Regression with Missing Data, a Comparison Study of Techniques Based on Random Forests

Irving Gómez-Méndez[a] and Emilien Joly[b]

[a]Centro de Investigación en Matemáticas, AC (CIMAT); [b]Centro de Investigación en Matemáticas, AC (CIMAT)

ARTICLE HISTORY
Compiled August 13, 2022

ABSTRACT
In this paper we present the practical benefits of a new random forest algorithm to deal with missing values in the sample. The purpose of this work is to compare the different solutions to deal with missing values using random forests and describe the new algorithm performance as well as its algorithmic complexity. A variety of data-missing mechanisms (MCAR, MAR, MNAR) are considered and simulated. We study the quadratic errors and the bias of our algorithm and compare it to the most popular missing values random forests algorithms in the literature. In particular, we compare those techniques for both a regression and prediction purpose. This work follows the paper of [1] on the consistency of this new algorithm.

KEYWORDS
Missing values, Random forests, Non-parametric regression, Prediction with missing values

1. General introduction

1.1. Random forests with missing values: a contemporary challenge

Random forests and recursive trees are widely used in applied statistics and computer science. The popularity of recursive trees relies on several factors: their easy interpretability, the fact that they can be used for both regression and classification tasks, the small number of hyper-parameters to be tuned and finally, their non-parametric nature that allows their use to infer arbitrarily complex relations between the input and the output space. A random forest combines several randomized trees, improving the prediction accuracy at a cost of a slight lost in interpretation. This technique is easily parallelizable which has made it one of the most popular tools for handling high dimensional data sets. It has been successfully involved in various practical problems, including chemioinformatics, ecology, 3D object recognition, bioinformatics and econometrics. [2] present a detailed list of applications as well as a review on random forests. In the present work we have focused on the ability of random forests to deal with missing values.

Contact Author: Irving Gómez-Méndez. Email: irving.gomez@cimat.mx

Three groups of algorithms. Approaches to handle missing values regarding the use of random forests could be classified in three groups. In the *first group*, we classify the algorithms that impute the missing values without using ad-hoc techniques, like k nearest neighbor imputation (*knn*-imputation) [3] or multiple imputation chained equations (MICE) [4], for example. Once performed this imputation step, the regular random forest algorithm is implemented on the completed data set. In this paper, we will not pay much attention to this first group of algorithms as the imputation step is unrelated with the later use of the random forest algorithm which is finally, only treated as a black box. The *second group* is composed of the algorithms that use the random forest structure (directly or through some extra method like proximity matrices) to impute the missing values. They typically use iterated updates of the imputations to refine them. We decided to store the methods proposed by [5–7] in this group. The *third group* consists in built-in methodologies that include the incomplete observations directly in the construction of the recursive trees (without imputation steps) to compute the random forest's estimation. Are included in this group: surrogate splits [8], missing incorporated in attributes (MIA) [9] or the algorithm proposed by [1], which is the main focus of this paper.

1.2. Literature on missing values treatment with random forests

Handling missing values through different algorithms has received much attention recently with several simulation studies comparing the performance of distinct methods [10–14]. In this paper, we propose a comparison study of techniques of the last two groups since they appear to be the most used and precise in practice. Of special focus for this present paper is the study of the practical performance of the new approach presented in [1], which has the advantage to be consistent under an MCAR mechanism and a generalized additive model.

The idea of handling missing values with random forests algorithms is not new whether it be for a "filling the gaps" task, a learning or a prediction task with corrupted data. In this section, we describe the current state-of-the-art in the study of this problem. The study developed by [10] is one of the first to present a result on missing values using recursive trees. This work compares the error rate between surrogate splits in a single decision tree, and the imputation procedure presented by [15,16]. Two data sets are considered in the study, given by the so-called *Waveform* data set (presented by [8]) and the *Prima Indian data sets* (available at the UCI machine learning repository). The percentage of missing values varies between 10% and 45% for the first data set, where the missing values are introduced accordingly to an MCAR mechanism. On the other hand, 10% of the observations present missing values in the second data set. In [10] the author concludes that the imputation procedure yields significantly less missclassification error rate.

The work of [11] presents a comparison of classification methods like support vector machines, *knn*-imputation and the decision tree method C4.5 [17] applied to missing data, and imputation algorithms, including single imputation and MICE. In total 15 data sets are considered, inducing missing values with up to 50% of the observations per variable being missing. The authors conclude that the application of MICE leads to better results in most of the instances.

In [12] the authors compare surrogate splits introduced in the conditional inference forests [18] with *knn*-imputation, with no clear advantage for either of the methods. This study considers classification and regression problems with three different cor-

relation structures and seven schemes for missing values. However, the percentage of missing values is kept constant.

In [13] there is presented a comparison study using trees built with the CART criterion, conditional inference trees and their corresponding random forests, focusing on surrogate splits to handle missing values and the use of MICE to impute missing values. The authors consider 12 real life data sets, half of them for regression and the other half for classification, 8 of the data sets already present missing values while in the rest 4 data sets missing values are induced. Arguing that [12] found similar results for MCAR and MAR mechanisms, the missing values are solely introduced according to an MCAR missing-data mechanism, considering missing rates between 0% (benchmark) to 40%. Their results do not show a clear improvement by using multiple imputation, with MICE even producing inferior results when missing values are limited in number and are not arbitrary spread across the data. The results also show a similar result between trees constructed with the CART criterion and conditional inference trees.

The simulation developed by [14] studies the performance of several methods to handle missing values in regression tasks. In this study, three different regression functions are considered, one of them being linear, one quadratic and the third being the so-called "friedman1" [19]. They consider the MCAR mechanism and censoring, inducing up to 20% of observations with missing values in the first variable. The authors compare conditional inference trees, trees and random forests based on the CART criterion and XGBoost [20]. The algorithms to handle missing values include MIA, surrogate splits for both trees built with the CART criterion and conditional inference trees, block propagation (only implemented in XGBoost), surrogate splits, mean-imputation and EM imputation [21]. The authors clearly favors the usage of MIA for tree-based methods, while block propagation could also be a good method.

