*Journal of Statistical Research* *https://doi.org/10.3329/jsr.v56i2.67466*
*2022, Vol. 56, No. 2, pp. 115-131* *ISSN 0256 - 422 X*

# DEBIAS RANDOM FOREST REGRESSION PREDICTORS

LIHUA CHEN*
*Department of Mathematics and Statistics, James Madison University, Harrisonburg, VA, USA*
*Email: chen3lx@jmu.edu*

PRABHASHI WITHANA GAMAGE
*Department of Mathematics and Statistics, James Madison University, Harrisonburg, VA, USA*
*Email: withanpw@jmu.edu*

JOHN RYAN
*Department of Economics, University of Wisconsin-Madison, Madison, WI, USA*
*Email: john.p.ryan@wisc.edu*

SUMMARY

The random forest can reduce the variance of regression predictors through bagging while leaving the bias mostly unchanged. In general, the bias is not negligible and consequently bias correction is necessary. The default bias correction method implemented in the R package `randomForest` often works poorly. Several approaches have been developed which in general outperform the R default. However, little work has been done to comprehensively evaluate the performance of these methods and thus guide users to select an appropriate method for bias correction. This paper fills this gap by providing an informative ranking of these bias correction methods based on an extensive numerical study. We further offered practical suggestions on the application of the winner of these methods and suggested a visualization technique to help users decide when bias correction is needed.

*Keywords and phrases:* Random forest, Regression predictor, Bias correction, Univariate smoothing, Boosted forest

*AMS Classification:* 62G08

## 1 Introduction

Random forest is one of the most popular and successful ensemble methods in machine learning used for classification and regression. In this paper we apply random forest to regression problems; i.e., to relate a quantitative response variable $Y$ to covariates or features $X$. A random forest averages predictions from many regression trees where each tree is built upon a bootstrap sample or a subsample of the original data with a randomized tree building scheme. The accuracy of random forests has been shown to be competitive among many supervised machine learning methods

* Corresponding author

(Breiman, 2001b; Uddin et al., 2019; Biau et al., 2008). Intuitively, random forests attain good accuracy through averaging many large trees. A large tree tends to have low bias but high variance. Averaging can reduce the variance, so random forests have potential for achieving both low bias and variance simultaneously.

However, very often the bias of a random forest is non-negligible even if it is built on large trees. This is because a random forest is essentially a $k$-nearest neighbor method where the estimation at the target point is the average of the response values of its nearest neighbors under a certain distance metric (Lin and Jeon, 2006). Therefore those extreme regression function values tend to be estimated with a negative bias near the upper extreme and positive bias near the lower extreme. Wager and Athey (2018) provided a bound on the magnitude of the bias and Biau et al. (2008) showed the decreasing rate of the bias under certain assumptions about the tree-growing mechanism and the data distribution. But in practical settings with limited sample sizes, the bias often remains an important component of the statistical risk and consequently, bias correction is necessary.

Multiple random forest bias correction approaches have been developed and they can be classified into two major categories. The first category uses a univariate smoothing approach. This method computes the corrected prediction directly through fitting a univariate function between the observed response $y$ and the random forest prediction $\hat{y}$ on the training data and then projecting the same relationship to the corrected and uncorrected prediction of the target point. The default correction method in the R package `randomForest` (Liaw and Matthew, 2002) fits a linear function between $y$ and $\hat{y}$. The linear model does not perform well when the true relationship between them is nonlinear. Zhang and Lu (2012) experimented with fitting a more flexible smooth spline model. A distinct feature of the univariate smoothing method is that it only utilizes the response variable without explicitly using the covariates information.

The second category models the bias explicitly. It is similar in spirit to the 'boosting' idea (Freund and Schapire, 1996; Friedman, 2001) as it tries to improve the estimators through correcting the errors from the previous model. Breiman (2001a) first proposed using a random forest to model the bias and then subtracting the estimated bias from the prediction. This approach was also investigated by Zhang and Lu (2012), Xu (2013) and Ghosal and Hooker (2021) respectively. Ghosal and Hooker (2021) named this method 'boosted forest' and we will refer to approaches based on this general idea as boosted forest in this paper. Lu and Hardin (2021) proposed a conditional bias approach which weighted the training data points by their closeness to the target data point. Hooker and Mentch (2018) used a traditional residual bootstrap technique for bias correction developed by Efron and Tibshirani (1993) where the bias could be assessed as the difference between an estimator and the average of the bootstrapped estimators.

The aforementioned bias correction methods have been partially compared. Zhang and Lu (2012) compared one variant of the univariate smoothing method and one-step boosted forest and concluded they performed similarly well. Ghosal and Hooker (2021) compared boosted forest to the residual bootstrap method and concluded boosted forest performed better in general. Lu and Hardin (2021) compared their conditional bias method to boosted forest and concluded its performance was comparable to that of boosted forest.

In summary, these partial comparisons have identified univariate smoothing, boosted forest and

the conditional bias method as competitive candidates for bias correction. It is more desirable to offer a more informative ranking of these methods for the benefit of the user. Therefore in this paper, we further evaluate and compare these competitive methods through neutral and comprehensive comparisons. In evaluating the univariate smoothing method, we also added quadratic and cubic functions to examine if a higher-degree polynomial model improved over the linear model. In evaluating the boosted forest method, we also implemented the bias correction iteratively and studied its performance with different iteration cycles. Through extensive numerical studies, we were able to identify boosted forest as a general winner in many settings. This finding is valuable to users who need to choose among different methods for bias correction. We further explored an easy-to-use stopping rule to help determine the optimal number of iterations in iterative boosted forest. In addition, we suggested a visualization technique which compares the predicted versus the observed response with the $45^o$ reference line to help users decide when bias correction is needed.

