RFpredInterval: An R Package for Prediction Intervals with Random Forests and Boosted Forests

by Cansu Alakuş, Denis Larocque and Aurélie Labbe

Abstract Like many predictive models, random forests provide point predictions for new observations. Besides the point prediction, it is important to quantify the uncertainty in the prediction. Prediction intervals provide information about the reliability of the point predictions. We have developed a comprehensive R package, **RFpredInterval**, that integrates 16 methods to build prediction intervals with random forests and boosted forests. The set of methods implemented in the package includes a new method to build prediction intervals with boosted forests (PIBF) and 15 method variations to produce prediction intervals with random forests, as proposed by Roy and Larocque (2020). We perform an extensive simulation study and apply real data analyses to compare the performance of the proposed method to ten existing methods for building prediction intervals with random forests. The results show that the proposed method is very competitive and, globally, outperforms competing methods.

Introduction

Predictive modelling is the general concept of building a model that describes how a group of covariates can be used to predict a response variable. The objective is to predict the unknown responses of observations given their covariates. For example, predictive models could be used to predict the sale price of houses given house characteristics (De Cock, 2011). In its simplest form, a predictive model aims to provide a point prediction for a new observation. However, a point prediction does not contain information about its precision that can tell us how close to the true response we can expect the prediction to be, which is often important in decision-making context. Hence, although the point prediction is often the main goal of predictive analysis, assessing its reliability is equally important, and this can be achieved with a prediction interval (PI). A PI contains a set of likely values for the true response with an associated level of confidence, usually, 90% or 95%. Given that shorter PIs are more informative, developing predictive models that can produce shorter PIs along with the point predictions is crucial in assessing and quantifying the prediction error. In real-world applications, knowing the prediction error alongside the point prediction increases the practical value of the prediction.

Regression analysis is a form of predictive modelling technique that examines the relationship between a response variable and a group of covariates. In this paper, we consider a general regression model

$$Y = g(X) + \epsilon \tag{1}$$

where Y is a univariate continuous response variable, X is a p-dimensional vector of predictors, and ϵ is an error term. We assume $g(.)$ is an unknown smooth function $\Re^p \rightarrow \Re$ and $E[Y|X = x] = g(X)$. A confidence interval of the prediction is a range likely to contain the location of the response variable's true population mean. However, a prediction interval for a new observation is wider than its corresponding confidence interval and provides a range likely to contain this new observation's response value.

In the past decade, random forests have increased in popularity and provide an efficient way to generate point predictions for model (1). A random forest is an ensemble method composed of many decision trees, which can be described with a simple algorithm (Breiman, 2001). For each tree $b = \{1, ..., B\}$, a bootstrap sample of observations is drawn and a fully grown tree is built such that a set of predictors is randomly selected at each node and the best split is selected among all possible splits with those predictors only. The random forest prediction for a new observation is the average of the B trees

$$\hat{y}_{new} = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_{new}^{b}$$

where $\hat{y}_{new}^{b}$ is the tree prediction for the new observation in the bth tree, i.e. the average of observations in the terminal node corresponding to the new observation. Besides this traditional description, the modern view also considers random forests as data-driven weight generators (Hothorn et al., 2004; Lin and Jeon, 2006; Moradian et al., 2017, 2019; Athey et al., 2019; Roy and Larocque, 2020; Tabib and Larocque, 2020; Alakuş et al., 2021).

Although random forests limit over-fitting by combining many trees, which reduces the variance of the estimator, final predictions can be biased (Mentch and Hooker, 2016; Wager and Athey, 2018). Since each tree is built under the same random process, all trees focus on the same part of the response signal, usually the strongest. Therefore, some parts of the response signal may be left untargeted, which could result in biased point predictions. Wager and Athey (2018) provide bounds for the extent of the bias of random forests under some assumptions about the tree growing process. Following their work, Ghosal and Hooker (2021) proposed a bias correction method in a regression framework called *one-step boosted forest*, which is introduced in Breiman (2001) and Zhang and Lu (2012). The main idea of the proposed method is to sum the predictions of two random forests, where the first is a regression forest fitted on the target data set and the second is fitted on the out-of-bag residuals of the former. Empirical studies show that this method provides a significant bias reduction when compared to a simple random forest.

The current paper proposes an R package providing, among other features, an extension of the one-step boosted forest method described above (Ghosal and Hooker, 2021). The literature on prediction intervals for random forests consists mostly of recent studies. The first method is the Quantile Regression Forests (QRF) method proposed by Meinshausen (2006). The aim of QRF is to estimate conditional quantiles of the response variable, instead of conditional means, using an estimated cumulative distribution function obtained with the nearest neighbour forest weights introduced by Hothorn et al. (2004). Prediction intervals can be built directly from the estimated conditional quantiles. The method is implemented in the CRAN package **quantregForest** (Meinshausen, 2017).

In a more recent study, Athey et al. (2019) proposed Generalized Random Forests (GRF), a very general framework to estimate any quantity, such as conditional means, quantiles or average partial effects, identified by local moment equations. Trees are grown with splitting rules designed to maximize heterogeneity with respect to the quantity of interest. Quantile regression forest is one of the applications of GRF. Similar to the QRF, the GRF method uses the neighbourhood information from different trees to compute a weighted set of neighbours for each test point. Unlike QRF, which grows trees with the least-squares criterion, GRF uses a splitting rule designed to capture heterogeneity in conditional quantiles. An implementation of quantile regression forest with GRF is available in the function `quantile_forest` of the CRAN package **grf** (Tibshirani et al., 2021).

Vovk et al. (2005, 2009) introduced a general distribution-free conformal prediction interval framework. Any predictive model, including random forests, can be used within the proposed methodology. The idea is to use an augmented data set that includes the new observation to be predicted to fit the model, and apply a set of hypothesis tests to provide an error bound around the point prediction for the new observation. Although this method does not require any distribution assumptions, it is computationally intensive. Lei et al. (2018) proposed a variant of this method, called Split Conformal (SC) prediction, which splits the data into two subsets, one to fit the model, and one to compute the quantiles of the residual distribution. We note that, while the original full conformal prediction interval framework produces shorter intervals, SC is computationally more efficient. The R package **conformalInference** (Tibshirani, 2019), available on GitHub, implements this method.

Roy and Larocque (2020) proposed 20 distinct variations of methods to improve the performance of prediction intervals with random forests. These approaches differ according to 1) the method used to build the forest and 2) the method used to build the prediction interval. Four methods can be used to build the forest: three from the classification and regression tree (CART) paradigm (Breiman and Breiman, 1984) and the transformation forest method (TRF) proposed by Hothorn and Zeileis (2021). Within the CART paradigm, in addition to the default least-squares (LS) splitting criterion, two alternative splitting criteria, L_1 and shortest prediction interval (SPI), are considered. Prediction intervals are built using the Bag of Observations for Prediction (BOP), which is the set of nearest neighbour observations previously used in Moradian et al. (2017, 2019). In addition to the type of forest chosen, there are also five methods to build prediction intervals: the classical method (LM), the quantile method (Quant), the shortest prediction interval (SPI), the highest density region (HDR), and the contiguous HDR (CHDR). LM is computed based on an intercept-only linear model using the BOP as the sample, and produces a symmetric PI around the point prediction. The quantile method, similar to the QRF method, is based on the quantiles of the BOP. SPI corresponds to the shortest interval among the intervals that contain at least $(1-\alpha)$ 100% of the observations in the BOP. As an alternative to SPI, HDR is the smallest region in the BOP, with the desired coverage $(1-\alpha)$. Note that HDR is not necessarily a single interval. If the distribution is multimodal, it can be formed by multiple intervals. Finally, CHDR is a way to obtain a single prediction interval from HDR intervals by building an interval with the minimum and maximum bounds of the HDR intervals.

Zhang et al. (2020) proposed a forest-based prediction interval method, called Out-of-Bag (OOB) prediction intervals, to estimate prediction intervals using the empirical quantiles of the out-of-bag prediction errors. The method assumes that OOB prediction errors are identically distributed and that their distribution can be well approximated by the out-of-bag prediction errors obtained from all training observations. The resulting prediction intervals have the same width for all test observations.

The method is implemented in the CRAN package **rfinterval** (Zhang, 2019).

Lu and Hardin (2021) proposed a very interesting and useful method to estimate the conditional prediction error distribution of a random forest. The main idea of the proposed method is to use a random forest to compute the out-of-bag residuals for the training observations and to form a set of out-of-bag neighbours for each test point. Then, the conditional prediction error distribution for each test point is determined with the out-of-bag residuals in the neighbourhood. Estimating the prediction error distribution enables the estimation of conditional quantiles, conditional biases and conditional mean squared prediction errors. The prediction interval for a test point x, defined by $\widehat{PI}_\alpha(x)$, is formed by adding the $\alpha/2$ and $1-\alpha/2$ quantiles of the conditional prediction error distribution to the random forest point prediction. The estimators are implemented in the CRAN package **forestError** (Lu and Hardin, 2020).

Note that the conformal inference, OOB approach of Zhang et al. (2020) and the $\widehat{PI}_\alpha(x)$ method of Lu and Hardin (2021) all use the prediction errors to build the prediction intervals. Instead of using the training responses directly to estimate quantiles, using prediction errors provides a better predictive power. However, unlike conformal inference and the OOB approach, the $\widehat{PI}_\alpha(x)$ method uses the nearest neighbour observations to estimate the prediction error distribution. This idea is very similar to the BOP idea (Roy and Larocque, 2020), but instead of using in-bag observations, Lu and Hardin (2021) use out-of-bag observations to form the neighbourhoods. This approach allows the local information for the test observations to be extracted.

In this paper, we introduce the R package **RFpredInterval** (Alakus et al., 2022), which is the novel implementation of 16 methods to build prediction intervals with random forests and boosted forests. The set of methods implemented in the package includes a new method to build prediction intervals with boosted forests and 15 method variations (three splitting rules with the CART paradigm which are LS, L_1 and SPI, and five methods to build prediction intervals which are LM, SPI, Quant, HDR and CHDR) proposed by Roy and Larocque (2020). These 15 methods had been thoroughly investigated before through simulation studies and with real data sets in Roy and Larocque (2020). However, they are not easily available to use. One of the main contributions of our package is the implementation of these competitive methods and the ability for users to compare various prediction interval methods within the same package. The other main contribution of our paper is a new method to build prediction intervals. Contrary to the 15 methods proposed by Roy and Larocque (2020), the newly introduced method was not tested before. That is why in the paper we placed a greater emphasis on investigating the new method through extensive simulation studies and with real data. For performance comparison purposes, we compared the new method to 10 existing methods which include:

- 5 of the 15 implemented method variations of Roy and Larocque (2020); see the Competing methods subsection for details.
- 5 other competing methods from the literature: Quantile Regression Forests (QRF), Generalized Random Forests (GRF), Split Conformal prediction method (SC), Out-of-Bag (OOB) prediction intervals method and $\widehat{PI}_\alpha(x)$ method.