In [5] an algorithm based on random forests to impute the missing values is proposed making use of the observed values and the proximity matrix. [6] presents an improvement which instead of considering only the observed values in the imputation procedure, it considers nearest neighbors for continuous variables and all the observations for categorical variables. [6] compares these two approaches and *knn*-imputation introducing missing values completely at random in the *Spam* data set and considering missing data rates from 5% to 60%, concluding that the proposed approach outperforms *knn*-imputation and the previous method proposed by [5]. However, the same author comments that other missing-data mechanisms should be considered.

The missForest algorithm is introduced in [7], this algorithm imputes the missing values iteratively considering it as a regression problem in which the imputation of the current variable is done using all the other variables. In the simulation study presented, they consider classification and regression problems in 7 different data sets where 10%, 20% or 30% of the values are removed completely at random, concluding that missForest outperforms *knn*-imputation and MICE. Furthermore, the authors ensure that the full potential of missForest is deployed when data includes interactions or non-linear relations between variables of unequal scales and different types.

In this paper, we compare the performance of most of these techniques with a new proposal taken from [1] when we let the percentage of missing values to vary from 0% to 95%.

The rest of the paper is organized as follows. In Section 2, we give the general background on random forests algorithms. We formally introduce the CART criterion and the missing-data mechanisms. In Section 3, we describe the proposal to build recursive trees including the missing values and we give upper bounds on its computational complexity. In Section 4, we describe the other methods that we use as benchmark in

the simulation study explained in detail in Section 4.2 and Section 4.3. In Section 5, we present and describe the results. Finally, Section 6 presents the conclusions.

2. Settling the concepts

2.1. Random forests built upon the CART criterion

Throughout this article, we assume to have access to a training data set $\mathcal{D}_n = (\mathbf{X}_i, Y_i)_{i=1,\dots,n}$ where the response variables Y_i are real-valued and the input variables $\mathbf{X}_i$ belong to some space $\mathcal{X} \subseteq \mathbb{R}^p$. The objective is to use the data $\mathcal{D}_n$ to construct a learning model, also called learner, predictor or estimator, $m_n : \mathcal{X} \to \mathbb{R}$ that estimates the regression function $m(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$.

The random forest is made of a set of regression trees that are later aggregated all together with a simple mean idea. Each branch of the tree will be random (in its construction process) and represents a partition of the input space in smaller regions. Moving along the path of the tree corresponds to a choice of one of the possible regions. To construct this partition of the input space, the trees are built in a recursive way (hence the name of recursive trees). The root of the tree corresponds to the whole input space $\mathcal{X}$. Then, recursively, a region is chosen and is split into two smaller regions. This process is continued until some stopping rule is applied. At each step of the tree construction, the partition performed over a cell (or equivalently its corresponding node) is determined by maximizing some split-criterion. The present work focuses in the so-called CART split criterion. We first introduce some important notations.

- A denotes a general node (or cell).
- $N(A)$ holds for the number of points in A.
- The notation $d = (h, z)$ denotes a cut in A, where
  h is a direction, $h \in \{1, \dots, p\}$, and
  z is the position of the cut in the hth direction, between the limits of A.
- $\mathcal{C}_A$ is the set of all possible cuts in node A.
- A cell A is split into two cells denoted $A_L = \{\mathbf{x} \in A : \mathbf{x}^{(h)} < z\}$ and $A_R = \{\mathbf{x} \in A : \mathbf{x}^{(h)} \geq z\}$.
- $\bar{Y}_A$ (resp. $\bar{Y}_{A_L}$, $\bar{Y}_{A_R}$) is the empirical mean of the response variable Y_i for the indexes such that $\mathbf{X}_i$ belongs to the cell A (resp. A_L, A_R).

Then, the CART split criterion for a generic cell A is defined as

$$\begin{aligned} L_n(A, d) =& \frac{1}{N(A)} \sum_{i=1}^{n} \left(Y_i - \bar{Y}_A\right)^2 \mathbb{1}_{\mathbf{X}_i \in A} \\ & - \frac{1}{N(A)} \sum_{i=1}^{n} \left(Y_i - \bar{Y}_{A_L} \mathbb{1}_{\mathbf{X}_i^{(h)} < z} - \bar{Y}_{A_R} \mathbb{1}_{\mathbf{X}_i^{(h)} \geq z}\right)^2 \mathbb{1}_{\mathbf{X}_i \in A} \end{aligned} \tag{1}$$

with the convention $0/0 = 0$.

As mentioned above, a random forest is a predictor consisting of $M(> 1)$ randomized trees. The randomization is introduced in two different parts of the tree construction. Prior to the construction of each tree, a_n observations are extracted at random with (or without) replacement from the learning data set $\mathcal{D}_n$. Only these a_n observations are taken into account in the tree construction. Then, at each cell a split (or cut) is performed by maximizing the split criterion over a number `mtry` of directions h, chosen

uniformly at random. The tree construction is stopped when each final node contains less or equal than `nodesize` points. Hence, the parameters of this algorithm are:

- $M > 1$, which is the number of trees in the forest.
- $a_n \in \{1, \ldots, n\}$, which is the number of observations in each tree.
- `mtry` $\in \{1, \ldots, p\}$, which is the number of directions (features) chosen, candidates to be split. We denote by $\mathcal{M}_{try}$ the features selected in each step.
- `nodesize` $\in \{1, \ldots, a_n\}$, which is the maximum number of observations for a node to be a final cell.