In Section 2, we will introduce the bias correction methods we will evaluate, including two versions of the univariate smoothing methods we have proposed. In Section 3 and 4, we will compare these methods through simulations and real data examples. We conclude the paper in Section 5 with a focus on offering user guidance on how to identify the need for bias correction and how to implement bias correction when the need arises.

# 2 Methods

## 2.1 Bias correction is necessary

In the regression setting, a common goal is to use the estimated regression function to predict a future response. The average mean-squared prediction error is the sum of the noise variance, the predictor bias and predictor variance or MSPE=error variance + (squared) predictor bias + predictor variance. We usually have no control over the inherent error variance. Random forest can reduce the variance of the predictor through averaging but has little effect on the bias, and thus the bias often remains an important component in the prediction error and needs to be corrected.

First we will demonstrate the existence of bias through a simulation study. We simulate $n = 500$ data points with covariates $x_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4})$ where each independent $x_{ip} \sim \text{Uniform}(0, 1)$ for $i = 1, \ldots, n; p = 1, \ldots, 4$. Generate $y_i = f(x_i) + \epsilon_i$ where $f(x) = 3x_1 + {x_2}^2 + \sin(x_3) - \sqrt{x_4}$ and $\epsilon_i \sim N(0, \sigma^2)$. We choose $\sigma = 0.5$. We will compare the out-of-bag (OOB) prediction $\hat{y}_i$ produced by the R package `randomForest` to the true regression function value $f(x_i)$.

OOB predictions are produced through bootstrapping samples in building a random forest. For different variants of random forests, each tree is built on a bootstrap or subsample of the original data with some random tree-building mechanism. In this paper we adopt the most popular variant developed by Breiman (2001b). It uses a bootstrap sample of the training data and random selection of features at each split to build each tree. Suppose we have a training set $T = (T_1, \ldots, T_n)$ of size $n$. A bootstrap sample is generated by sampling $n$ cases independently from $T$ with replacement. In a bootstrap sample, we call those left-out observations out-of-bag (OOB) observations. On average, a training observation is OOB in about $37\%$ of the trees in a random forest. The OOB prediction of

a training case $T_i$ is the average of predictions produced by those trees that do not contain this case. Let $B(T_i)$ denote the set of trees that do not contain $T_i$ and $\hat{y}_i^j$ denote the predicted response of $T_i$ by the $j$th tree. Then the OOB prediction of $T_i$ is

$$\hat{y}_i = \frac{\sum_{j \in B(T_i)} \hat{y}_i^j}{\mid B(T_i) \mid},$$

where $\mid B(T_i) \mid$ is the number of elements in $B(T_i)$. OOB predictions are made on the training cases but they act like predictions made on the test data. They are integral to many random forest bias correction methods. The R package `randomForest` produces OOB predictions of each observation in the training data.

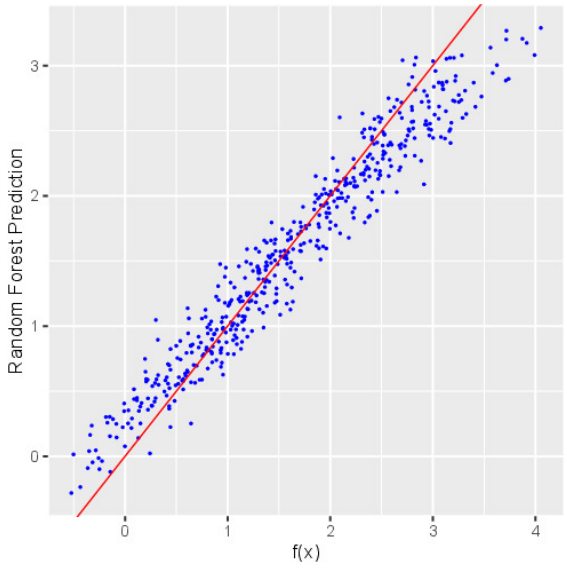


Figure 1: Comparisons of the OOB predictions to the true function values.

In Figure 1, we plot the OOB prediction versus the true function value $f(x)$. The points above (below) the $45^o$ line indicate the true function values are estimated with a positive (negative) bias. It is obvious that the high values of the true function are under-estimated and the low values are over-estimated. We experimented with various true functions, sample sizes and error variances and observed a similar pattern. We observed for a fixed error variance, the bias tended to be smaller with a bigger sample size; but the magnitude of the bias did not exhibit a clear positive relationship with the size of the error variance.