The new proposed method to build **P**rediction **I**ntervals with **B**oosted **F**orests is called **PIBF**. This approach integrates the idea of using the nearest neighbour out-of-bag observations to estimate the conditional prediction error distribution presented in Lu and Hardin (2021) to the one-step boosted forest proposed by Ghosal and Hooker (2021). We will show in this paper that PIBF significantly improves the performance of prediction intervals with random forests when compared with 10 existing methods using a variety of simulated and real benchmark data sets.

The rest of the paper is organized as follows. In the next section, we describe the algorithm implemented in PIBF. We then present the details of the package and provide a practical and reproducible example. We also perform a simulation study to compare the performance of our proposed method to existing competing methods, and we investigate the performance of the proposed method with real data sets. Lastly, we conclude with a discussion of the results.

Method and implementation

The proposed method is based on the one-step boosted forest method proposed by Ghosal and Hooker (2021). It consists in fitting two regression random forests: the first is fitted to get point predictions and out-of-bag (OOB) residuals using the given data set, whereas the second is fitted to predict those residuals using the original covariates. As empirical studies demonstrate, the one-step boosted forest provides point predictions with reduced bias compared to the simple random forest. Ghosal and Hooker (2021) use subsampling for their theoretical investigations, even though random forests were originally described with bootstrap samples and obtain notable performance improvements. They

have also investigated the effect of using bootstrapping with the one-step boosted forest on the bias estimations. From the results presented in their appendix, the use of bootstrapping yields the best performance and reduces the bias the most in exchange for an increase in their proposed variance estimator, which is defined under asymptotic normality. In this paper, we use bootstrapping for the one-step boosted forest method following the better performance results on bias reduction. The final prediction for a new observation, x_{new}, is the sum of the predictions from the two random forests

$$\hat{y}^*_{new} = \hat{y}_{new} + \hat{\epsilon}_{new} \tag{2}$$

where $\hat{y}_{new}$ is the point prediction obtained from the first random forest and $\hat{\epsilon}_{new}$ is the bias estimation from the second forest.

Besides bias correction, we use the second random forest as a way to construct a prediction interval by finding the nearest neighbour observations that are close to the one we want to predict. The idea of finding the nearest neighbour observations, a concept very similar to the 'nearest neighbour forest weights' (Hothorn et al., 2004; Lin and Jeon, 2006), was introduced in Moradian et al. (2017) and later used in Moradian et al. (2019), Roy and Larocque (2020), Tabib and Larocque (2020) and Alakuş et al. (2021). For a new observation, the set of in-bag training observations that are in the same terminal nodes as the new observation forms the set of nearest neighbour observations. Roy and Larocque (2020) called this set of observations the Bag of Observations for Prediction (BOP). We can define the BOP for a new observation x_{new} as

$$BOP\left(x_{new}\right) = \bigcup_{b=1}^{B} I_b\left(x_{new}\right) \tag{3}$$

where $I_b\left(x_{new}\right)$ is the set of in-bag training observations, *i.e.*, observations in the bootstrap sample that are in the same terminal node as x_{new} in the bth tree. $I_b\left(.\right)$ consists of the training observations that are in the bootstrap sample of the bth tree.

Instead of forming the set of nearest neighbour observations with the in-bag training observations, we can use the out-of-bag observations which are not in the bootstrap sample, as used in Lu and Hardin (2021). We can define the out-of-bag equivalent of the BOP for a new observation x_{new} (3) as

$$BOP^*\left(x_{new}\right) = \bigcup_{b=1}^{B} O_b\left(x_{new}\right) \tag{4}$$

where $O_b\left(x_{new}\right)$ is the set of out-of-bag observations that are in the same terminal node as x_{new} in the bth tree. $O_b\left(.\right)$ consists of the training observations that are not in the bootstrap sample of the bth tree.

Out-of-bag observations are not used in the tree growing process. Thus, for the trees where the training observations are out-of-bag, they are like the unobserved test observations for those trees. The only difference is that, for a new observation, we use all the trees in the forest whereas for an out-of-bag observation we have only a subset of the forest trees. By using the out-of-bag equivalent of the BOP for a new observation, we can make use of the analogy between the out-of-bag observations and test observations. The out-of-bag neighbours of a new observation represent the new observation better than the in-bag neighbours.

Any desired measure can be obtained by using the constructed BOPs. In this paper, we use the BOP idea to build a prediction interval for a test observation. For a new observation with covariates x_{new}, we firstly form $BOP^*\left(x_{new}\right)$ (4) using the out-of-bag neighbours. Then, as proposed by Lu and Hardin (2021), we estimate the conditional prediction error distribution, $\hat{F}\left(x_{new}\right)$, but now with the bias-corrected out-of-bag residuals of the observations in $BOP^*\left(x_{new}\right)$. Lastly, we build a prediction interval for the new observation as

$$PI\left(x_{new}\right) = \left[\hat{y}^*_{new} + SPI^l_\alpha\left(\hat{F}\left(x_{new}\right)\right), \hat{y}^*_{new} + SPI^u_\alpha\left(\hat{F}\left(x_{new}\right)\right)\right] \tag{5}$$

where $\hat{y}^*_{new}$ is the bias-corrected prediction, $SPI^l_\alpha\left(\hat{F}\left(x_{new}\right)\right)$ and $SPI^u_\alpha\left(\hat{F}\left(x_{new}\right)\right)$ are the lower and upper bounds of the $SPI_\alpha\left(\hat{F}\left(x_{new}\right)\right)$, which is the shortest interval formed by the observations in $BOP^*\left(x_{new}\right)$ that contains at least $\left(1-\alpha\right)100\%$ of the observations. By using the bias-corrected residuals to form prediction error distribution and picking the shortest interval among the qualified intervals, we can expect narrower prediction intervals.

We can summarize the steps of the proposed method as follows:

1. Train the first regression RF with covariates X to predict the response variable Y, and get the OOB predictions $\hat{Y}_{oob}$
2. Compute the OOB residuals as $\hat{\epsilon}_{oob} = Y - \hat{Y}_{oob}$
3. Train the second regression RF with covariates X to predict the OOB residuals $\hat{\epsilon}_{oob}$, and get the

OOB predictions for residuals $\hat{\hat{\epsilon}}_{oob}$

4. Update the OOB predictions as $\hat{Y}^*_{oob} = \hat{Y}_{oob} + \hat{\hat{\epsilon}}_{oob}$
5. Compute the updated OOB residuals after bias-correction as $\hat{\epsilon}^*_{oob} = Y - \hat{Y}^*_{oob}$
6. For a new observation x_{new}, get the point predictions $\hat{y}_{new}$ from the first RF, and get the predicted residuals $\hat{\epsilon}_{new}$ from the second RF, then the final prediction for the new observation is

$$\hat{y}^*_{new} = \hat{y}_{new} + \hat{\epsilon}_{new}$$

where $\hat{\epsilon}_{new}$ is the estimated bias.

7. Form a BOP for x_{new} with the OOB neighbours using the second RF, $BOP^*(x_{new})$ (4) and estimate the conditional prediction error distribution for x_{new} as,

$$\hat{F}(x_{new}) = \left\{\hat{\epsilon}^*_{oob,i} | i \in BOP^*(x_{new})\right\}$$

8. Build a PI for x_{new} as $PI(x_{new}) = \left[\hat{y}^*_{new} + SPI^l_\alpha\left(\hat{F}(x_{new})\right), \hat{y}^*_{new} + SPI^u_\alpha\left(\hat{F}(x_{new})\right)\right]$

Calibration

The principal goal of any prediction interval method is to ensure the desired coverage level. In order to attain the desired coverage level $(1-\alpha)$, we may need a calibration procedure. The goal of the calibration is to find the value of α_w, called the working level in Roy and Larocque (2020), such that the coverage level of the PIs for the training observations is closest to the desired coverage level. Roy and Larocque (2020) presented a calibration procedure that uses the BOPs that are built using only the trees where the training observation x_i is OOB. The idea is to find the value of α_w using the OOB-BOPs. In this paper, we call this procedure OOB calibration.

We also include a cross-validation-based calibration procedure with the proposed method to acquire the desired $(1-\alpha)$ coverage level. In this calibration, we apply k-fold cross-validation to form prediction intervals for the training observations. In each fold, we split the original training data set into training and testing sets. For the training set, we go through the steps 1-5 defined above. Then, for each observation in the testing set, we apply steps 6-8 and build a PI. After completing CV, we compute the coverage level with the constructed PIs and if the coverage is not within the acceptable coverage range, then we apply a grid search to find the α_w such that α_w is the closest to the target α among the set of α_w's. Once we find the α_w, we use this level to build the PI for the new observations.

The RFpredInterval package

In our package, we implement 16 methods that apply random forest training. Ten of these methods have specialized splitting rules in the random forest growing process. These methods are the ones with L_1 and shortest prediction interval (SPI) splitting rules proposed by Roy and Larocque (2020). To implement these methods, we have utilised the custom split feature of the **randomForestSRC** package (Ishwaran and Kogalur, 2021).

The **randomForestSRC** package allows users to define a custom splitting rule for the tree growing process. The user needs to define the customized splitting rule in the `splitCustom.c` file with C-programming. After modifying the `splitCustom.c` file, all C source code files in the package's `src` folder must be recompiled. Finally, the package must be re-installed for the custom split rule to become available.

In our package development process, we froze the version of **randomForestSRC** to the latest one available at the time, which is version 2.11.0, to apply specialized splitting rules. After defining the L_1 and SPI splitting rules, all C files were re-compiled. Finally, all package files including our R files for prediction interval methods were re-built to make the package ready for the user installation.

The **RFpredInterval** package has two main R functions as below:

- `pibf()`: Constructs prediction intervals with the proposed method, PIBF.
- `rfpi()`: Constructs prediction intervals with 15 distinct variations proposed by Roy and Larocque (2020).

Table 1 presents the list of functions and methods implemented in **RFpredInterval**. For `pibf()`, **RFpredInterval** uses the CRAN package **ranger** (Wright et al., 2020) to fit the random forests. For `rfpi()`, **RFpredInterval** uses **randomForestSRC** package. For the least-squares splitting rule, both **randomForestSRC** and **ranger** packages are applicable.