The randomization introduced in the trees (independent from the original source of randomness in the sample $\mathcal{D}_n$) is represented in a symbolic random variable Θ. To each tree – randomized with the random variable Θ_k – there is associated a predicted value at a query point $\mathbf{x}$, denoted as $m_n(\mathbf{x}; \Theta_k)$. The different trees are constructed by the same procedure but with independent randomization, so the random variables $\Theta_1, \ldots, \Theta_M$ are i.i.d. with common law Θ. In our choice of the construction rules, Θ consists in the observations selected for the tree and the candidate variables to split at each step. Finally, the kth tree's estimation at point $\mathbf{x}$ is defined as

$$m_n(\mathbf{x}; \Theta_k) = \sum_{i \in \mathcal{I}_{n,\Theta_k}} \frac{Y_i \mathbb{1}_{\mathbf{X}_i \in A_n(\mathbf{x}; \Theta_k)}}{N(A_n(\mathbf{x}; \Theta_k))}$$

where $\mathcal{I}_{n,\Theta_k}$ is the set of the a_n observations selected prior to the construction of the kth tree, $A_n(\mathbf{x}; \Theta_k)$ is the unique final cell that contains $\mathbf{x}$, and $N(A_n(\mathbf{x}; \Theta_k))$ is the number of observations which belong to the cell $A_n(\mathbf{x}; \Theta_k)$. The average of the trees forms the random forest's estimation given by

$$m_{M,n}(\mathbf{x}; \Theta_1, \ldots, \Theta_M) = \frac{1}{M} \sum_{k=1}^{M} m_n(\mathbf{x}; \Theta_k).$$

It is known from the work of [22] that the random forest does not overfit when M tends to infinity. This makes the parameter M only restricted by computational power.

2.2. Missing-data mechanisms

The concept of missing-data mechanism (introduced by [23]) establishes the relationship between missingness and data. Before introducing the missing-data mechanisms, let us define a new variable, called the missing-data indicator

$$\mathbf{M}^{(h)} = \begin{cases} 1 & \text{if } \mathbf{X}^{(h)} \text{ is missing} \\ 0 & \text{otherwise} \end{cases}, \quad 1 \leq h \leq p.$$

We assume throughout this work that the response Y has no missing values which makes unnecessary to define an indicator of missing variable for Y. Then, the mechanisms are fully characterized by the information of the conditional distribution of $\mathbf{M}^{(h)}$ given $(\mathbf{X}, Y)$. There are three possible missing-data mechanisms.

Missing Completely at Random (MCAR). We say that data are MCAR if $\mathbf{M}^{(h)}$ is independent from $(\mathbf{X}, Y)$. In other words, under the MCAR assumption, the coordinates $\mathbf{X}^{(h)}$ have some probability to be missing in the sample and this probability does not depend on the value of $\mathbf{X}$ nor the response variable Y.

Missing at Random (MAR). Data are say to be MAR if the probability of missingness is related to some measured variables but not to the missing values.

Missing Not at Random (MNAR). If the probability of missingness depends on missing values we say that the data are MNAR.

3. A new approach

It is of special interest to study the performance of the algorithm presented by [1] who have proven its consistency for an MCAR mechanism and a generalized additive model, being one of the first results on the consistency of random forests with missing values. This algorithm adapts the original CART criterion to manage missing data directly in the construction of the regression trees. We now recall this proposal.

3.1. Description of the algorithm

Assume for now that we have a partition of the input space. When there are missing values in the data set, there is uncertainty on the region to which each observation belongs. Thus, the original CART criterion (see Equation (1)) becomes intractable since the quantities $N(A)$, $N(A_L)$, $N(A_R)$, $\bar{Y}_A$, $\bar{Y}_{A_L}$, $\bar{Y}_{A_R}$, $\mathbb{1}_{\mathbf{X}_i \in A}$, $\mathbb{1}_{\mathbf{X}_i^{(h)} < z}$ and $\mathbb{1}_{\mathbf{X}_i^{(h)} \geq z}$ can not be computed. The proposed approach keeps the form of the CART criterion and makes use of adapted imputations for the intractable parts which allows the computation of a modified version of the CART criterion. Unlike most of the imputation techniques, the imputation step is not performed independently of the evaluation of the CART criterion but is integrated to its later optimization. While a cut is selected by maximizing the original CART criterion, now a couple (cut, imputation) is chosen at each split in the creation of the random tree. The idea is that, for a cut, the observations with missing values are assigned to the child node that maximizes this modified CART criterion. At the end, the missing observations will belong to a final node of the tree, which can give an "imputation" of the missing values as a region of the input space, which in turn would be translated into a "cloud" of possible regions for the missing values when the random forest is considered. For sake of clarity the proposal has been divided in two parts, Algorithm 1 describes the steps for the construction of the random forest with missing entries and Algorithm 2 establishes the steps to find the best cut and assignation of missing data. At the beginning of the construction of each tree it is necessary to initialize the list of current not-final cells $\mathcal{P} = \{\mathcal{X}\}$ as well as the induced partitions of the input variables and the target variable $\mathcal{X}_\mathcal{P} = \{(\mathbf{X}_1, \ldots, \mathbf{X}_n)\}$ and $\mathcal{Y}_\mathcal{P} = \{(Y_1, \ldots, Y_n)\}$, respectively. Along with the initialization of these sets, the final partition of the input space $\mathcal{P}_f = \{\}$ and the corresponding partitions of the input variables $\mathcal{X}_f = \{\}$ and the target variable $\mathcal{Y}_f = \{\}$ are initialized. When a cell A satisfies the criteria to be a final cell, it is removed from $\mathcal{P}$ and added to $\mathcal{P}_f$, the input variables $\mathcal{X}_A$ and the target variables $\mathcal{Y}_A$ belonging to A are also removed from $\mathcal{X}_\mathcal{P}$ and $\mathcal{Y}_\mathcal{P}$, and added to $\mathcal{X}_f$ and $\mathcal{Y}_f$, respectively. In the

sequel, we denote as $N_{obs}^{(h)}(A)$ the number of points belonging to A, whose value in the direction h has been observed and $N_{miss}^{(h)}(A)$ the number of observations assigned to cell A whose value in the direction h is missing.