The bias issue is common with real data sets as well. We illustrate the presence of bias with the Boston housing data which is arguably the most commonly used benchmark data in machine learning. We randomly chose 1/3 of the data as test data. As shown in Figure 2, random forest performed well for most of the test data, with significant bias presented for a few data points at the

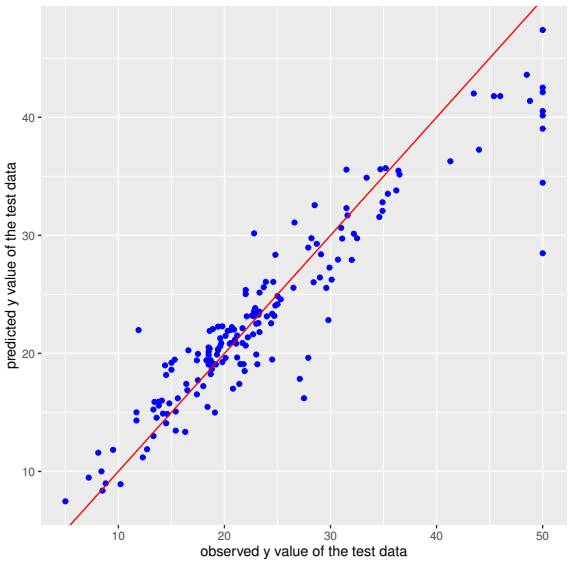


Figure 2: Comparisons of the random forest predictions to observed test data for the Boston housing data.

upper right corner. Although random forest prediction often works well, it could be further improved with bias correction.

## 2.2 Bias correction methods

In this section, we will introduce the methods we implement and compare in the numerical studies. We use the uncorrected results as a baseline for comparison.

### 2.2.1 Univariate smoothing methods

The univariate smoothing methods we explore include the default correction method in `randomForest` and a few variants. The default correction method estimates the corrected response directly as below:

- Let $b_0$ and $b_1$ denote the estimated intercept and slope, respectively, obtained by fitting a simple linear regression of the observed training response $y_i$ on its out-of-bag prediction $\hat{y}_i$:

$$y_i = b_0 + b_1 \hat{y}_i, i = 1, \dots n.$$

- For a test case with covariates $x_0$, let $\hat{y}_0$ be the predicted response from the random forest. Then the bias-corrected prediction is $\hat{y}_0^C = b_0 + b_1 \hat{y}_0$.

This method assumes a linear relationship between $y_i$ and $\hat{y}_i$ and assumes the same linear relationship can be extended to the test case. To model the relationship more flexibly, we also fit a

quadratic, cubic, and smooth spline function between $y_i$ and $\hat{y}_i$ and project the estimated relationship to the test case. We used *smooth.spline* function in the R package `splines` for fitting the spline function. This univariate smoothing method does not use the features directly. Xu (2013) showed that the bias of random forest predictions could vary non-linearly across different feature regions. Therefore univariate smoothing may be insufficient and additionally it may be necessary to incorporate covariates information in bias correction.

### 2.2.2 Boosted forest

This method was first proposed by Breiman (2001a). It uses a random forest to model the bias explicitly and then subtracts the estimated bias $\hat{b}$ from the original predictor $\hat{y}$ to form the bias-corrected predictor $\hat{y}^C$: $\hat{y}^C = \hat{y} - \hat{b}$.

The bias is estimated from $b_i = \hat{y}_i - y_i$, which is just the negative of the residual $r_i = y_i - \hat{y}_i, i = 1, \ldots n$. Note the residual used here is the difference between the observed responses $y_i$ and its OOB prediction $\hat{y}_i$. Ordinary residuals on a training set from regressions do not work as they have a tendency to be too small. Breiman (2001a) argued these OOB residuals represent the 'true residuals' and provide an unbiased estimate of the bias term. The method follows the steps as below:

- A new random forest built on training data $(r_i, x_i)_{i=1}^n$ or $(b_i, x_i)_{i=1}^n$ is used to estimate $r_i$ (or $b_i$).

- The OOB predictions of the training data are bias-corrected as

$$\hat{y}_i^C = \hat{y}_i - \hat{b}_i \quad \text{or} \quad \hat{y}_i^C = \hat{y}_i + \hat{r}_i, i = 1, \ldots n.$$

- For a test case $T_0$ with features $x_0$, obtain its prediction $\hat{y}_0$ and $\hat{r}_0$ (or $\hat{b}_0$) from the original and the new random forest respectively and update its corrected prediction as

$$\hat{y}_0^C = \hat{y}_0 - \hat{b}_0 = \hat{y}_0 + \hat{r}_0$$

This process can be repeated iteratively; i.e., obtain new residuals from the bias-corrected predictors, use another random forest to estimate the new residuals and then add the estimated new residuals to the bias-corrected predictor to further update the predictor, and so on. For the random forests built in different stages, the input values $x$ are kept the same while the response values are the successively updated residuals.

We change the notations slightly to summarize the iterations:

- Let $y_i^{(j)}$ denote the response value of case $T_i$ in the $j$th random forest, with $y_i^{(1)}$ being the initial response value.

- At the $j$th stage, let $\hat{y}_i^{(j)}$ be the OOB prediction of case $T_i$ produced by the $j$th random forest and set $y_i^{(j+1)} = y_i^{(j)} - \hat{y}_i^{(j)}$. Note this formula produces the successive residuals.

- For a test case $T_0$ with features $x_0$, obtain the prediction $\hat{y}_0^{(j)}$ from the $j$th random forest, $j = 1, 2, \ldots, J$ and get the corrected prediction of case $T_0$ as

$$\hat{y}_0^C = \sum_{j=1}^{J} \hat{y}_0^{(j)}.$$

Boosted forest is in spirit similar to the 'gradient boosting' idea with the difference that the base learner is a random forest rather than a single tree. Another major difference is that boosted forest does not use a tuning parameter which shrinks the contribution of each base learner to control overfitting. Therefore it is important to stop the iteration at a proper stage to avoid overfitting. In working with real data, we suggest monitoring the average MSPE on the test data over sufficient rounds of data splits and stopping iteration when the average MSPE starts to increase. Our numerical studies in Section 4 showed that this simple-to-use stopping rule did not lead to insufficient number of iterations. Our studies also showed two iterations were sufficient in most settings.