Table 1: List of functions and methods with characteristics

Function	Method		Details
`pibf()`	PIBF		Builds prediction intervals with boosted forests. The **ranger** package is used to fit the random forests. Calibration options are `"cv"` and `"oob"`. Returns constructed PIs and bias-corrected point predictions for the test data.
`rfpi()`	*Split method*	*PI method*	
	LS	LM SPI Quant HDR CHDR	Splitting rule is the least-squares (LS) from the CART paradigm. `"ls"` is used for the `"split_method"` argument. A vector of characters `c("lm","spi","quant","hdr","chdr")` is used for the `"pi_method"` argument to apply all or a subset of the PI methods. **ranger** or **randomForestSRC** can be used to fit the random forest. Returns a list of constructed PIs for the selected PI methods and point predictions for the test data.
	L_1	LM SPI Quant HDR CHDR	Splitting rule is the L_1 from the CART paradigm (Roy and Larocque, 2020). `"l1"` is used for the `"split_method"` argument. A vector of characters `c("lm","spi","quant","hdr","chdr")` is used for the `"pi_method"` argument to apply all or a subset of the PI methods. Only **randomForestSRC** can be used to fit the random forest since the split rule is implemented with the custom split feature of that package. Returns a list of constructed PIs for the selected PI methods and point predictions for the test data.
	SPI	LM SPI Quant HDR CHDR	Splitting rule is the shortest PI (SPI) from the CART paradigm (Roy and Larocque, 2020). `"spi"` is used for the `"split_method"` argument. A vector of characters `c("lm","spi","quant","hdr","chdr")` is used for the `"pi_method"` argument to apply all or a subset of the PI methods. Only **randomForestSRC** can be used to fit the random forest since the split rule is implemented with the custom split feature of that package. Returns a list of constructed PIs for the selected PI methods and point predictions for the test data.
`piall()`	All methods		Builds prediction intervals with all of the implemented PI methods. The **ranger** package is used to fit the random forests for the PIBF and methods with LS split rule. **randomForestSRC** package is used for the methods with L_1 and SPI split rules. Default values are assigned to the function arguments of `pibf()` and `rfpi()`. Returns an object of class `"piall"` containing a list of constructed PIs with 16 methods, and point predictions obtained with the PIBF method, LS, L_1 and SPI split rules for the test data.
`plot()`			Plots the 16 constructed PIs obtained with `piall()` function for a test observation.
`print()`			Prints the summary output of `pibf()`, `rfpi()` and `piall()` functions.

LM: Classical method, **SPI**: Shortest PI, **Quant**: Quantiles, **HDR**: Highest density region, **CHDR**: Contiguous HDR

In this section, we illustrate the usage of the **RFpredInterval** package with the Ames Housing data set (De Cock, 2011). The data set was introduced as a modern alternative to the well-known Boston Housing data set. The data set contains many explanatory variables on the quality and quantity of physical attributes of houses in Ames, IA sold from 2006 to 2010. Most of the variables give information to a typical home buyer who would like to know about a house (*e.g.* number of bedrooms and bathrooms, square footage, heating type, lot size, etc.).

The **AmesHousing** (Kuhn, 2020) package contains the raw data and processed versions of the Ames Housing data set. The raw data contains 2930 observations and 82 variables, which include 23 nominal, 23 ordinal, 14 discrete, and 20 continuous variables, involved in assessing house values. The processed version of the data set has 2330 observations and 81 variables, including the target variable `Sale_Price` representing the value of houses in US dollars. The usual goal for this data set is to predict the sale price of each house given covariates.

We load the processed version of the Ames Housing data set from the **AmesHousing** package and prepare the data set that we will use for the analyses. The preprocessing steps are presented in the Supplementary Material. This version of the data set contains 22 factors and 59 numeric variables, including 1 response variable `Sale_Price`, for 2929 observations. We split the data set into training and testing samples.

```
set.seed(3456)
n <- nrow(AmesHousing)
trainindex <- sample(1:n, size = round(0.7*n), replace = FALSE)
traindata <- AmesHousing[trainindex, ]
testdata <- AmesHousing[-trainindex, ]
```

We fit a random forest with 1000 trees using the training data and construct 95% prediction intervals for the observations in the testing data with the proposed method. We apply 5-fold cross-validation based calibration and set the acceptable coverage range to $[.945, .955]$. We can pass the list of random forest parameters for **ranger** package.

```
out <- pibf(formula = Sale_Price ~ .,
            traindata = traindata,
            testdata = testdata,
            alpha = 0.05,
```

```
            calibration = "cv",
            numfolds = 5,
            coverage_range = c(0.945, 0.955),
            params_ranger = list(num.trees = 1000),
            oob = TRUE)
```

We can then analyze the constructed PIs and bias-corrected random forest predictions for the testing data, as shown below. The PI output is a list containing lower and upper bounds. For example, we can print the point prediction and prediction interval for the tenth observation in the testing data.

```
out$pred_interval
out$test_pred
c(out$pred_interval$lower[10], out$test_pred[10], out$pred_interval$upper[10])
[1] 133.8629 160.2426 194.5804
```

We can also print the summary output. In the summary output, we can always see the mean PI length over the test data set. If calibration is applied, we can see the working level of α. If the test data set has true response information, as in our example, coverage and prediction errors for the test set are also printed. Moreover, since we have entered `oob = TRUE` in the function arguments, in the summary output we can see the mean PI length and coverage measures along with the prediction errors for the training set. The prediction intervals are built with the out-of-bag (OOB) predictions and prediction errors.

```
print(out)

>                       alpha_w: 0.050
>                Mean PI length: 73.081
>                      Coverage: 96.8%
>       MAE of test predictions: 12.773
>      RMSE of test predictions: 19.545
>
>      Mean PI length (OOB PIs): 74.823
>            Coverage (OOB PIs): 94.7%
>  MAE of OOB train predictions: 14.179
> RMSE of OOB train predictions: 23.875
```

Next, we construct 95% prediction intervals using the variations proposed by Roy and Larocque (2020). In the following example, the splitting is rule is set to L_1 and we want to apply LM, Quant and SPI methods for building prediction intervals. We apply OOB calibration and set the acceptable coverage range to $[.945, .955]$. We can pass the the list of random forest parameters for **randomForestSRC** package.

```
out2 <- rfpi(formula = Sale_Price ~ .,
             traindata = traindata,
             testdata = testdata,
             alpha = 0.05,
             calibration = TRUE,
             split_rule = "l1",
             pi_method = c("lm", "quant", "spi"),
             params_rfsrc = list(ntree = 1000),
             params_calib = list(range = c(0.945, 0.955)),
             oob = FALSE)
```

We can analyze the constructed PIs for the testing data as below. Each PI output is a list containing lower and upper bounds. For instance, we can print the point prediction and LM prediction interval for the tenth observation in the testing data.

```
out2$lm_interval
out2$quant_interval
out2$spi_interval
c(out2$lm_interval$lower[10], out2$test_pred[10], out2$lm_interval$upper[10])
[1] 129.9474 154.2098 176.8429
```

We print the summary output. In the summary output, we can see the splitting rule selected in the first row. Since the test data set has true responses in our example, we can see the coverage information for the selected PI methods besides the mean PI length and α_w in the printed table. Below the table, we have the mean prediction errors for the test set.

```
print(out2)
```

```
>                    Split rule: L1
> ---------------------------------------------------------------------------------
>                                      Mean PI length      Coverage          alpha_w
> Classical method (LM)                    81.641            96.2%            0.140
> Shortest prediction interval (SPI)       81.953            96.1%            0.100
> Quantile method (Quant)                  80.893            95.8%            0.120
> ---------------------------------------------------------------------------------
>       MAE of test predictions: 14.605
>      RMSE of test predictions: 22.603
```

Although, with the `pibf()` and `rfpi()` functions, we have more flexibility to set the arguments for the methods, we can build prediction intervals with all 16 methods implemented in the package with the `piall()` function. We will build 95% prediction intervals for the test set.

```
out3 <- piall(formula = Sale_Price ~ .,
              traindata = traindata,
              testdata = testdata,
              alpha = 0.05,
              num.trees = 1000)
```

The output is a list of constructed prediction intervals with 16 methods and point predictions obtained with the PIBF method, LS, L_1, and SPI split rules. Hence, the output includes 16 prediction intervals and 4 point predictions which is a list of 20 items in total.

We print the summary output.

```
print(out3)
```

```
> ----------------------------------------
>              Mean PI length    Coverage
> PIBF             72.752          96.6%
> LS-LM            81.800          96.5%
> LS-SPI           82.662          95.4%
> LS-Quant         81.523          95.2%
> LS-HDR           81.733          96.0%
> LS-CHDR          83.025          96.1%
> L1-LM            81.649          96.1%
> L1-SPI           81.805          96.1%
> L1-Quant         80.836          95.3%
> L1-HDR           83.032          96.4%
> L1-CHDR          82.482          96.1%
> SPI-LM           81.578          96.0%
> SPI-SPI          81.960          96.2%
> SPI-Quant        81.036          95.4%
> SPI-HDR          84.404          96.6%
> SPI-CHDR         82.927          96.1%
> ----------------------------------------
>                    MAE            RMSE
> PIBF             12.774         19.428
> LS split         14.412         22.413
> L1 split         14.596         22.569
> SPI split        14.558         22.600
```

Lastly, we plot the constructed prediction intervals with all 16 methods, for the 15th observation in the test set.

```
plot(out3, test_id = 15)
```

Figure 1 presents the prediction intervals and point predictions for the test observation. The methods are ordered in the y-axis based on their resulting PI length. For each method, the red point presents the point prediction and blue lines show the constructed prediction interval(s) for the test observation. If the true response of the test observation is known, it is demonstrated with a dashed vertical line. Note that we may have multiple prediction intervals with the HDR PI method. As we can see from the figure, we may have four different point predictions for the same test observation. The PIBF method and the three splitting rules LS, L_1 and SPI can produce different point predictions. But all PI method variations for the same splitting rule have the same point prediction.

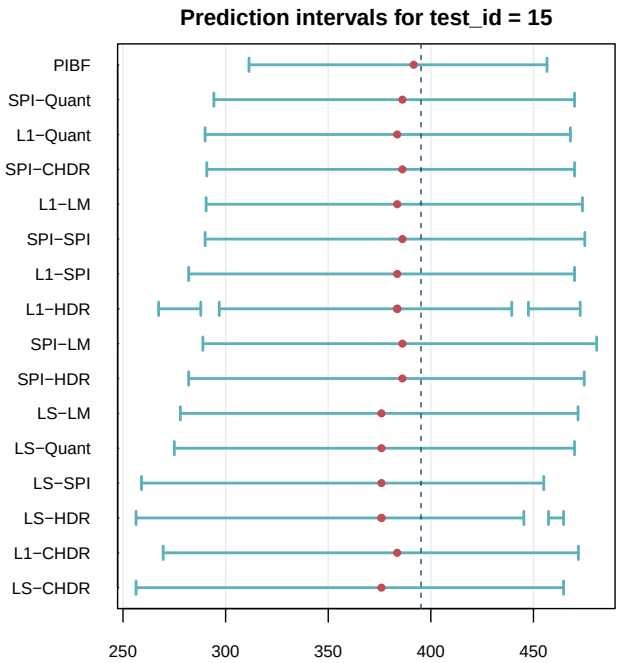

Figure 1: Prediction intervals for the 15th test observation in the test data. The x-axis represents the sale price of houses in thousands. For each method, red dots represent the point prediction and blue lines show the prediction interval(s). The vertical dashed line shows the true response value for the test observation. PIBF: Prediction intervals with boosted forests (the proposed method). The notation for the other 15 methods is *split rule-PI method*. Splitting rules are LS: Least-squares, L1: L_1, SPI: Shortest PI split rule. PI methods are LM: Classical method, Quant: Quantiles, SPI: Shortest PI, HDR: Highest density region, CHDR: Contiguous HDR.

Simulation study

In this section, we compare the predictive performance of the prediction intervals constructed with our proposed method to the existing methods presented in the Introduction using a variety of simulated and real benchmark data sets.