Algorithm 1 Random forest with assignation of missing entries.

Input: Training sample $\mathcal{D}_n$, number of trees $M > 1$, `mtry`$\in \{1, \dots, p\}$, $a_n \in \{1, \dots, n\}$, `nodesize`$\in \{1, \dots, a_n\}$.

Output: Random forest $m_{M,n}$.

for $i = 1, \dots, M$ **do**
  Select a_n points uniformly in $\mathcal{D}_n$.
  Initialize $\mathcal{P}$, $\mathcal{X}_\mathcal{P}$, $\mathcal{Y}_\mathcal{P}$, $\mathcal{P}_f$, $\mathcal{X}_f$ and $\mathcal{Y}_f$.
  while $\mathcal{P} \neq \varnothing$ **do**
    Let A, $\mathcal{X}_A$, $\mathcal{Y}_A$ be the first elements of $\mathcal{P}$, $\mathcal{X}_\mathcal{P}$ and $\mathcal{Y}_\mathcal{P}$, resp.
    Set $N(A)$ the number of points which belong or were assigned to A.
    if $N(A) \leq$`nodesize` **then**
      Remove A, $\mathcal{X}_A$ and $\mathcal{Y}_A$ from $\mathcal{P}$, $\mathcal{X}_\mathcal{P}$ and $\mathcal{Y}_\mathcal{P}$, and add them to $\mathcal{P}_f$, $\mathcal{X}_f$ and $\mathcal{Y}_f$.
    else
      for $j = 1, \dots, p$ **do**
        Compute $N_{obs}^{(j)}(A)$.
      end for
      Let h_{obs} be the features h such that $N_{obs}^{(h)}(A) > 1$.
      if $h_{obs} = \varnothing$ **then**
        Remove A, $\mathcal{X}_A$ and $\mathcal{Y}_A$ from $\mathcal{P}$, $\mathcal{X}_\mathcal{P}$ and $\mathcal{Y}_\mathcal{P}$, and add them to $\mathcal{P}_f$, $\mathcal{X}_f$ and $\mathcal{Y}_f$.
      else
        Set $m_{obs} = |h_{obs}|$.
        if $m_{obs} \leq$`mtry` **then**
          Set $\mathcal{M}_{try} = h_{obs}$.
        else
          Select uniformly, without replacement, a subset $\mathcal{M}_{try} \subset h_{obs}$ of cardinality `mtry`.
        end if
        Apply Algorithm 2 on cell A along the features in $\mathcal{M}_{try}$.
        Call A_L and A_R the two resulting cells.
        Remove A, $\mathcal{X}_A$ and $\mathcal{Y}_A$ from $\mathcal{P}$, $\mathcal{X}_\mathcal{P}$ and $\mathcal{Y}_\mathcal{P}$.
        Add A_L, A_R, $\mathcal{X}_{A_L}$, $\mathcal{X}_{A_R}$, $\mathcal{Y}_{A_L}$ and $\mathcal{Y}_{A_R}$ to $\mathcal{P}$, $\mathcal{X}_\mathcal{P}$ and $\mathcal{Y}_\mathcal{P}$.
      end if
    end if
  end while
end for

Algorithm 2 Best cut and assignation.

Input: Cell A, $(\mathcal{X}_A, \mathcal{Y}_A)$, $\mathcal{M}_{try}$.

Output: Best cut and assignation $(\hat{z}, \hat{w})$.

Set $max_{CART} \leftarrow 0$
for $h \in \mathcal{M}_{try}$ **do**
    Compute the midpoint of two consecutive values of $\mathbf{X}^{(h)}$ between those points $\mathbf{X} \in A$, let be $Z_A^{(h)}$ the set of these midpoints.
    Let be $W_A^{(h)}$ the set with all the possible assignation for the missing values in h.
    for $z \in Z_A^{(h)}$ **do**
        for $w \in W_A^{(h)}$ **do**
            Let c_A be the CART-criterion computed with the cut (h, z) and the assignation w.
            if $c_A > max_{CART}$ **then**
                $max_{CART} \leftarrow c_A$.
                $(\hat{z}, \hat{w}) \leftarrow (z, w)$
            end if
        end for
    end for
end for

3.2. Complexity of the algorithm

Several algorithms do not only compute the random forest estimators but many other by-products, like the proximity matrix or measures of feature importance. More importantly, they are not programmed with the same quality, some take advantage of parallel computation, while others do not, which makes challenging to compare directly the methods from a perspective of computational resources or time of execution, and might not reflect the advantages of some algorithms. Due to all this variability in the computation of the algorithms, we have considered as an alternative to study their algorithmic complexity. Thus, in this section we calculate the complexity of the algorithm. With an efficient use of computational resources, the use of parallel computing, and a high-quality code in a low-level programming language, it is not a prohibited algorithm, but a technique that might be used for real applications.

At first sight, it looks that the number of possible assignations defined by $W_A^{(h)}$ in Algorithm 2 are all the combinations for the assignations of the $N_{miss}^{(h)}(A)$ missing observations. This would lead to exponential complexity in the number of calculations of the CART criterion and then to an intractable algorithm. However, the number of possible candidate assignations to maximize the CART criterion is lower than this quantity. To see this, fix a cell A and a cut (h, z) in A, to keep a simple notation let $\bar{Y}_{L,obs}$ (resp. $\bar{Y}_{R,obs}$) be the mean of the response variable for the points belonging to the left (right) node and observed in the direction h. Suppose without lost of generality that $\bar{Y}_{L,obs} \leq \bar{Y}_{R,obs}$ and denote by $\mathbf{i}_{miss} = \{1, \ldots, N\}$ the set of indexes of the observations assigned to the cell A whose direction h is missing, without lost of generality assume that $Y_1 \leq \cdots \leq Y_N$. Because $\bar{Y}_{L,obs} \leq \bar{Y}_{R,obs}$ and maximizing the CART criterion implies to make as different as possible the average of the target on the left node from the average of the target on the right node, then observations $i \in \mathbf{i}_{miss}$ with the

lowest values Y_i should be assigned to the left node and the observations $i \in \mathbf{i}_{miss}$ with the largest values Y_i should be assigned to the right node. Therefore, there exists $w \in \{1, \dots, N+1\}$ such that assigning $Y_1, \dots, Y_{w-1} \in A_L$ and $Y_w, \dots, Y_N \in A_R$ maximizes the CART criterion. Hence, the set with all possible assignations $W_A^{(h)}$ has a cardinality of $N_{miss}^{(h)}(A) + 1$ and there is only a linear number of assignations to be considered. Now, we can calculate the complexity of the algorithm.