### 2.2.3 Conditional bias correction

This method was developed by Lu and Hardin (2021) which also uses a random forest to model the bias term explicitly. It estimates the bias of a test case $T_0$ with features $x_0$ as a weighted average of the OOB biases $\hat{y}_i - y_i$ of the training observations:

$$\hat{b}_0(x_0) = \sum_{i=1}^{n} v_i(x_0)(\hat{y}_i - y_i).$$

The weight $v_i(x_0)$ assigned to a training case $T_i = (x_i, y_i)$ depends on how close it is to the test case $T_0$. In fact, $v_i(x_0)$ is proportional to a quantity which measures how many times the training observation $T_i$ is OOB and would fall in the same terminal nodes as $T_0$ among all the trees in the random forest. Specifically, the weight is computed as

$$v_i(x_0) = \frac{\sum_{j=1}^{J} I(T_i \notin D_{n,j}^* \text{and} x_i \in R_{s(x_0,\theta_j)})}{\sum_{l=1}^{n}\sum_{j=1}^{J} I(T_l \notin D_{n,j}^* \text{and} x_l \in R_{s(x_0,\theta_j)})},$$

where $I$ is the indicator function, $D_{n,j}^*$ denotes the bootstrap sample in tree $j$, $s(x_0, \theta_j)$ denotes the terminal node containing $x_0$ in tree $j$ and $R_{s(x_0,\theta_j)}$ denotes the corresponding subspace. That is, a training case closer to the target case will receive a higher weight.

Let $\hat{y}_0$ be the prediction of the test case $T_0$ and bias-corrected prediction at $x_0$ is obtained as:

$$\hat{y}_0^C = \hat{y}_0 - \hat{b}_0(x_0).$$

Recall a random forest is essentially a $k$-nearest neighbor method. In the boosted forest approach, when a random forest is used to estimate the biases, it also assigns different weights to the OOB biases of the training cases. Therefore this conditional bias method is essentially similar to one-step boosted forest.

# 3 Simulation Analysis

In this section, we investigate the finite sample performance of the aforementioned bias-corrected methods through in-depth simulation studies. With synthetic data we can compute the bias as the true regression function is known. All computations were performed in R. The conditional bias method was implemented using the R package `forestError` (Lu and Hardin, 2021).

## 3.1 Study I

In the first study, we generated data based on a benchmark function in machine learning studies, given in Friedman (1991):

$$\text{Function 1: } f(\mathbf{x}) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5,$$

where $x_i \sim \text{Uniform}(0,1); i = 1, 2, \ldots, 5$. The response was then generated from the normal distribution; i.e., $y_i \sim \text{N}(f(x_i), \sigma^2)$ with $\sigma \in \{1, 2, 4\}$.

We computed the MSPE and the bias on the test data as comparison metrics. To compute MSPE, at each replication, we randomly generated a training data set of size $n$ and a test data set of size $n_t$. We fit the training data by each method and used the fitted model to obtain predicted response of the test data $\hat{y}_i$. The MSPE on the test data was computed as

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

We replicated this process 1000 times and reported the mean and the standard deviation of these 1000 MSPEs.

To compute the bias, we randomly generated the test data set of size $n_t$ and held it fixed. At each replication $k$, we generated a training data set of size $n$. We obtained the predicted response on the test data; i.e., $\widehat{y_k} = (\hat{y}_1, \ldots, \hat{y}_{n_t})_k$ by each method. Then we replicated the process 1000 times and averaged the 1000 prediction vectors $\widehat{y_k}$'s to obtain the averaged predictions $\widetilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_{n_t})$. The squared bias was obtained as

$$\text{bias}^2 = \frac{1}{n_t}\sum_{i=1}^{n_t}\Big(\tilde{y_i} - f(x_i)\Big)^2.$$

Table 1 provides a summary of the simulation results. The comparison was conducted across the methods including uncorrected (Un), default (De), quadratic (Qu), cubic (Cu) and spline smoothing(Sp), Breiman's boosted forest (Bf) with additional $k$ iterations (Bf (k)), and conditional bias (Cond). At lower noise levels, i.e., $\sigma = 1$ or 2, when the noise variance did not dominate the MSPE, the bias correction methods were effective in reducing the bias with boosted forest achieved the smallest MSPE. The change in MSPE and bias showed the reduction in MSPE was mainly due to the reduction in bias. However, at a high noise level, i.e., $\sigma = 4$, the noise variance $\sigma^2 = 16$ dominated the MSPE (21.90 for uncorrected). Though the bias correction methods could still significantly reduce the bias, they also significantly increased the variance, leading to an overall negligible improvement in MSPE. This demonstrated the debiasing methods were not effective for very noisy

Table 1: Comparison results for Function 1 with training data size $n = 200$ or $2000$ , test data size $n_t = 2000$ and number of replications $K = 1000$. This summary includes the average mean square prediction error (MSPE), the standard deviation of 1000 MSPEs (SD), and the squared bias (bias$^2$). The minimum MSPE is in boldface.