Simulation design

We apply a simulation study based on seven simulated data sets from the literature. The first three of the data sets are Friedman's benchmark regression problems described in Jerome H. Friedman (1991) and Breiman (1996). We use the CRAN package **mlbench** (Leisch and Dimitriadou, 2021) to generate these data sets.

In Friedman Problem 1, the inputs are 10 independent variables uniformly distributed on the interval $[0, 1]$. The first five covariates are used to generate the response:

$$y = 10 \sin(\pi x_1 x_2) + 20 (x_3 - 0.5)^2 + 10 x_4 + 5 x_5 + \epsilon$$

where ϵ is $N(0, \sigma^2)$ and the default standard deviation of ϵ is 1 which yields a signal-to-noise ratio (SNR) (*i.e.*, the ratio of the standard deviation of signal to the standard deviation of error) of 4.8:1.

In Friedman Problem 2, the response is generated as

$$y = \left(x_1^2 + \left(x_2 x_3 - \frac{1}{x_2 x_4} \right)^2 \right)^{0.5} + \epsilon$$

where the inputs are four independent variables uniformly distributed over the ranges

$$\begin{aligned} 0 \leq & x_1 \leq 100 \\ 40\pi \leq & x_2 \leq 560\pi \\ 0 \leq & x_3 \leq 1 \\ 1 \leq & x_4 \leq 11 \end{aligned}$$

and ϵ is $N\left(0, \sigma^2\right)$. The default value of 125, which yields a SNR of 3:1, is used for the standard deviation of ϵ.

In Friedman Problem 3, the inputs are four independent variables uniformly distributed over the same ranges as Friedman Problem 2. The response is generated as

$$y = \arctan\left(\frac{x_2 x_3 - \frac{1}{x_2 x_4}}{x_1} \right) + \epsilon$$

where ϵ is $N\left(0, \sigma^2\right)$ and the default value of 0.01 for the standard deviation of ϵ is used, which yields a SNR of 3:1.

The fourth data set is the Peak Benchmark Problem which is also from the **mlbench** package. Let $r = 3u$ where u is uniform on $[0, 1]$ and let x be uniformly distributed on the d-dimensional sphere of radius r. The response is $y = 25 \exp\left(-0.5 r^2\right)$. The default value of $d = 20$ dimensions is used.

The fifth one is a modification of Friedman Problem 1, which was used in Hothorn and Zeileis (2021) in their H2c setup. This data set was designed to have heteroscedasticity. The inputs are 10 independent variables uniformly distributed on the interval $[0, 1]$. The first five covariates are used in the mean function and the unscaled mean function is defined as

$$\mu = 10 \sin\left(\pi x_1 x_2\right) + 20\left(x_3 - 0.5\right)^2 + 10 x_4 + 5 x_5$$

Then, the scaled mean function on the interval $[-1.5, 1.5]$ is

$$\mu^S = \frac{3\left(\mu - \mu_{min}\right)}{\mu_{max} - \mu_{min}} - 1.5$$

where μ_{min} and μ_{max} are the minimum and maximum values of μ over the sample. The last five covariates are used in the standard deviation function and the unscaled standard deviation function is

$$\sigma = 10 \sin\left(\pi x_6 x_7\right) + 20\left(x_8 - 0.5\right)^2 + 10 x_9 + 5 x_{10}$$

The standard deviation is scaled as

$$\sigma^S = \exp\left(\frac{3\left(\sigma - \sigma_{min}\right)}{\sigma_{max} - \sigma_{min}} - 1.5 \right)$$

where σ_{min} and σ_{max} are the minimum and maximum values of σ over the sample. The response is generated as a normal random variable with mean μ^S and standard deviation σ^S.

The last two data sets, which were used in Roy and Larocque (2020), have a tree-based response variable. The inputs are seven independent variables generated from the standard normal distribution. The response is generated with the seven covariates according to a tree model with a depth of three, with eight terminal nodes:

$$\begin{aligned} y = & u_1 I\left(x_1 < 0, x_2 < 0, x_4 < 0\right) \\ & + u_2 I\left(x_1 < 0, x_2 < 0, x_4 \geq 0\right) \\ & + u_3 I\left(x_1 < 0, x_2 \geq 0, x_5 < 0\right) \\ & + u_4 I\left(x_1 < 0, x_2 \geq 0, x_5 \geq 0\right) \\ & + u_5 I\left(x_1 \geq 0, x_3 < 0, x_6 < 0\right) \\ & + u_6 I\left(x_1 \geq 0, x_3 < 0, x_6 \geq 0\right) \\ & + u_7 I\left(x_1 \geq 0, x_3 \geq 0, x_7 < 0\right) \\ & + u_8 I\left(x_1 \geq 0, x_3 \geq 0, x_7 \geq 0\right) + \epsilon \end{aligned}$$

where the terminal node means are $u = (5, 10, 15, 20, 25, 30, 35, 40)$ and I is the indicator function. The difference in the two data sets is the distribution of the error. In the first one, ϵ is generated from a standard normal distribution and in the other it is from an exponential distribution with mean 1. The signal-to-noise ratio is 11.5:1 for both data sets.

We use training sample sizes of $n_{train} = \{200, 500, 1000, 5000\}$, resulting in 28 scenarios. Each scenario is repeated 500 times. In each run, we generate an independent test set of new observations with $n_{test} = 1000$.

Competing methods

We compare our proposed prediction interval estimator with 10 competing methods which were presented in the Introduction. The first is the $\widehat{PI}_\alpha$ method. We fit the random forest with the **ranger** package and use the **forestError** package to build PIs. The second is the OOB method. The **rfinterval** package is used. The third is the split conformal method. The **conformalInference** package is used. The fourth is the QRF method and the **quantregForest** package is used. The fifth is the GRF method. The function `quantile_forest` in the **grf** package is used.

The last five are variations of Roy and Larocque (2020). To compare the performance of the method variations, a comprehensive simulation study and real data analyses were performed in Roy and Larocque (2020). One of the biggest conclusions from these comparison studies was that, among the three alternative splitting criteria within the CART paradigm, the impact of the choice of the splitting rule on the performance of the prediction intervals was moderate whereas the selection of the PI method had a much greater impact on the performance. Hence, in this simulation study, we set up the splitting rule to the least-squares (LS) and only compare the five PI methods: LM, Quant, SPI, HDR, and CHDR. For those methods, the `rfpi()` function of the **RFpredInterval** package is used. We fit the random forest with the **ranger** package.

Parameter settings

For the simulations, we use the following parameters. For all methods, we set the number of trees to 2000. Letting p be the number of covariates, then the number of covariates to randomly split at each node, *mtry*, is set to $\max\{\lfloor p/3 \rfloor, 1\}$ (except for the GRF method). For the GRF method, following Athey et al. (2019), *mtry* is set to $\min\{\lceil \sqrt{p} + 20 \rceil, p\}$. Also, we use the honest splitting regime with the default fraction of 0.5 for the GRF method. The minimum node size parameter for all forests is set to 5. The desired coverage is set to 95% for all methods. For the proposed method, we perform the cross-validation-based calibration as the primary calibration procedure, but we also investigate the OOB calibration. For the method variations in Roy and Larocque (2020), we perform the OOB calibration procedure as they proposed. For both calibration procedures, the acceptable range of coverage is set to $[.945, .955]$. Calibration is not performed for the competing methods since no option for calibration is offered in their CRAN packages.

Performance with the simulated data sets

We can evaluate the performance of the competing methods with two measures: the mean coverage and the mean prediction interval length. Table 2 presents the average coverage rate of each method on the test set over 500 replications for a given simulated data set and sample size, with average mean prediction interval lengths shown in parentheses. The principal goal of any prediction interval method is to ensure the desired coverage level. In this simulation study, the desired coverage level is set to 95% for all methods. The left plot in Figure 2 shows the mean coverages over the 28 scenarios for all methods. Overall, all of the methods, except the QRF and GRF methods which tend to be conservative, provide a mean coverage close to the desired level. QRF and GRF methods have an average mean coverage of 0.975 and 0.974 over all scenarios, respectively. Although the $\widehat{PI}_\alpha$ method has an average of the mean coverages of 0.957, close to the desired level, its variability is large. Over 308 (11 methods $\times$ 28 scenarios) average coverage values, there is only one case where the mean coverage is below 0.94. It corresponds to the $\widehat{PI}_\alpha$ method in Friedman Problem 2 with $n_{train} = 5000$.

Once the prediction intervals provide the desired coverage level, the next goal of any PI method is to provide the shortest PI length. Prior to carrying out a detailed comparison of interval lengths, we can globally compare the interval lengths over all scenarios with the percentage increase in mean PI length of a method with respect to the best method for a given run. For a given run, define ml_i as the mean PI length of method i and ml^* as the shortest mean PI length over the 10 competing methods. The percentage increase in PI length for method i is computed as $100 \times (ml_i - ml^*)/ml^*$. Smaller values for this measure indicate better performances. The right plot in Figure 2 presents the relative

lengths of the methods across 14,000 runs (28 scenarios × 500 replications). The prediction intervals with the GRF method are the widest, followed by the QRF method. However, as we saw in the left plot in Figure 2, GRF and QRF produce conservative prediction intervals, so their PI lengths cannot be fairly compared to the other methods with a coverage closer to 0.95. Based on the global results, the proposed method, PIBF, performs the best. Following PIBF, $\widehat{PI}_\alpha$, OOB, LM, SPI, HDR and CHDR perform similarly well, with $\widehat{PI}_\alpha$ being slightly better. Among the variations of Roy and Larocque (2020), the PIs with quantiles produce longer prediction intervals than the other four variations.

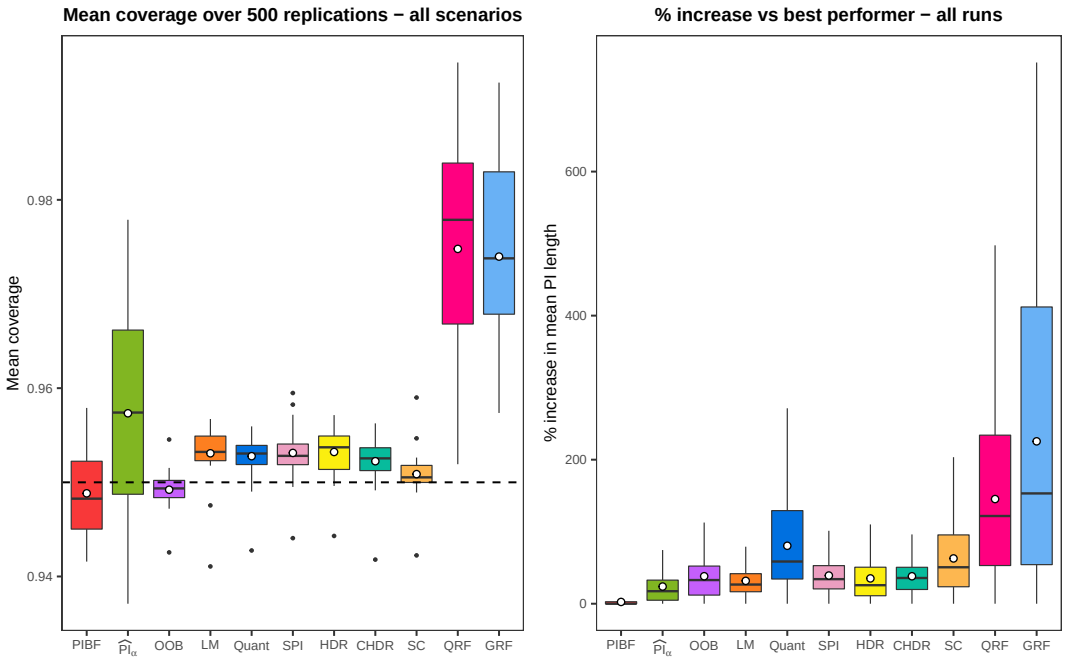

Figure 2: *(Left)* Boxplots for the mean coverage over all scenarios. All methods, except the QRF and GRF methods, are able to provide a mean coverage close to the desired coverage level of 0.95. Each white circle is the average of the mean coverages over 28 scenarios. *(Right)* Boxplots for the percentage increase in mean PI length of each method compared to the shortest PI length for a given run across 14,000 runs. The smallest is the percentage increase, the better is the method. Each white circle is the average of the relative lengths over 14,000 runs. Since the outlier values are distorting the scales, they are removed from the graph. PIBF: Prediction intervals with boosted forests (the proposed method), $\widehat{PI}_\alpha$: Conditional α-level prediction interval, OOB: Out-of-Bag approach, LM: Classical method, Quant: Quantiles, SPI: Shortest PI, HDR: Highest density region, CHDR: Contiguous HDR, SC: Split conformal, QRF: Quantile regression forest, GRF: Generalized random forest.