Let be O_{CART} the number of operations needed to calculate the CART criterion for a given cut (h, z) and assignation w of missing values. It makes sense to consider the complexity of our algorithm with respect to O_{CART} since every other random forest algorithm that we compare to also makes a certain number of evaluation of the CART criterion. Consider the Algorithm 2 and note that $|Z_A^{(h)}| = N_{obs}^{(h)}(A) - 1$ and $|W_A^{(h)}| = N_{miss}^{(h)}(A) + 1$. Thus, the number of necessary operations to get the best cut and assignation is

$$\sum_{h \in \mathcal{M}_{try}} |Z_A^{(h)}||W_A^{(h)}|O_{CART}$$

On the other hand, denote by $\mathcal{P}_{nf}$ the set of non-final nodes of a tree, it is clear that $|\mathcal{P}_{nf}| \leq n - 1$, where the equality holds when each final cell contains just one observation. Now, let be O_{tree} the number of operations needed to build a regression tree with our approach, then

$$\begin{aligned}
O_{tree} &= \sum_{A \in \mathcal{P}_{nf}} \sum_{h \in \mathcal{M}_{try}} |Z_A^{(h)}||W_A^{(h)}|O_{CART} \\
&= \sum_{A \in \mathcal{P}_{nf}} \sum_{h \in \mathcal{M}_{try}} \left(N_{obs}^{(h)}(A) - 1\right) \left(N_{miss}^{(h)}(A) + 1\right) O_{CART} \\
&\leq \sum_{A \in \mathcal{P}_{nf}} \sum_{h \in \mathcal{M}_{try}} \left(N(A) - 1\right) \left(N(A) + 1\right) O_{CART} \\
&\leq \texttt{mtry} \times n^3 \times O_{CART}.
\end{aligned}$$

3.3. On simplifications of the algorithm

In this section we discuss a remark that leads to further simplification of our algorithm in order to reach an algorithmic complexity of the same order than MIA. Recall from Section 3.2 that the algorithmic complexity of our proposal is of the order $O(n^3)$ times the calculation required to compute one single CART criterion and that MIA algorithm (in $O(n^2)$) is quicker by a linear factor.

From the CART criterion written in Equation (1), we see that the criterion is convex with respect to the variable that counts the number of observations assigned into A_L, see Figure 1 for a graphical representation of this fact. This simple remark leads us to opt for a dichotomy strategy for the optimal assignation of the missing variables. Indeed, at first step, we assign half of the missing variables to the left (in A_L) and half to the right (in A_R). The corresponding assignation is then given by a $k = \lfloor N/2 \rfloor$ (called pivot for the dichotomy) such that $Y_1 \leq \cdots \leq Y_k \in A_L$ and $Y_{k+1} \leq \cdots \leq Y_N \in A_R$, assuming once again that $\bar{Y}_{L,obs} \leq \bar{Y}_{R,obs}$ without lost of generality. Such configuration gives a CART criterion resumed (abusively) in the

notation $CART(k)$. We search for the optimal assignation through calculations of the local gradient given by

$$\nabla CART(k) = CART(k+1) - CART(k).$$

If the value is positive (resp. negative), then the optimal k is bigger or equal (resp. smaller or equal) to $\lfloor N/2 \rfloor$. Say (for example), that $\nabla CART(k) > 0$. Then, we place the new pivot at the point $k = \lfloor 3N/4 \rfloor$ and compute $\nabla CART(k)$. Once again, the sign of the gradient tells us in which sub interval of $[\lfloor N/2 \rfloor, N]$ the optimal assignation is. We, then repeat this simple searching procedure until ending with an interval containing only two points. The biggest $CART$ value of the two gives the optimal assignation of our variables. This dichotomy procedure allows to consider only $\log(N)$ computations of the local gradient and a simple comparison at the end. This allows us to replace the complexity of the algorithm by

$$O_{tree} \leq \texttt{mtry} \times 2n^2 \log(n) \times O_{CART}$$

which is comparable to the MIA algorithm complexity up to a logarithmic factor. This optimization of the procedure is particularly important when the algorithm deals with a very corrupted data set where the missing entries could represent a significant proportion of the all data.