| | | Un | De | Qu | Cu | Sp | Bf | Bf(1) | Bf(2) | Bf(3) | Cond |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma = 1$ | MSPE | 6.44 | 3.92 | 3.79 | 3.81 | 3.83 | 3.39 | **2.95** | 3.13 | 3.13 | 3.92 |
| n=200 | (SD) | (0.42) | (0.41) | (0.34) | (0.34) | (0.46) | (0.29) | (0.23) | (0.26) | (0.33) | (0.34) |
| | bias$^2$ | 4.76 | 1.44 | 1.43 | 1.42 | 1.43 | 1.46 | 1.46 | 0.74 | 0.53 | 2.01 |
| $\sigma = 2$ | MSPE | 9.42 | 7.11 | 7.15 | 7.19 | 7.20 | **6.78** | 6.78 | 6.87 | 7.71 | 7.20 |
| n=200 | (SD) | (0.71) | (0.5) | (0.51) | (0.51) | (0.57) | (0.46) | (0.46) | (0.44) | (0.54) | (0.52) |
| | bias$^2$ | 4.36 | 1.53 | 1.52 | 1.53 | 1.52 | 1.38 | 1.39 | 0.77 | 0.57 | 1.96 |
| $\sigma = 4$ | MSPE | 21.90 | **20.58** | 20.67 | 20.82 | 20.81 | 20.85 | 20.85 | 22.94 | 26.45 | 20.58 |
| n=200 | (SD) | (1.03) | (0.91) | (0.93) | (0.98) | (1.09) | (0.90) | (0.90) | (1.14) | (1.63) | (0.90) |
| | bias$^2$ | 4.39 | 1.75 | 1.73 | 1.79 | 1.75 | 1.22 | 1.22 | 0.64 | 0.45 | 1.90 |
| $\sigma = 1$ | MSPE | 3.22 | 2.27 | 2.28 | 2.28 | 2.27 | **1.78** | 1.78 | 1.60 | 1.63 | 2.01 |
| n=2000 | (SD) | (0.23) | (0.19) | (0.18) | (0.18) | (0.18) | (0.13) | (0.13) | (0.12) | (0.11) | (0.16) |
| | bias$^2$ | 1.80 | 0.88 | 0.87 | 0.88 | 0.88 | 0.35 | 0.35 | 0.12 | 0.05 | 0.64 |

data. Finally, we observed the increased training sample size ($n = 2000$ at $\sigma = 1$) led to a smaller bias and MSPE for each method.

## 3.2 Study II

The second simulation study focused on one more benchmark function (Function 2) in machine learning (Friedman, 1991) and three other regression functions of our choice (Function 3 to 5). We choose one linear, one polynomial and one non-linear function to cover a wide variety of functions.

Function 2: $f(\mathbf{x}) = 0.1\exp(4x_1) + 4/[1+\exp(-20x_2+10)] + 3x_2 + 2x_4 + x_5 + 0\sum_{i=6}^{10} x_i,$

Function 3: $f(\mathbf{x}) = 10(0.878x_1 + 0.642x_2 + 0.766x_3 + 0.666x_4 + 0.933x_5) + 0\sum_{i=6}^{10} x_i,$

Function 4: $f(\mathbf{x}) = 10x_1 + 0.5x_2^2 + 15x_2 - 25\sqrt{x_3} + 0.05x_4 + 0x_5,$

Function 5: $f(\mathbf{x}) = 10\cos(\pi x_1 x_3) + 10x_3^2 + 10(x_1 + x_4).$

The data were generated with the same generating process described in simulation study I with $\sigma = 1$. The coefficients in Function 3 were generated independently from uniform(0.5, 1). Note in Functions 2, 3 and 4, we included both signal variables and noise variables with 0 coefficients.

Table 2 summarizes the results. The results showed all the bias correction methods outperformed the uncorrected random forest with the best result produced by boosted forest at no iteration for Function 2 and at 2 iterations for the other functions. The results also confirmed the reduction in MSPE was mainly achieved through reduction in bias.

Our simulation results with multiple other functions also identified boosted forest as a general winner.

Table 2: Comparison results for Function 2 to 5 with training data size $n = 200$, test data size $n_t = 2000$, and the number of replications $K = 1000$. This summary includes the average mean square prediction error (MSPE), the standard deviation of 1000 MSPEs (SD), and the squared bias (bias$^2$). The minimum MSPE is in boldface.

| Function | | Un | De | Qu | Cu | Sp | Bf | Bf(1) | Bf(2) | Bf(3) | Cond |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | MSPE | 2.01 | 1.50 | 1.49 | 1.50 | 1.52 | **1.41** | 1.41 | 1.47 | 1.65 | 1.49 |
| 2 | (SD) | (0.21) | (0.12) | (0.12) | (0.15) | (0.12) | (0.12) | (0.12) | (0.11) | (0.14) | (0.13) |
| | bias$^2$ | 0.79 | 0.15 | 0.14 | 0.13 | 0.14 | 0.11 | 0.11 | 0.02 | 0.01 | 0.21 |
| | MSPE | 8.68 | 4.12 | 4.15 | 4.18 | 4.19 | 3.49 | 3.49 | **2.40** | 2.46 | 5.04 |
| 3 | (SD) | (0.78) | (0.58) | (0.59) | (0.59) | (0.61) | (0.40) | (0.41) | (0.21) | (0.17) | (0.61) |
| | bias$^2$ | 6.50 | 0.78 | 0.76 | 0.74 | 0.75 | 1.34 | 1.34 | 0.18 | 0.06 | 2.56 |
| | MSPE | 8.83 | 4.15 | 4.12 | 4.14 | 4.17 | 3.06 | 3.07 | **2.59** | 2.96 | 3.76 |
| 4 | (SD) | (1.18) | (0.67) | (0.67) | (0.68) | (0.73) | (0.46) | (0.46) | (0.27) | (0.29) | (0.65) |
| | bias$^2$ | 7.51 | 1.30 | 1.25 | 1.22 | 1.23 | 1.01 | 1.00 | 0.22 | 0.21 | 1.80 |
| | MSE | 2.79 | 2.20 | 2.20 | 2.15 | 2.12 | 1.68 | 1.68 | **1.61** | 1.80 | 1.67 |
| 5 | (SD) | (0.18) | (0.19) | (0.19) | (0.18) | (0.19) | (0.10) | (0.10) | (0.07) | (0.08) | (0.11) |
| | bias$^2$ | 1.45 | 0.77 | 0.76 | 0.67 | 0.67 | 0.29 | 0.30 | 0.08 | 0.04 | 0.37 |

