen, N. (2006). Quantile regression forests. *The Journal of Machine Learning Research* **7** 983–999. MR2274394

[20] Meinshausen, N. (2017). quantregForest: Quantile Regression Forests R package version 1.3-7. MR2274394

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

[22] Redmond, M. and Baveja, A. (2002). A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research* **141** 660–678.

[23] Rousseeuw, P. J. and Leroy, A. M. (2005). *Robust regression and outlier detection* **589**. John Wiley & Sons. MR0914792

[24] Roy, M.-H. and Larocque, D. (2012). Robustness of random forests for regression. *Journal of Nonparametric Statistics* **24** 993–1006. MR2995488

[25] Sage, A. J. (2019). RFLOWESS R package version 1.0.

[26] R Core Team (2018). R: A Language and Environment for Statistical Computing R Foundation for Statistical Computing, Vienna, Austria.

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

[28] Yeh, I.-C. (2007). Modeling slump flow of concrete using second-order regressions and artificial neural networks. *Cement and Concrete Composites* **29** 474–480.

Andrew J. Sage
Department of Mathematics, Statistics, and Computer Science
Lawrence University, Appleton, WI
United States
E-mail address: andrew.j.sage@lawrence.edu

Ulrike Genschel
Department of Statistics
Iowa State University, Ames IA
United States
E-mail address: ulrike@iastate.edu

Dan Nettleton
Department of Statistics
Iowa State University, Ames IA
United States
E-mail address: dnett@iastate.edu