Now, we investigate the performance of the methods separately for each scenario. See figures S1 to S7 in the Supplementary Material for the mean PI length results of each method for each simulated data set. Each figure has four facets corresponding to the four levels of the training sample size. For all methods and data sets, the mean PI lengths and their variability decrease as the sample size increases (except the GRF method for the tree-based data sets). We see that for Friedman Problem 1, from Figure S1, for all sample sizes, PIBF consistently outperforms the 10 competing methods in terms of mean PI length while ensuring the desired coverage level (see Table 2 for the mean coverage results). QRF and GRF provide the widest prediction intervals for all sample sizes. However, as presented in Table 2, these methods heavily over-cover and are therefore not comparable with the other methods. While the $\widehat{PI}_\alpha$ method also slightly over-covers, it has shorter PIs than other methods.

For Friedman Problem 2 (see Figure S2 in the Supplementary Material), we see that the proposed method has the shortest mean PI length for the smallest sample size, and as the sample size increases $\widehat{PI}_\alpha$ provides shorter PIs. However, we should also take into account the mean coverages presented in Table 2. The proposed method has smaller coverage levels for $n_{train} = 200$ compared to $\widehat{PI}_\alpha$, but as the sample size increases the coverage levels decrease for the $\widehat{PI}_\alpha$ method (up to 0.937 for $n_{train} = 5000$) whereas PIBF keeps it around 0.945. For $n_{train} = 5000$, the OOB method builds shorter PIs while ensuring the desired coverage level. Again, QRF and GRF have the widest PIs for all sample sizes due to their conservative PIs.

The performance of PIBF and $\widehat{PI}_\alpha$ is very similar for Friedman Problem 3 (see Figure S3 in the Supplementary Material). Both methods provide the shortest PIs with similar coverage levels and

Table 2: Results of the simulation study. Average coverage rates of each method in each simulation, with average mean PI lengths shown in parentheses. The desired coverage level is 0.95. Shortest average mean PI lengths are emphasized with bold text. PIBF: Prediction intervals with boosted forests (the proposed method), $\widehat{PI}_\alpha$: Conditional α-level prediction interval, OOB: Out-of-Bag approach, LM: Classical method, Quant: Quantiles, SPI: Shortest PI, HDR: Highest density region, CHDR: Contiguous HDR, SC: Split conformal, QRF: Quantile regression forest, GRF: Generalized random forest.

n_{train}	Data	PIBF	$\widehat{PI}_\alpha$	OOB	LM	Quant	SPI	HDR	CHDR	SC	QRF	GRF
200	Friedman 1	0.952 (**7.69**)	0.959 (9.7)	0.949 (10.3)	0.955 (11.2)	0.956 (12.5)	0.956 (12.1)	0.955 (12.2)	0.956 (11.5)	0.952 (12)	0.978 (15.3)	0.968 (17.7)
	Friedman 2	0.942 (**635**)	0.951 (679)	0.947 (762)	0.956 (772)	0.955 (990)	0.956 (898)	0.955 (793)	0.954 (800)	0.951 (912)	0.969 (1103)	0.972 (1241)
	Friedman 3	0.944 (**0.57**)	0.948 (0.62)	0.949 (0.75)	0.956 (0.69)	0.953 (0.83)	0.952 (0.74)	0.951 (0.88)	0.954 (0.76)	0.953 (0.93)	0.961 (0.89)	0.968 (0.88)
	Peak	0.958 (**9.06**)	0.956 (12.4)	0.949 (13.3)	0.955 (12)	0.956 (17.9)	0.957 (10.5)	0.955 (15.4)	0.955 (11.1)	0.951 (15.5)	0.972 (20.6)	0.978 (20.2)
	H2c	0.954 (6.5)	0.954 (6.3)	0.955 (6.26)	0.955 (**5.86**)	0.956 (6.42)	0.957 (6.47)	0.956 (6.61)	0.955 (6.45)	0.959 (6.68)	0.952 (6.09)	0.957 (6.35)
	Tree-N	0.949 (9.12)	0.967 (12.3)	0.947 (14.2)	0.956 (11.9)	0.956 (22.3)	0.958 (15.4)	0.955 (**9.08**)	0.955 (11.9)	0.951 (18.9)	0.979 (29.6)	0.960 (36)
	Tree-exp	0.949 (**9.31**)	0.967 (12.4)	0.947 (14.2)	0.955 (11.9)	0.956 (22.1)	0.959 (15.1)	0.956 (10.4)	0.956 (12.5)	0.950 (18.9)	0.979 (29.5)	0.960 (35.8)
500	Friedman 1	0.952 (**6.32**)	0.962 (8.25)	0.948 (8.81)	0.953 (9.37)	0.952 (10.1)	0.953 (9.87)	0.952 (9.81)	0.953 (9.47)	0.951 (10.1)	0.986 (14)	0.978 (16.9)
	Friedman 2	0.945 (586)	0.946 (**585**)	0.948 (650)	0.952 (676)	0.952 (820)	0.952 (744)	0.951 (679)	0.951 (690)	0.952 (751)	0.975 (998)	0.979 (1120)
	Friedman 3	0.943 (**0.5**)	0.946 (0.53)	0.948 (0.62)	0.952 (0.61)	0.951 (0.71)	0.951 (0.65)	0.951 (0.7)	0.951 (0.67)	0.952 (0.75)	0.965 (0.79)	0.970 (0.77)
	Peak	0.957 (**6.44**)	0.967 (9.97)	0.951 (11)	0.953 (9.75)	0.954 (15)	0.954 (8.22)	0.956 (11.6)	0.953 (8.93)	0.952 (12.8)	0.980 (18.7)	0.984 (17.8)
	H2c	0.953 (5.81)	0.956 (5.88)	0.951 (5.89)	0.952 (**5.43**)	0.953 (5.83)	0.953 (5.81)	0.954 (5.92)	0.952 (5.84)	0.955 (6.17)	0.954 (5.79)	0.959 (5.94)
	Tree-N	0.948 (**6.16**)	0.969 (8.17)	0.949 (9.85)	0.952 (8.23)	0.953 (15.5)	0.953 (10)	0.954 (7.31)	0.952 (9.6)	0.952 (13.3)	0.984 (25.8)	0.969 (35.7)
	Tree-exp	0.947 (**6.31**)	0.966 (8.3)	0.949 (9.93)	0.952 (8.19)	0.953 (15.4)	0.954 (9.67)	0.954 (7.93)	0.953 (9.96)	0.953 (13.4)	0.984 (25.8)	0.970 (35.5)
1000	Friedman 1	0.953 (**5.67**)	0.964 (7.42)	0.949 (7.95)	0.954 (8.37)	0.954 (8.89)	0.954 (8.75)	0.954 (8.65)	0.953 (8.4)	0.950 (8.85)	0.990 (12.9)	0.985 (15.9)
	Friedman 2	0.945 (565)	0.942 (**546**)	0.950 (594)	0.953 (635)	0.953 (735)	0.953 (682)	0.952 (637)	0.953 (645)	0.951 (658)	0.977 (910)	0.983 (1022)
	Friedman 3	0.944 (**0.47**)	0.945 (0.48)	0.948 (0.55)	0.954 (0.58)	0.952 (0.65)	0.952 (0.61)	0.951 (0.63)	0.952 (0.63)	0.949 (0.63)	0.967 (0.73)	0.971 (0.72)
	Peak	0.957 (**5**)	0.972 (8.44)	0.950 (9.54)	0.952 (8.51)	0.953 (13.2)	0.953 (7.21)	0.956 (9.35)	0.952 (7.82)	0.951 (11)	0.983 (17.5)	0.986 (16.7)
	H2c	0.952 (5.48)	0.955 (5.64)	0.949 (5.69)	0.948 (**5.17**)	0.949 (5.49)	0.950 (5.51)	0.950 (5.49)	0.949 (5.49)	0.949 (5.78)	0.954 (5.61)	0.958 (5.71)
	Tree-N	0.946 (**5.11**)	0.965 (6.2)	0.950 (7.52)	0.954 (6.67)	0.953 (12.1)	0.954 (7.91)	0.954 (6.21)	0.953 (8.33)	0.950 (9.94)	0.985 (21.6)	0.976 (35.1)
	Tree-exp	0.946 (**5.16**)	0.964 (6.39)	0.949 (7.64)	0.954 (6.62)	0.953 (11.9)	0.954 (7.45)	0.955 (6.5)	0.953 (8.45)	0.950 (10)	0.985 (21.6)	0.975 (34.9)
5000	Friedman 1	0.950 (**4.71**)	0.967 (6.04)	0.950 (6.47)	0.953 (6.67)	0.953 (6.97)	0.953 (6.91)	0.954 (6.78)	0.953 (6.67)	0.950 (7.03)	0.995 (10.8)	0.992 (13.2)
	Friedman 2	0.945 (543)	0.937 (**507**)	0.950 (529)	0.952 (574)	0.951 (610)	0.951 (592)	0.951 (570)	0.951 (573)	0.950 (549)	0.975 (717)	0.982 (781)
	Friedman 3	0.945 (0.44)	0.941 (**0.43**)	0.950 (0.46)	0.953 (0.53)	0.952 (0.56)	0.952 (0.55)	0.951 (0.54)	0.952 (0.56)	0.950 (0.49)	0.966 (0.62)	0.971 (0.62)
	Peak	0.955 (**2.84**)	0.978 (5.92)	0.952 (7.11)	0.952 (6.54)	0.954 (10.1)	0.952 (5.68)	0.957 (6.46)	0.951 (6.04)	0.950 (8)	0.988 (14.9)	0.990 (15.2)
	H2c	0.949 (5)	0.953 (5.31)	0.943 (5.39)	0.941 (**4.82**)	0.943 (5.01)	0.944 (5.06)	0.944 (5.03)	0.942 (4.98)	0.942 (5.41)	0.953 (5.33)	0.958 (5.31)
	Tree-N	0.945 (**4.33**)	0.949 (4.37)	0.951 (4.77)	0.957 (5.25)	0.951 (7.03)	0.950 (5.4)	0.953 (4.78)	0.951 (5.89)	0.950 (5.55)	0.979 (10.5)	0.986 (32.1)
	Tree-exp	0.945 (**4.08**)	0.959 (4.39)	0.950 (4.91)	0.955 (5.12)	0.951 (6.87)	0.952 (4.78)	0.954 (4.36)	0.949 (5.49)	0.950 (5.73)	0.978 (10.5)	0.986 (31.8)

mean PI lengths. Results for the Peak Benchmark Problem presented in Figure S4 are very similar to those of Friedman Problem 1. For all sample sizes, PIBF consistently outperforms the 10 competing methods in terms of mean PI length. But this time, SPI method with LS splitting rule comes in second place.