# 4 Data Application

We applied the bias-correction methods to the benchmark data sets in machine learning studies available at the UCI Machine Learning Repository. Information about these data sets is available in Table 3.

At each replication, we randomly selected $1/3$ of the data as test data, and computed the MSPE on the test data by each method. The average MSPE from 1000 replications are reported in Table 4 with the standard deviation given in parentheses.

The results showed that boosted forest distinctly outperformed the other bias correction methods except for a few noisy data sets: abalone, wine-red, wine-white and skillcraft 1 data. As an illustration, Figure 3 compared the uncorrected results to the optimal univariate smoothing method and the optimal boosted forest method for the yacht hydrodynamics and concrete strength data set, showing

Table 3: Data set summary

| Number | Data set | No. of Observations | No. of Predictors |
|---|---|---|---|
| 1 | Auto MPG | 398 | 7 |
| 2 | Concrete Compressive Strength | 1030 | 8 |
| 3 | Abalone | 4177 | 8 |
| 4 | Winequality-red | 1599 | 12 |
| 5 | Winequality-white | 4898 | 12 |
| 6 | Servo | 167 | 5 |
| 7 | Yacht Hydrodynamics | 308 | 6 |
| 8 | Forest Fires | 517 | 12 |
| 9 | Skillcraft1 | 3395 | 18 |
| 10 | Boston | 506 | 13 |
| 11 | Computer Hardware | 209 | 7 |
| 12 | Energy Efficiency(Heating) | 768 | 8 |
| 13 | Energy Efficiency(Cooling) | 768 | 8 |

an obvious advantage of boosted forest. We found that with iterative boosted forest, typically two iterations were sufficient. This finding was consistent with the results in the simulation study.

In Section 2.2.2, we suggest an easy-to-use rule to choose the number of iterations for boosted forest: stop iterations when the average MSPE starts to increase. Will this rule lead to too early stopping? To answer this question, we tracked the average MSPE for up to 6 iterations for all the data sets used in this paper. The universal U shaped curves of test error versus the number of iterations as shown in Figure 5 indicate our rule does not risk insufficient number of iterations in general.

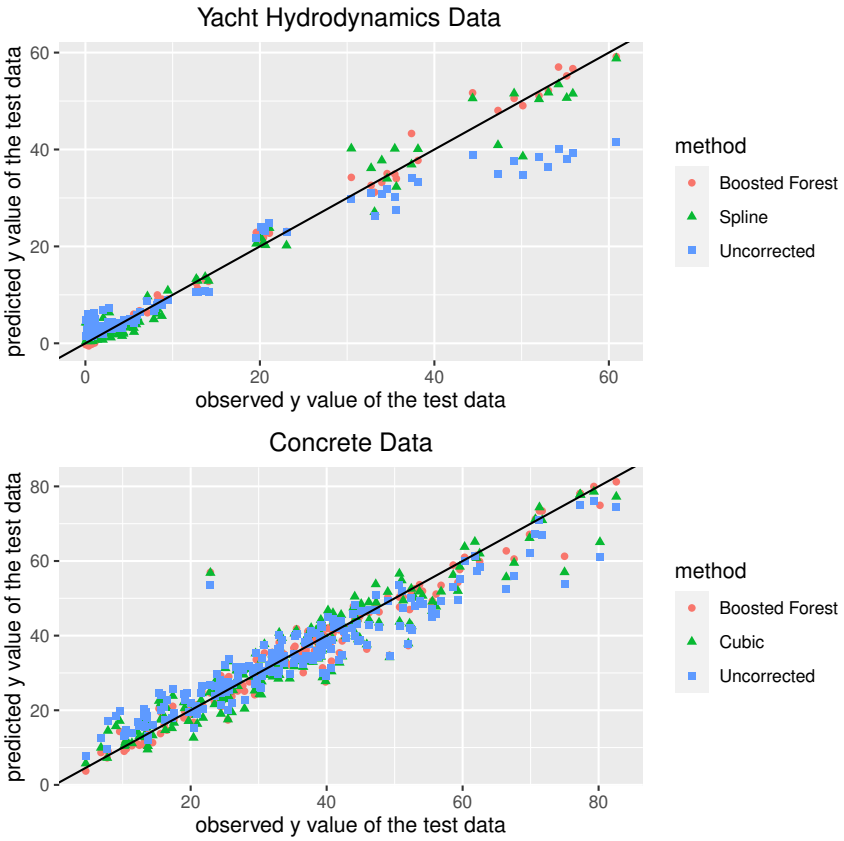


Figure 3: Comparison of the uncorrected results to the best performance of univariate smoothing and boost forest for the yacht hydrodynamics and concrete strength data set.


Figure 4: Comparison of the uncorrected results to the best performance of univariate smoothing and boost forest for the abalone data set.