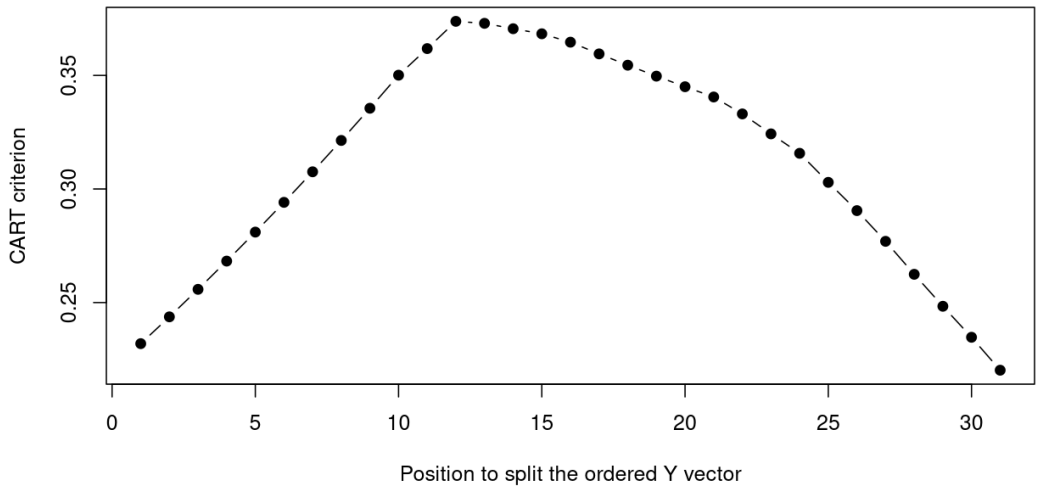

Figure 1. CART criterion as a function of the position where we split $Y_1, \ldots, Y_N$.

4. Experimental details

Planning a computational study for machine learning algorithms is crucial to achieve relevant results. However, this kind of studies do not hold an evident answer on how they should be done. There are simply too many factors and questions to investigate

without a trivial answer: like the methods taken as benchmark, the metrics to compare the algorithms and their estimators, the data sets to use, the mechanisms to introduce missing values, the percentage of missingness, etc. Thus, for sake of clarity, in this article we have limited the analysis to simulated data, which gives us several advantages over real data sets. For example, we can compare the estimations directly with the real regression function, analyzing the bias and the mean squared error (MSE). On the other hand, to present a most exhaustive study, we consider many algorithms that have been used in previous studies as a benchmark, allowing the comparison of both simple approaches to handle missing data as well as more complicated state-of-the-art algorithms. Moreover, we introduce the missing values considering not only one data-missing mechanism, but several of them, and vary the ratio of missingness from a very low percentage (5%) to a large percentage (95%).

We now present the details of the simulation study. For sake of clarity, we first introduce the methods taken as benchmark, then present the parameters of the random forest algorithm, and finally introduce the mechanisms considered to incorporate missing values.

4.1. Benchmark methods

Many methods proposed in the literature to handle missing data using random forests operate through imputation in a recursive way. They start by using the original training data set $\mathcal{D}_n$ to fill the blank spaces in a rough way. For example, using the median of the observed values in the specific direction. We denote this new data set as $\mathcal{D}_{n,1}$.

The imputed data set $\mathcal{D}_{n,1}$ is used to build a random forest. Then, some structures of the forest are exploited, like the so-called proximity matrix, improving the imputation and resulting in a new data set $\mathcal{D}_{n,2}$. The procedure continues by iterations until some stopping rule is applied. These stopping rules trigger, for example, when the update in imputed variables becomes negligible or when a fix number of iterations is achieved. More formally, let us define

$$\mathbf{X}_{i,\ell}^{(h)} = \begin{cases} \mathbf{X}_i^{(h)} & \text{if } \mathbf{M}_i^{(h)} = 0 \\ \widehat{\mathbf{X}}_{i,\ell}^{(h)} & \text{if } \mathbf{M}_i^{(h)} = 1 \end{cases}$$

where $\widehat{\mathbf{X}}_{i,\ell}^{(h)}$ is the imputation of $\mathbf{X}_i^{(h)}$ at time $\ell \geq 1$, and let $\mathbf{X}_{i,\ell} = \left(\mathbf{X}_{i,\ell}^{(1)}, \ldots, \mathbf{X}_{i,\ell}^{(p)}\right)$. To properly introduce the methods considered in the simulation study, we need to define the connectivity between two points in a tree and the proximity matrix of the forest. Let $K_{\Theta,n}(\mathbf{X}, \mathbf{X}')$ be 1 if and only if $\mathbf{X}$ and $\mathbf{X}'$ belong to the same final cell in the tree designed with $\mathcal{D}_n$ and the parameter Θ. In such case we say that $\mathbf{X}$ and $\mathbf{X}'$ are connected in the tree $m_n(\cdot; \Theta)$. Finally, the proximity between $\mathbf{X}$ and $\mathbf{X}'$ in the random forest $m_{M,n}(\cdot; \Theta_1, \ldots, \Theta_M)$, is defined as

$$K_{M,n}(\mathbf{X}, \mathbf{X}') = \frac{1}{M} \sum_{k=1}^{M} K_{\Theta_k,n}(\mathbf{X}, \mathbf{X}').$$

Analogously, we define the proximity $K_{M,\ell}(i,j)$ between $\mathbf{X}_i$ and $\mathbf{X}_j$ at time ℓ (i.e. using the data set $\mathcal{D}_{n,\ell}$). We also define $\mathbf{i}_{miss}^{(h)} \subseteq \{1, \ldots, n\}$ as the indexes where $\mathbf{X}^{(h)}$ is missing, and $\mathbf{i}_{obs}^{(h)} = \{1, \ldots, n\} \setminus \mathbf{i}_{miss}^{(h)}$ as the indexes where $\mathbf{X}^{(h)}$ is observed.

We consider three different approaches which impute missing values through random forests. These methods correspond to the algorithms presented in [5–7]. We refer to those as Breiman's approach, Ishioka's approach and missForest approach, respectively. These algorithms impute the missing values through iterative improvements. We also consider MIA, which is an algorithm that handle the missing values directly in the construction of the trees, assigning all the missing values to the same cell. As simple baselines we consider median-imputation of the missing values and listwise deletion (i.e. removing the observations with missing values) before the construction of the random forest.

4.1.1. With imputation

In this section, we describe the algorithms that belong to the iterative imputation group that are included in the upcoming simulations.