For the H2c setup (see Figure S5 in the Supplementary Material), which is the modification of Friedman Problem 1, we can see that all methods are comparable since QRF and GRF do not over-cover. In this setting, all methods also perform fairly well with respect to PI length. Overall, the LM prediction interval method with the LS splitting rule provides slightly shorter PIs.

For the tree-based data sets (see figures S6 and S7 in the Supplementary Material), overall, it seems that the distribution of the error does not have a significant effect on the results. Again, QRF and GRF have conservative PIs. Unlike the other data sets, we see here that the mean PI lengths of the GRF method decrease very slowly as the sample size increases. For $n_{train} = 5000$, all methods (except QRF and GRF) perform similarly. For the smallest sample size, HDR PI building methods and the proposed method perform slightly better than other methods. As the sample size increases, PIBF produces the shortest prediction intervals.

Effect of calibration on the performance of prediction intervals

In this section, we investigate the effect of the proposed calibration on performance of the prediction intervals. We apply the same simulation study using the seven simulated data sets. We compare the results of the proposed method without calibration, with OOB calibration and calibration with cross-validation. The desired coverage level is set to 95%. In Figure 3, the left plot presents the mean coverages over 28 scenarios for the three variations, and the right plot shows the percentage increase in mean PI length of each of the three calibration variants across 14,000 runs (28 scenarios $\times$ 500 replications). Although we obtain the shortest prediction intervals without calibration, the variability of the mean coverage level is larger and sometimes the coverage falls below 0.94. Looking at the left plot, we can say that the variability of the mean coverage level decreases with both calibration procedures. However, applying OOB calibration provides conservative PIs. The median of the mean coverage level is more than 0.96 and the PIs with the OOB calibration are the widest. Applying calibration with CV produces slightly longer PIs than those with no calibration, but these PIs have coverage levels closer to the desired level.

In the package, both calibration procedures are implemented for the PIBF method. Simulation study results show that, compared to the OOB calibration, calibration with CV produces shorter PIs while maintaining the desired coverage level. Therefore, the default calibration procedure is set to CV in the `pibf()` function. In terms of computational time (see tables 4 and 5), calibration with k-fold CV is slower than OOB calibration since it needs to fit two additional random forests for each fold.

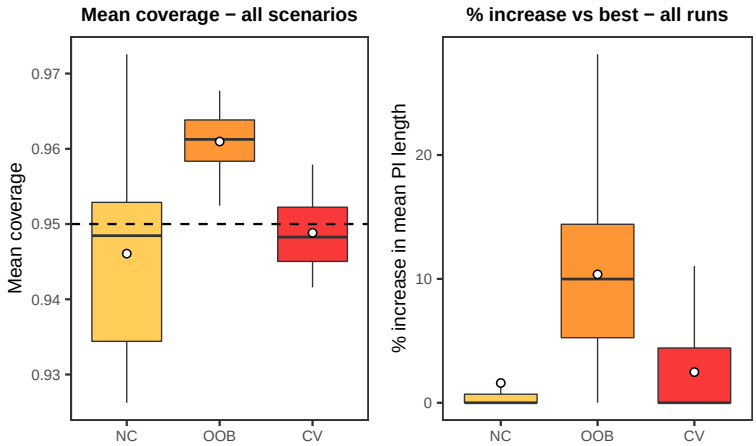

Figure 3: Global performance results of the proposed method for the simulated data sets with different calibration procedures. NC: No calibration, OOB: OOB calibration, CV: Calibration with cross-validation. *(Left)* Boxplots for the mean coverage over 500 replications across all scenarios. *(Right)* Boxplots for the percentage increase in mean PI length of each calibration procedure compared to the shortest PI length for a given run across 14,000 runs. Smaller values are better. Since outlier values are distorting the scales, they are removed.

Performance with real data sets

To further explore the performance of the prediction intervals built with the proposed method, we use 11 real data sets. Since two of the data sets have two response variables, we consider that we are analyzing 13 real data sets. Boston housing and Ames housing data sets are from the R packages **mlbench** and **AmesHousing**, respectively. The other data sets are obtained through the UCI Machine Learning Repository (Dua and Graff, 2017).

For each data set, we apply 100 times 10-fold cross-validation for each method. Hence, for each fold, the training and testing sets correspond to 90% and 10% of the whole data set, respectively. The desired coverage level is set to 95% for all methods. Table 3 presents the results of the real data analyses (n is the number of observations and p is the number of predictors) with the average coverage rate of each method over 100 repetitions, and mean prediction interval lengths averaged over 100 repetitions shown in parentheses. Figure 4 illustrates the global results of the analyses across datasets. The left plot in Figure 4 shows the mean coverages over the 13 real data sets for all methods. Similar to what we have seen with the simulated data sets, the QRF and GRF methods produce conservative prediction intervals, whereas the other methods provide a mean coverage close to the desired level. Again, the $\widehat{PI}_\alpha$ method maintains the target level on average with 0.959, but its variability is the highest among all methods. Across all data sets, there are three cases where the mean coverage is below 0.94: the proposed method for Concrete slump, and the $\widehat{PI}_\alpha$ method for Auto MPG and Computer hardware.

The right plot in Figure 4 presents the relative lengths of methods across 13 real data sets. For each method, there are 13 points in the boxplot, and each point corresponds to the percentage increase in mean PI length compared to the best method for a single real data set. Again, the prediction intervals with GRF and QRF are the widest among eleven methods. Among the other nine methods, the proposed method performs the best, followed by $\widehat{PI}_\alpha$.

For each real data set, we analyze the performance of each method through the mean PI lengths presented in figures S8 to S10 in the Supplementary Material. For the Abalone data set, the HDR method produces the shortest prediction intervals, followed by the SPI and Quant methods. While the QRF method over-covers, its PIs are no wider than those of most of the other methods. The proposed method, PIBF, is distinctly the best prediction interval method yielding the shortest PI lengths for the Air quality data set with absolute and relative humidity response variables, Airfoil selfnoise, Ames housing, Boston housing, Concrete compression, Energy efficiency data set with cooling and heating load response variables, and Servo data sets. In the Auto MPG data set, $\widehat{PI}_\alpha$ has the shortest mean

PI length but with a mean coverage of 0.929. Among the other methods, PIBF, OOB, LM, Quant and SPI methods show similarly good performances while maintaining the desired coverage level. For the Computer hardware data set, PIBF, $\widehat{PI}_\alpha$, and SPI methods perform better than the other methods. They have similar mean PI lengths. For the Concrete slump data set, the proposed method has the shortest mean PI length, but with a slightly smaller coverage of 0.939. This data set is the only one of the simulated and real data sets where the proposed method has a mean coverage below 0.94. After PIBF, $\widehat{PI}_\alpha$ and LM show a good performance with a mean coverage close to the target level. Overall, we can conclude that the proposed method shows better performance than the 10 competing methods for almost all of the real data sets.

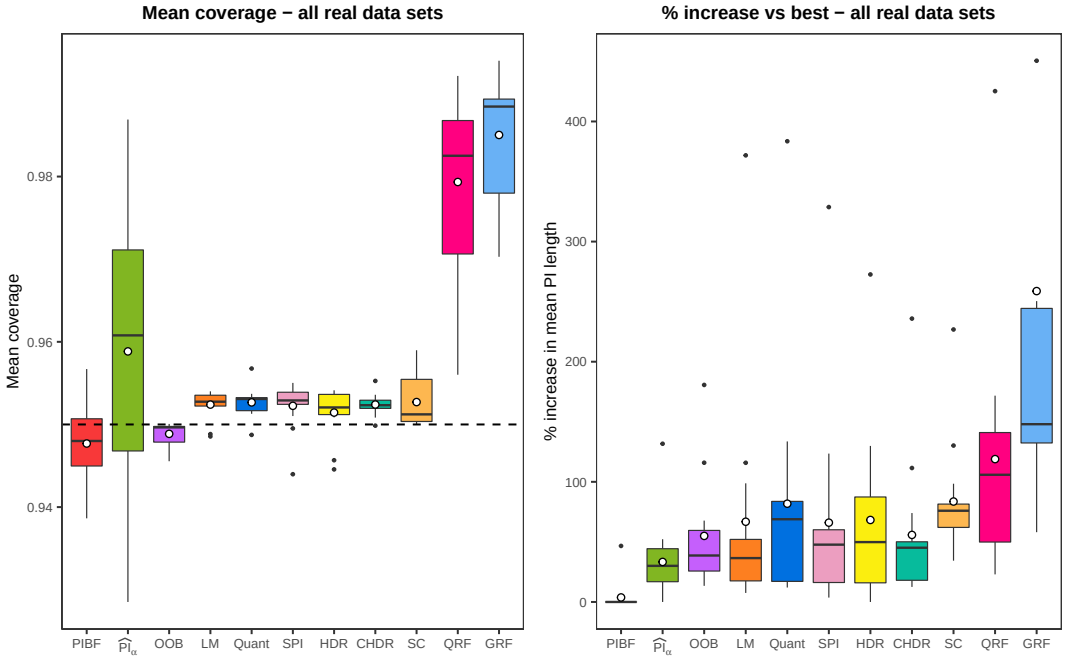

Figure 4: *(Left)* Boxplots for the mean coverage over all real data sets. All methods except the QRF and GRF methods are able to provide a mean coverage close to the desired coverage level of 0.95. Each white circle is the average of the mean coverages over 13 real data sets. *(Right)* Boxplots for the percentage increase in mean PI length of each method compared to the shortest PI length for a given real data set across 13 data sets. The smallest the percentage increase, the better the method. Each white circle is the average of the relative lengths over 13 real data sets. One of the outlier values for GRF with the percentage increase of 1244% is removed from the graph since it is distorting the scales. PIBF: Prediction intervals with boosted forests (the proposed method), $\widehat{PI}_\alpha$: Conditional α-level prediction interval, OOB: Out-of-Bag approach, LM: Classical method, Quant: Quantiles, SPI: Shortest PI, HDR: Highest density region, CHDR: Contiguous HDR, SC: Split conformal, QRF: Quantile regression forest, GRF: Generalized random forest.