We found bias correction was not effective for very noisy data including the abalone, wine and skillcraft1 data. Checking the correlations between the response and the feature variables in these data sets revealed weak correlations for all features. This implied it was challenging to obtain reliable predictions using the given features for any method. Figure 4 demonstrates the difficulty of bias corrections with the noisy abalone data set.

Both our simulation and real data results demonstrated boosted forest outperformed univariate smoothing in general. We conjecture the following factors contribute to the advantage of boosted forest:

1. Univariate smoothing only uses the response variable without explicitly using the covariates information. When the bias varies across different regions of the covariates, it is beneficial to also incorporate the covariates to estimate the bias as boosted forest does.

2. It is more stable to use a random forest to estimate the bias compared to using a univariate regression function.

3. Boosted forest can further reduce the bias with more iteration cycles when the need arises. Breiman (2001a) has heuristically shown that under the assumption that the correlation between the true function and its estimator is bigger than $0.5(1+\epsilon)$ for $\epsilon > 0$, the bias after $k$ iterations will decrease to zero as $k \to \infty$.

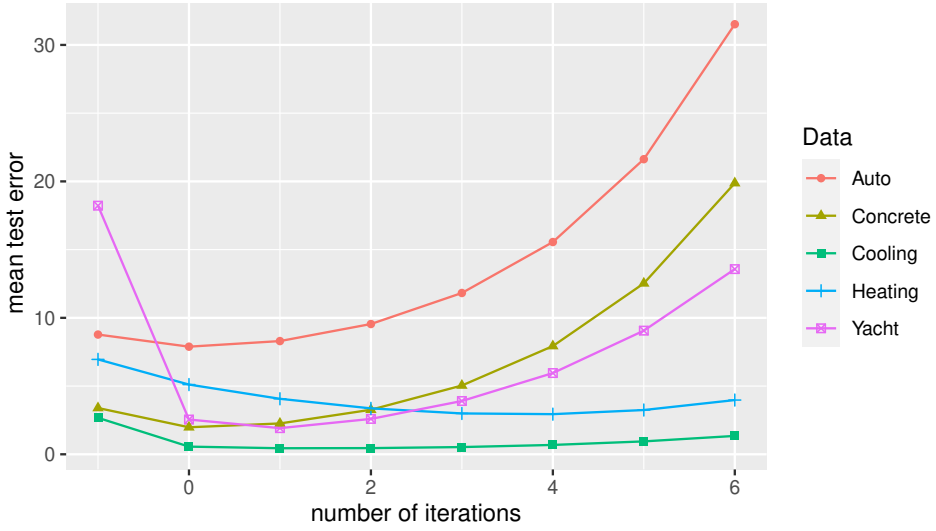


Figure 5: Test error as a function of the number of iterations in boosted forest. The starting point in each curve indicates the test error for uncorrected random forest. Test errors for some data sets are rescaled for visual comparison.

Table 4: Comparison results on real data sets. This summary includes the average mean square prediction error (MSPE) and the standard deviation of 1000 MSPEs (SD). The mininum MSPE is in boldface. The symbol '-' indicates no further iteration is needed.

| | | Un | De | Qu | Cu | Sp | Bf | Bf (1) | Bf (2) | Bf (3) | Cond |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Auto | MSPE | 8.10 | 8.02 | 8.00 | 8.02 | 8.27 | **7.41** | 7.41 | 9.65 | -- | 7.65 |
| | (SD) | (1.62) | (1.59) | (1.61) | (1.62) | (1.83) | (1.46) | (1.59) | (1.93) | -- | (1.47) |
| Concrete | MSPE | 33.16 | 27.74 | 27.8 | 27.37 | 27.53 | **19.52** | 22.19 | 31.81 | -- | 23.22 |
| | (SD) | (3.80) | (3.55) | (3.57) | (3.59) | (3.61) | (3.16) | (3.52) | (4.78) | -- | (3.19) |
| Abalone | MSPE | 4.61 | **4.60** | 4.60 | 4.60 | 4.61 | 4.85 | 5.33 | 6.01 | -- | 4.67 |
| | (SD) | (0.23) | (0.23) | (0.23) | (0.23) | (0.24) | (0.23) | (0.25) | (0.27) | -- | (0.23) |
| Wine-Red | MSPE | 0.35 | 0.35 | **0.33** | 0.33 | 0.35 | 0.35 | 0.41 | 0.54 | -- | 0.34 |
| | (SD) | (0.02) | (0.02) | (0.02) | (0.02) | (0.02) | (0.02) | (0.03) | (0.03) | -- | (0.02) |
| Wine-White | MSPE | 0.39 | 0.38 | 0.38 | 0.38 | 0.38 | **0.37** | 0.43 | 0.57 | -- | 0.37 |
| | (SD) | (0.02) | (0.02) | (0.02) | (0.02) | (0.02) | (0.02) | (0.02) | (0.02) | -- | (0.02) |
| Servo | MSPE | 0.75 | 0.54 | 0.45 | 0.44 | 0.67 | 0.44 | 0.35 | **0.34** | 0.35 | 0.52 |
| | (SD) | (0.28) | (0.20) | (0.25) | (0.23) | (0.65) | (0.22) | (0.21) | (0.20) | (0.20) | (0.19) |
| Yacht | MSPE | 12.89 | 5.75 | 5.36 | 4.52 | 4.08 | 1.69 | **1.55** | 2.19 | -- | 3.46 |
| | (SD) | (4.65) | (1.58) | (1.45) | (1.63) | (1.62) | (0.99) | (0.69) | (0.81) | -- | (2.24) |
| Fires | MSPE | 4426 | 4189 | 4215 | 4257 | 4653 | **4147** | 5502 | -- | -- | 4151 |
| | (SD) | (3499) | (3658) | (3638) | (3616) | (3404) | (3517) | (3153) | -- | -- | 3453 |
| Skills | MSPE | **0.94** | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 1.05 | 1.10 | -- | 1.11 |
| | (SD) | (0.04) | (0.04) | (0.04) | (0.04) | (0.04) | (0.04) | (0.04) | (0.04) | -- | (0.03) |
| Boston | MSPE | 11.81 | 11.12 | 11.05 | 11.09 | 11.38 | **9.75** | 10.58 | 12.57 | 16.26 | 9.99 |
| | (SD) | (3.57) | (3.51) | (3.56) | (3.58) | (3.67) | (2.79) | (3.18) | (4.22) | (6.20) | (3.09) |
| Hardware | MSPE | 3379 | 2692 | 3988 | 18735 | 19992 | **1740** | 2027 | -- | -- | 2406 |
| | (SD) | (3531) | (3054) | (5695) | (9085) | (12867) | (2463) | (2261) | -- | -- | (3112) |
| Heating | MSPE | 1.22 | 1.22 | 1.20 | 1.10 | 0.67 | 0.27 | **0.22** | 0.24 | 0.31 | 0.68 |
| | (SD) | (0.30) | (0.30) | (0.29) | (0.29) | (0.25) | (0.10) | (0.09) | (0.10) | (0.18) | (0.19) |
| Cooling | MSPE | 3.48 | 3.46 | 3.46 | 3.42 | 3.78 | 2.51 | 2.00 | 1.65 | **1.46** | 3.14 |
| | (SD) | (0.36) | (0.36) | (0.36) | (0.36) | (0.63) | (0.34) | (0.36) | (0.36) | (0.37) | (0.34) |