Breiman's Approach. If $\mathbf{X}^{(h)}$ is a continuous variable, $\widehat{\mathbf{X}}^{(h)}_{j,\ell+1}$ is the weighted mean of the observed values in $\mathbf{X}^{(h)}$, where the weights are defined by the proximity matrix of the previous random forest, that is

$$\widehat{\mathbf{X}}^{(h)}_{j,\ell+1} = \frac{\sum_{i\in\mathbf{i}^{(h)}_{obs}} K_{M,\ell}(i,j)\mathbf{X}^{(h)}_i}{\sum_{i\in\mathbf{i}^{(h)}_{obs}} K_{M,\ell}(i,j)}, \qquad \begin{array}{l}\ell \geq 1\\ j \in \mathbf{i}^{(h)}_{miss}\end{array}$$

On the other hand, if $\mathbf{X}^{(h)}$ is a categorical variable, $\widehat{\mathbf{X}}^{(h)}_{j,\ell+1}$ is given by

$$\widehat{\mathbf{X}}^{(h)}_{j,\ell+1} = \underset{\mathbf{x}\in\mathcal{X}^{(h)}}{\arg\max} \sum_{i\in\mathbf{i}^{(h)}_{obs}} K_{M,\ell}(i,j)\mathbb{1}_{\mathbf{X}^{(h)}_i=\mathbf{x}}, \qquad \begin{array}{l}\ell \geq 1\\ j \in \mathbf{i}^{(h)}_{miss}\end{array}$$

That is, $\widehat{\mathbf{X}}^{(h)}_{j,\ell+1}$ is the class that maximizes the sum of the proximity considering the observed values in the class.

Ishioka's Approach. If $\mathbf{X}^{(h)}$ is a continuous variable, $\widehat{\mathbf{X}}^{(h)}_{j,\ell+1}$ is the weighted mean of the k nearest neighbors, according to the proximity matrix, over all the values, both imputed and observed. The k closest values are chosen to make more robust the method and avoid values which are outliers.

$$\widehat{\mathbf{X}}^{(h)}_{j,\ell+1} = \frac{\sum_{\substack{i\in \text{neigh}_k \\ i\neq j}} K_{M,\ell}(i,j)\widehat{\mathbf{X}}^{(h)}_{i,\ell}}{\sum_{\substack{i\in \text{neigh}_k \\ i\neq j}} K_{M,\ell}(i,j)}, \qquad \begin{array}{l}\ell \geq 1\\ j \in \mathbf{i}^{(h)}_{miss}\end{array}$$

For categorical variables, it is not necessary to see only the k closest values because the outliers of $\mathbf{X}$ will have few attention. Meanwhile the proximity with missing values should have more attention, especially when the missing rate is high. Hence, if $\mathbf{X}^{(h)}$ is a categorical variable, $\widehat{\mathbf{X}}^{(h)}_{j,\ell+1}$ is given by

$$\widehat{\mathbf{X}}^{(h)}_{j,\ell+1} = \underset{\mathbf{x}\in\mathcal{X}^{(h)}}{\arg\max} \sum_{i\neq j} K_{M,\ell}(i,j)\, \mathbb{1}_{\widehat{\mathbf{X}}^{(h)}_{i,\ell}=\mathbf{x}}, \qquad \begin{array}{l}\ell \geq 1 \\ j \in \mathbf{i}^{(h)}_{miss}\end{array}$$

MissForest. This algorithm treats the imputation as a regression problem by itself, where the target variable is the variable with missing values. MissForest predicts the missing values using a random forest trained on the observed parts of the data set. More formally for an arbitrary variable $\mathbf{X}^{(h)}$ we can separate the data set into four parts:

- the observed parts of the variable $\mathbf{X}^{(h)}$, denoted as $\mathbf{y}^{(h)}_{obs}$;
- the missing values of the variable $\mathbf{X}^{(h)}$, denoted as $\mathbf{y}^{(h)}_{miss}$;
- the variables other than $\mathbf{X}^{(h)}$ and the observations whose indexes belong to $\mathbf{i}^{(h)}_{obs}$, denoted by $\mathbf{x}^{(h)}_{obs}$;
- the variables other than $\mathbf{X}^{(h)}$ and the observations whose indexes belong to $\mathbf{i}^{(h)}_{miss}$, denoted by $\mathbf{x}^{(h)}_{miss}$.

To begin, the original training data set $\mathcal{D}_n$ is used to fill the blank spaces in a rough way, for example, with the median of the observed values in the variable. Then, for each variable $\mathbf{X}^{(h)}$ a random forest is trained with target $\mathbf{y}^{(h)}_{obs}$ and predictors $\mathbf{x}^{(h)}_{obs}$, then the missing values $\mathbf{y}^{(h)}_{miss}$ are imputed with the prediction of $\mathbf{x}^{(h)}_{miss}$ using the random forest.

4.1.2. Without imputation

Missing Incorporated in Attributes. The Missing Incorporated in Attributes (MIA) consists in keeping all the missing values together when a split is performed. That is, missing values are assigned together to the child node that maximizes the CART criterion (or any other considered criterion). Thus, the splits with this approach assign the values according to one of the following rules:

- $\{\mathbf{X}^{(h)} < z \text{ and } \mathbf{M}^{(h)} = 1\}$ versus $\{\mathbf{X}^{(h)} \geq z\}$,
- $\{\mathbf{X}^{(h)} < z\}$ versus $\{\mathbf{X}^{(h)} \geq z \text{ and } \mathbf{M}^{(h)} = 1\}$,
- $\{\mathbf{M}^{(h)} = 0\}$ versus $\{\mathbf{M}^{(h)} = 1\}$.

4.2. Description of the parameters

The regression function considered in this study is the so-called "friedman1" [19], which has been used in previous simulation studies [12,14,19,24,25], given by

$$m(\mathbf{x}) = 10\sin\left(\pi\mathbf{x}^{(1)}\mathbf{x}^{(2)}\right) + 20\left(\mathbf{x}^{(3)} - 0.5\right)^2 + 10\mathbf{x}^{(4)} + 5\mathbf{x}^{(5)}.$$

Our simulation study is based on the previous work of [12], with the following characteristics:

- We simulate $\mathbf{X}$ uniformly distributed on $[0,1]^5$ and introduce missing values in $\mathbf{X}^{(1)}, \mathbf{X}^{(3)}$ and $\mathbf{X}^{(4)}$, considering 7 different missing-data mechanisms.