Comparison of the computational times

All simulations and real data analyses were conducted in R version 3.6.0 on a Linux machine with Intel(R) Xeon(R) E5-2667 v3 @ 3.20GHz with 396 GB of memory. The average computational time of each method for the simulated and real data sets are presented in tables 4 and 5. For the proposed method (PIBF), the computational times for both calibration methods, cross-validation and OOB, are presented in the tables. We can see from the tables that, for most of the data sets, calibration with cross-validation has longer running times than OOB calibration, which is expected since with the k-fold cross-validation, we fit $2k$ more random forests than applying OOB calibration.

For the variations of Roy and Larocque (2020), since we can build prediction intervals with the five PI methods by only fitting a single random forest with a selected splitting rule, we present the total computational time for building the five variations under RFPI. To be clear, for a given splitting rule, the `rfpi()` function fits a random forest and then the set of PI methods requested by the user are applied to the output of the random forest. In our simulations, we choose to return all five PI methods for the selected splitting rule, *i.e.* when we measure the running time of the `rfpi()` function, we get the total running time of building five prediction intervals. Therefore, we should interpret the

Table 3: Results of the real data analysis. Average coverage rates of each method for each real data set, with average mean PI lengths shown in parentheses. The desired coverage level is 0.95. Shortest average mean PI lengths are shown in bold. PIBF: Prediction intervals with boosted forests (the proposed method), $\widehat{PI}_\alpha$: Conditional α-level prediction interval, OOB: Out-of-Bag approach, LM: Classical method, Quant: Quantiles, SPI: Shortest PI, HDR: Highest density region, CHDR: Contiguous HDR, SC: Split conformal, QRF: Quantile regression forest, GRF: Generalized random forest.

Data	n	p	PIBF	$\widehat{PI}_\alpha$	OOB	LM	Quant	SPI	HDR	CHDR	SC	QRF	GRF
Abalone	4177	8	0.947 (8.18)	0.946 (8.05)	0.950 (8.9)	0.949 (8.09)	0.953 (6.81)	0.951 (6.73)	0.946 (**5.58**)	0.951 (8.12)	0.950 (9.04)	0.971 (8.03)	0.978 (8.82)
Air quality (AH)	6941	13	0.948 (**0.22**)	0.966 (0.29)	0.950 (0.34)	0.954 (0.29)	0.954 (0.33)	0.954 (0.31)	0.954 (0.34)	0.954 (0.29)	0.950 (0.4)	0.992 (0.55)	0.994 (0.78)
Air quality (RH)			0.949 (**12.9**)	0.971 (18.7)	0.950 (21.6)	0.953 (19.6)	0.953 (22.5)	0.954 (20.6)	0.954 (21.1)	0.953 (19.3)	0.950 (25.5)	0.989 (35)	0.989 (44.3)
Airfoil selfnoise	1503	5	0.946 (**8.47**)	0.972 (12.9)	0.950 (13.3)	0.952 (18.3)	0.952 (19.8)	0.953 (18.9)	0.952 (19.5)	0.953 (17.9)	0.951 (14.7)	0.987 (20.4)	0.988 (18.1)
Ames housing	2929	80	0.955 (**69.3**)	0.961 (81.6)	0.950 (90.3)	0.954 (79.3)	0.953 (80.5)	0.953 (80.5)	0.952 (80.3)	0.952 (80.4)	0.951 (96.4)	0.987 (118)	0.993 (172)
Auto MPG	392	7	0.945 (10)	0.929 (**9.76**)	0.947 (11.1)	0.953 (10.5)	0.953 (10.9)	0.953 (10.5)	0.954 (13.2)	0.952 (11.5)	0.955 (13.1)	0.967 (12)	0.977 (16.1)
Boston housing	506	13	0.942 (**10.5**)	0.948 (11.2)	0.949 (12.3)	0.954 (11.7)	0.953 (11.8)	0.953 (11.4)	0.952 (12.2)	0.952 (12)	0.951 (15)	0.982 (15.7)	0.990 (25.9)
Computer hardware	209	6	0.942 (**139**)	0.939 (140)	0.946 (171)	0.952 (163)	0.951 (163)	0.950 (144)	0.951 (157)	0.950 (156)	0.957 (248)	0.965 (182)	0.985 (365)
Concrete compression	1030	8	0.948 (**14.8**)	0.957 (18)	0.950 (19.6)	0.954 (20.2)	0.953 (27.2)	0.954 (22.8)	0.953 (27.7)	0.953 (21.5)	0.950 (26)	0.982 (34.5)	0.989 (46.6)
Concrete slump	103	7	0.939 (**12.2**)	0.947 (14.2)	0.948 (15.3)	0.949 (14.9)	0.949 (20.6)	0.944 (19.4)	0.945 (16.3)	0.951 (15.4)	0.958 (22.1)	0.956 (22.3)	0.970 (28.3)
Energy efficiency (CL)	768	8	0.953 (**3.71**)	0.977 (5)	0.950 (8.01)	0.952 (7.37)	0.952 (7.9)	0.952 (7.04)	0.951 (8.03)	0.953 (6.46)	0.951 (8.54)	0.985 (8.18)	0.986 (20.4)
Energy efficiency (HL)			0.951 (**1.48**)	0.987 (3.43)	0.950 (4.16)	0.953 (6.99)	0.952 (7.17)	0.954 (6.35)	0.951 (5.52)	0.952 (4.98)	0.952 (4.84)	0.988 (7.79)	0.989 (19.9)
Servo	167	4	0.957 (**18.4**)	0.966 (24.2)	0.948 (25.5)	0.953 (26.6)	0.957 (32.9)	0.955 (27.2)	0.954 (28.2)	0.955 (26.8)	0.959 (30.4)	0.983 (37.9)	0.977 (42.9)

values for RFPI with care while comparing the computational times of the methods. Although not all PI methods have similar computational complexities, we can say that even the average time of building prediction intervals with one of these variations, assuming they have similar running times, is reasonable. Since, for the HDR-based PI methods, an optimal bandwidth has to be chosen, which is a time-consuming process, among the five PI methods, the slowest ones are the HDR and CHDR. From the remaining variations, the classical method, LM, is the fastest, followed by the Quant and SPI methods.

For both the simulations and real data analyses, the OOB and GRF methods have the smallest running times. For most of the methods, the increase in the sample size has a mild effect on the ratio of increase in running times. However, for the split conformal method with the simulated data sets, running times increase more than the proportional increase in sample sizes. Similarly, we can see that the QRF method is also affected from the training sample size as it rises to 5000.

Concluding remarks

In this paper, we have introduced an R package named **RFpredInterval**. This package implements 16 methods to build prediction intervals with random forests: a new method to build **P**rediction **I**ntervals with **B**oosted **F**orests (PIBF) and 15 different variations to produce prediction intervals with random forests proposed by Roy and Larocque (2020). PIBF provides bias-corrected point predictions obtained with the one-step boosted forest and prediction intervals by using the nearest neighbour out-of-bag observations to estimate the conditional prediction error distribution.

We performed an extensive simulation study with a variety of simulated data sets and applied real data analyses to investigate the performance of the proposed method. The performance was evaluated based on the coverage level and length of the prediction intervals. We compared the performance of the proposed method to 10 existing methods for building prediction intervals with random forests. The proposed method was able to maintain the desired coverage level with both the simulated and real data sets. In terms of the PI lengths, globally, the proposed method provided the shortest prediction intervals among all methods. The conclusions drawn from the analysis of real data sets were very similar to those with the simulated data sets. This provides evidence for the reliability of the proposed method. All results obtained indicate that the proposed method can be used with confidence for a variety of regression problems.

Note that the coverage rate of prediction intervals for new observations can have several interpretations. An interesting discussion about this issue is given in Mayr et al. (2012). In that paper, the authors presented two interpretation of coverage: sample coverage and conditional coverage. Sample coverage means that if we draw a new sample from the same population as the training sample and build PIs with a desired coverage level of $(1-\alpha)$, then the global coverage rate over this sample will be $(1-\alpha)$. The conditional coverage means that if we sample many new observations always having the same set of covariates and build PIs for them with a desired coverage level of $(1-\alpha)$, then about $(1-\alpha)\,100\%$ of these prediction intervals will contain the true value of the response. To hold a desired level of conditional coverage, the predictive method needs to provide the desired coverage level over the entire covariate space. On the other hand, sample coverage needs only maintain the desired coverage level over the new sample, on average. Therefore, if the conditional coverage holds, then the sample coverage also holds. In practice, predictive models are mostly evaluated with their

Table 4: Average computational time (in seconds) of each method over 500 replications for each simulated data set. The values represent the average time to build prediction intervals for a test set with 1000 observations. PIBF-CV: The proposed method with the cross-validation calibration, PIBF-OOB: The proposed method with the OOB calibration, RFPI: The average total running time of the five PI methods, *i.e.* LM + Quant + SPI + HDR + CHDR.

n_{train}	Data	PIBF-CV	PIBF-OOB	$\widehat{PI}_\alpha$	OOB	RFPI	SC	QRF	GRF
200	Friedman 1	21.36	13.44	9.62	0.25	87.79	0.61	3.05	0.27
	Friedman 2	21.83	13.96	9.62	0.23	59.50	0.44	2.61	0.26
	Friedman 3	28.84	16.03	11.46	0.20	75.47	0.41	2.50	0.22
	Peak	18.76	10.95	11.91	0.28	73.15	0.89	4.05	0.43
	H2c	12.92	9.83	7.99	0.27	36.94	0.79	4.05	0.30
	Tree-N	22.75	21.17	12.21	0.23	100.02	0.50	2.81	0.26
	Tree-exp	16.28	15.62	15.65	0.23	87.49	0.53	2.85	0.25
500	Friedman 1	38.37	14.63	8.50	0.44	96.18	2.45	9.29	0.66
	Friedman 2	27.07	13.64	7.64	0.43	84.19	1.80	6.79	0.47
	Friedman 3	18.85	17.49	7.98	0.47	70.78	1.86	4.96	0.45
	Peak	29.16	18.14	9.72	0.70	212.23	2.29	7.49	1.26
	H2c	14.84	11.29	11.20	0.44	58.60	1.74	4.95	0.78
	Tree-N	17.80	20.31	7.70	0.51	183.92	2.21	4.28	0.51
	Tree-exp	18.05	19.05	7.74	0.51	170.24	1.72	4.45	0.49
1000	Friedman 1	32.92	18.07	12.11	0.64	181.30	3.35	9.54	1.50
	Friedman 2	23.07	15.94	10.25	0.45	135.64	2.14	6.26	0.90
	Friedman 3	23.24	16.07	7.37	0.44	96.76	2.27	6.40	0.69
	Peak	51.93	33.12	7.95	0.83	374.89	5.35	19.71	1.65
	H2c	26.81	12.56	8.07	0.59	80.38	3.68	13.11	0.87
	Tree-N	22.84	16.70	7.11	0.47	288.35	2.72	9.53	0.78
	Tree-exp	21.06	17.26	7.03	0.48	272.52	2.62	10.38	0.69
5000	Friedman 1	155.81	139.88	28.70	3.72	928.84	40.79	134.97	4.84
	Friedman 2	155.62	101.58	35.25	2.30	430.80	33.43	60.99	2.65
	Friedman 3	106.70	91.72	34.41	2.36	330.81	28.47	73.18	2.72
	Peak	287.67	245.17	22.76	6.84	1914.56	57.96	123.19	11.80
	H2c	283.53	107.57	24.12	4.31	271.18	42.69	79.19	5.01
	Tree-N	106.17	71.78	22.97	3.19	753.32	26.88	74.81	3.86
	Tree-exp	95.85	68.90	21.76	3.32	779.42	26.09	64.02	3.85

Table 5: Average computational time (in seconds) of each method over 100 times 10-fold cross-validation for each real data set. PIBF-CV: The proposed method with the cross-validation calibration, PIBF-OOB: The proposed method with the OOB calibration, RFPI: The average total running time of the five PI methods, *i.e.* LM + Quant + SPI + HDR + CHDR.