# 5 Conclusions

The research on debiasing random forest predictors has not been very active. The purpose of this paper is to aid users in properly selecting and implementing a random forest bias correction method when the need arises. To this end, we conducted a comprehensive comparison of the bias-correction methods that were identified as winners in previous studies. These methods essentially fall into two categories: univariate smoothing and boosted forest. Univariate smoothing computes the corrected prediction directly by fitting a linear or a more flexible function between the observed and predicted response. Boosted forest uses a random forest to estimate the bias first and then subtracts the estimated bias from the predictor to form the corrected predictor. Though some previous studies showed these two types of methods had comparable performance, we conjectured boosted forest would perform better due to the factors we listed in Section 4. Our empirical studies produced results consistent with our conjecture. Being able to identify a general winner is valuable for practical purposes.

With boosted forest as the identified winner, we further explored its performance with iterations as this method was not widely studied before. We suggest a simple stopping rule to determine the optimal number of iterations. Our numerical studies showed that this easy-to-use rule did not lead to an insufficient number of iterations. Except for one data set, our studies showed two iterations were sufficient in general.

As a user guide we also suggest using visualizations to determine if bias correction is needed. A plot of the OOB predictions or the predicted test data versus the observed response with the $45^o$ reference line serves well to indicate whether serious bias exists. Plotting the bias-corrected response versus the observed response also helps examine whether bias correction is effective.

# Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments and suggestions to improve the paper. This work is supported by NSF grant (1950370).

# References

Biau, G., Devroye, L., and Lugosi, G. (2008), "Consistency of random forests and other averaging classifiers," *Journal of Machine Learning Research*, 2015–2033.

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

— (2001b), "Using iterated bagging to debias regressions," *Machine Learning*, 45, 261–277.

Efron, B. and Tibshirani, R. (1993), *An Introduction to the Bootstrap*, Chapman and Hall.

Freund, Y. and Schapire, R. E. (1996), "Experiments with a new boosting algorithm," *Machine Learning: Proceedings of the Thirteenth International Conference*, 148–156.

Friedman, J. H. (1991), "Multivariate adaptive regression splines," *Annals of Statistics*, 19 (1), 1–67.

— (2001), "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, 29, 1189–1232.

Ghosal, I. and Hooker, G. (2021), "Boosting random forests to reduce bias; One-step boosted forest and its variance estimate," *Journal of Computational and Graphical Statistics*, 30(2), 493–502.

Hooker, G. and Mentch, L. (2018), "Improvements to random forest methodology," *Statistics and Computing*, 28, 77–86.

Liaw, A. and Matthew, W. (2002), "Classification and regression by randomforest," *R News*, 2(3), 18–22.

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

Lu, B. and Hardin, J. (2021), "A unified framework for random forest prediction error estimation," *Journal of Machine Learning Research*, 22, 1–41.

Uddin, S., Khan, A., and Hossain, M. E. H. (2019), "Comparing different supervised machine learning algorithms for disease prediction," *BMC Medical Informatics and Decision Making*, 19, 281.

Wager, S. and Athey, S. (2018), "Estimation and inference of heterogeneous treatment effects using random forests," *Journal of the American Statistical Association*, 113, 1228–1242.

Xu, R. (2013), *Improvements to random forest methodology*, Dissertation, Iowa State University.

Zhang, G. and Lu, Y. (2012), "Bias-corrected random forests in regression," *Journal of Applied Statistics*, 39, 151–160.

*Received: October 19, 2022*

*Accepted: March10, 2023*