Data	n	p	PIBF-CV	PIBF-OOB	$\widehat{PI}_\alpha$	OOB	RFPI	SC	QRF	GRF
Abalone	4177	8	102.37	64.39	17.65	5.20	422.10	20.42	45.39	6.32
Air quality (AH)	6941	13	383.60	126.95	26.63	11.55	1256.56	45.45	126.36	16.07
Air quality (RH)			383.53	124.71	26.60	11.55	1111.16	46.13	127.02	15.95
Airfoil selfnoise	1503	5	256.92	16.61	3.57	0.31	271.35	1.80	3.65	0.65
Ames housing	2929	80	195.07	28.56	15.64	8.73	333.28	42.95	226.49	11.34
Auto MPG	392	7	11.02	7.08	1.82	0.35	47.84	0.58	1.73	0.40
Boston housing	506	13	14.77	8.89	2.32	0.49	63.20	1.31	3.63	0.70
Computer hardware	209	6	6.53	3.80	0.93	0.22	26.49	0.28	0.98	0.24
Concrete compression	1030	8	28.28	17.80	3.38	0.38	257.19	2.03	5.86	0.66
Concrete slump	103	7	9.52	2.10	0.65	0.14	26.60	0.13	0.62	0.11
Energy efficiency (CL)	768	8	85.97	15.93	2.51	0.34	341.73	0.94	2.39	0.52
Energy efficiency (HL)			99.95	20.31	2.47	0.34	380.97	0.92	2.42	0.42
Servo	167	4	12.05	3.99	0.72	0.15	53.14	0.12	0.65	0.15

global predictive performance. Hence, ensuring that the sample coverage level is achieved should be sufficient for most applications. The proposed calibration method with cross-validation is designed to ensure the sample coverage property. From the simulation study and real data analyses, we can see that the sample coverage is attained with the proposed calibration method.

Availability

The package is available from CRAN at `https://cran.r-project.org/package=RFpredInterval`. The development version is available at `https://github.com/calakus/RFpredInterval`.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by Fondation HEC Montréal.

Bibliography

C. Alakus, D. Larocque, and A. Labbe. *RFpredInterval: Prediction Intervals with Random Forests and Boosted Forests*, 2022. URL https://cran.r-project.org/package=RFpredInterval. R package version 1.0.5. [p306]

C. Alakuş, D. Larocque, S. Jacquemont, F. Barlaam, C.-O. Martin, K. Agbogba, S. Lippé, and A. Labbe. Conditional canonical correlation estimation based on covariates with random forests. *Bioinformatics*, 2021. URL https://doi.org/10.1093/bioinformatics/btab158. [p304, 307]

S. Athey, J. Tibshirani, and S. Wager. Generalized random forests. *The Annals of Statistics*, 47(2): 1148–1178, Apr. 2019. URL https://doi.org/10.1214/18-AOS1709. [p304, 305, 314]

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug. 1996. URL https://doi.org/10.1007/BF00058655. [p312]

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct. 2001. URL https://doi.org/10.1023/A:1010933404324. [p304, 305]

L. Breiman and L. Breiman. *Classification and regression trees*. The Wadsworth & Brooks/Cole statistics/probability series. Wadsworth International Group, Belmont, Calif., 1984. ISBN 0-534-98054-6 978-0-534-98054-2. [p305]

D. De Cock. Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project. *Journal of Statistics Education*, 19(3), Nov. 2011. URL https://doi.org/10.1080/10691898.2011.11889627. [p304, 309]

D. Dua and C. Graff. UCI machine learning repository, 2017. URL https://archive.ics.uci.edu/ml. [p317]

I. Ghosal and G. Hooker. Boosting Random Forests to Reduce Bias; One-Step Boosted Forest and Its Variance Estimate. *Journal of Computational and Graphical Statistics*, 30(2):493–502, Apr. 2021. URL https://doi.org/10.1080/10618600.2020.1820345. [p305, 306]

T. Hothorn and A. Zeileis. Predictive Distribution Modeling Using Transformation Forests. *Journal of Computational and Graphical Statistics*, pages 1–16, Jan. 2021. URL https://doi.org/10.1080/10618600.2021.1872581. [p305, 313]

T. Hothorn, B. Lausen, A. Benner, and M. Radespiel-Tröger. Bagging survival trees. *Statistics in Medicine*, 23(1):77–91, 2004. URL https://doi.org/10.1002/sim.1593. [p304, 305, 307]

H. Ishwaran and U. B. Kogalur. *randomForestSRC: Fast Unified Random Forests for Survival, Regression, and Classification (RF-SRC)*, 2021. URL https://cran.r-project.org/package=randomForestSRC. R package version 2.11.0. [p308]

Jerome H. Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1–67, Mar. 1991. URL https://doi.org/10.1214/aos/1176347963. [p312]

M. Kuhn. *AmesHousing: The Ames Iowa Housing Data*, 2020. URL https://cran.r-project.org/package=AmesHousing. R package version 0.0.4. [p309]

J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman. Distribution-Free Predictive Inference for Regression. *Journal of the American Statistical Association*, 113(523):1094–1111, July 2018. URL https://doi.org/10.1080/01621459.2017.1307116. [p305]

F. Leisch and E. Dimitriadou. *mlbench: Machine Learning Benchmark Problems*, 2021. URL https://cran.r-project.org/package=mlbench. R package version 2.1-3. [p312]

Y. Lin and Y. Jeon. Random Forests and Adaptive Nearest Neighbors. *Journal of the American Statistical Association*, 101(474):578–590, June 2006. URL https://doi.org/10.1198/016214505000001230. [p304, 307]

B. Lu and J. Hardin. *forestError: A Unified Framework for Random Forest Prediction Error Estimation*, 2020. URL https://cran.r-project.org/package=forestError. R package version 0.2.0. [p306]

B. Lu and J. Hardin. A Unified Framework for Random Forest Prediction Error Estimation. *Journal of Machine Learning Research*, 22(8):1–41, 2021. URL https://jmlr.org/papers/v22/18-558.html. [p306, 307]

A. Mayr, T. Hothorn, and N. Fenske. Prediction intervals for future BMI values of individual children - a non-parametric approach by quantile boosting. *BMC Medical Research Methodology*, 12(1):6, Jan. 2012. URL https://doi.org/10.1186/1471-2288-12-6. [p319]

N. Meinshausen. Quantile Regression Forests. *Journal of Machine Learning Research*, 7(35):983–999, 2006. URL https://jmlr.org/papers/v7/meinshausen06a.html. [p305]

N. Meinshausen. *quantregForest: Quantile Regression Forests*, 2017. URL https://cran.r-project.org/package=quantregForest. R package version 1.3-7. [p305]

L. Mentch and G. Hooker. Quantifying Uncertainty in Random Forests via Confidence Intervals and Hypothesis Tests. *Journal of Machine Learning Research*, 17(26):1–41, 2016. URL https://jmlr.org/papers/v17/14-168.html. [p305]

H. Moradian, D. Larocque, and F. Bellavance. L1 splitting rules in survival forests. *Lifetime Data Analysis*, 23(4):671–691, Oct. 2017. URL https://doi.org/10.1007/s10985-016-9372-1. [p304, 305, 307]

H. Moradian, D. Larocque, and F. Bellavance. Survival forests for data with dependent censoring. *Statistical Methods in Medical Research*, 28(2):445–461, Feb. 2019. URL https://doi.org/10.1177/0962280217727314. [p304, 305, 307]

M.-H. Roy and D. Larocque. Prediction intervals with random forests. *Statistical Methods in Medical Research*, 29(1):205–229, Jan. 2020. URL https://doi.org/10.1177/0962280219829885. [p304, 305, 306, 307, 308, 309, 310, 313, 314, 315, 318, 319]

S. Tabib and D. Larocque. Non-parametric individual treatment effect estimation for survival data with random forests. *Bioinformatics*, 36(2):629–636, Jan. 2020. URL https://doi.org/10.1093/bioinformatics/btz602. [p304, 307]

J. Tibshirani, S. Athey, E. Sverdrup, and S. Wager. *grf: Generalized Random Forests*, 2021. URL https://cran.r-project.org/package=grf. R package version 2.0.2. [p305]

R. Tibshirani. *conformalInference: Tools for conformal inference in regression*, 2019. URL https://github.com/ryantibs/conformal. R package version 1.1. [p305]

V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic Learning in a Random World*. Springer Science & Business Media, Dec. 2005. ISBN 978-0-387-25061-8. [p305]

V. Vovk, I. Nouretdinov, and A. Gammerman. On-line predictive linear regression. *The Annals of Statistics*, 37(3):1566–1590, June 2009. URL https://doi.org/10.1214/08-AOS622. [p305]

S. Wager and S. Athey. Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *Journal of the American Statistical Association*, 113(523):1228–1242, July 2018. URL https://doi.org/10.1080/01621459.2017.1319839. [p305]

M. N. Wright, S. Wager, and P. Probst. *ranger: A Fast Implementation of Random Forests*, 2020. URL https://cran.r-project.org/package=ranger. R package version 0.12.1. [p308]

G. Zhang and Y. Lu. Bias-corrected random forests in regression. *Journal of Applied Statistics*, 39(1):151–160, Jan. 2012. URL https://doi.org/10.1080/02664763.2011.578621. [p305]

H. Zhang. *rfinterval: Predictive Inference for Random Forests*, 2019. URL https://cran.r-project.org/package=rfinterval. R package version 1.0.0. [p306]

H. Zhang, J. Zimmerman, D. Nettleton, and D. J. Nordman. Random Forest Prediction Intervals. *The American Statistician*, 74(4):392–406, Oct. 2020. URL https://doi.org/10.1080/00031305.2019.1585288. [p305, 306]

Cansu Alakus
Department of Decision Sciences
HEC Montréal
Canada
(ORCiD: 0000-0001-7669-5863)
cansu.alakus@hec.ca

Denis Larocque
Department of Decision Sciences
HEC Montréal
Canada
(ORCiD: 0000-0002-7372-7943)
denis.larocque@hec.ca

Aurélie Labbe
Department of Decision Sciences
HEC Montréal
Canada
aurelie.labbe@hec.ca