- For each missing-data mechanism we create 100 training data sets, each one with 200 observations.
- In $\mathbf{X}^{(1)}$ 20% of the data is missing, in $\mathbf{X}^{(3)}$ the amount is 10%, and in $\mathbf{X}^{(4)}$ there is 20% again.
- We also create a testing data set with 2000 observations without missing values.

  This amount of data is to have an appropriate approximation to the mean squared error (MSE)

$$\mathbb{E}_{\mathbf{X}|\mathcal{D}_n}\left[m_{M,n}(\mathbf{X}) - m(\mathbf{X})\right]^2$$

  and the bias

$$\mathbb{E}_{\mathbf{X}|\mathcal{D}_n}\left[m_{M,n}(\mathbf{X}) - m(\mathbf{X})\right].$$

  Note that these expressions are conditioned on the training sample $\mathcal{D}_n$ and thus they are random variables which take a different value for each one of the 100 training data sets.
- A random forest is built for each training data set and each missing-data mechanism as well as for the data sets without missing values (which are used as benchmark).
- We use $M = 100$ trees, which has been seen by simulation to be sufficient to stabilize the error in the case of the complete data sets.

  For the rest of parameters we use the default values in the regression mode of the `R` package `randomForests`.
- The parameter `mtry` is set to $\lfloor p/3 \rfloor$.
- We have sampled without replacement, so a_n is set to $\lceil 0.632n \rceil$.
- And `nodesize` is set to 5.

These parameters are the same for the median-imputation and MIA approaches. For Breiman's approach, Ishioka's approach and missForest we initialize the algorithms with the median-imputation and consider the same parameters as before for the construction of the regression trees. The number of iterations is set to 10 and random forests are built with 100 trees in each iteration, corresponding to the default values of the package `missForest` in `R`. We now describe the missing-data mechanisms, which are based on those presented by [12].

4.3. Missing-data mechanisms

Missing Completely at Random (MCAR). We select as many locations as desired sampled out of the n observations and replace them by `NA`.

Missing at Random

The choice of the locations that are replaced by missing values in the "missing" variable now depends on the value of a second variable, called the "determining" variable. Therefore, the values of the "determining" variable now have influence on whether a value in the "missing" variable is missing or not. For $\mathbf{X}^{(1)}$ the "determining" variable is $\mathbf{X}^{(2)}$, while $\mathbf{X}^{(5)}$ is used as the "determining" variable for $\mathbf{X}^{(3)}$ and $\mathbf{X}^{(4)}$.

Creation of ranks (MAR1). The probability for a missing value in a certain location in the "missing" variable is computed by dividing the rank of the location in the "determining" variable by $n(n+1)/2$. The locations for `NA` in the "missing" variable are then sampled with the resulting probability vector.

Creation of two groups (MAR2). We divide the data set in two groups defined by the "determining" variable. A value belongs to the first group if the value in the "determining" variable is greater than or equal to the median of the "determining" variable, otherwise it belongs to the second group. An observation has a missing value with probability of 0.9 for the first group (0.1 for the second group) divided by the number of members in the respective group. The locations for `NA` in the "missing" variable are then sampled with the resulting probability vector.

Dexter truncation (MAR3). The observations with the biggest values in the "determining" variable have the "missing" variable replaced by `NA` until the desired fraction of `NA` has been achieved.

Symmetric truncation (MAR4). This method is similar to the previous one but we replace by `NA` the values in the "missing" variable in the observations with the biggest and the smallest values in the "determining" variable.

Missing depending on Y (DEPY). The missing values depend on the value of the response, the probability is 0.1 for observations where $Y \geq 13$, otherwise it is 0.4. The locations for `NA` in the "missing" variable are then sampled with the resulting probability vector.

Missing Not at Random

Logit modelling (LOG). In this method the probability for `NA` no longer depends on a single "determining" variable but in all the other variables. It is modeled as

$$\text{logit}\left(\mathbb{P}\left[\mathbf{M}^{(h)} = 1\right]\right) = -0.5 + \sum_{\substack{k=1 \\ k \neq h}}^{5} \mathbf{X}^{(k)}$$

Therefore, the probability of missingness depends on variables with observed values and variables with missing values.

Complete Observations

Additionally, we consider the data sets with no missing values as a benchmark, which is denoted as "COMP".

5. Results

5.1. Change of missing-data mechanism

Figure 2 presents the average MSE over all the training data sets for each of the approaches considered. In green there is listwise deletion, those approaches that im-

plement some imputation in the data set (median-imputation, Breiman's approach, Ishioka's approach and missForest) are in blue and those approaches that handle missing values directly in the construction of the trees (MIA and our proposal) are in red. Listwise deletion (denoted as "NoRows") generates the largest MSE. Hence, it could be taken as a bound of the minimum expected performance for a method that attempts to estimate the regression function with missing values. We observe that missForest consistently generates estimators with the lowest MSE regardless of the missing-data mechanism. On the other hand, we observe that our proposed method outperforms MIA, Breiman's approach and Ishioka's approach and can achieve similar MSE as missForest. For the DEPY data-missing mechanism, we can see that the algorithms form two groups, keeping apart listwise deletion with the largest MSE, while the group with the lowest average MSE is form by missForest and our proposal, the rest of the algorithms here considered present a similar average MSE which stands somewhere between listwise deletion and the second group. It is worthy to observe that even a simple approach as imputing with the median can outperforms most of the methods considered or can achieve a similar behavior in several missing-data mechanisms (see MAR1, MAR2, MAR3, LOG).[1]

Analogously, Figure 3 presents the average bias over all the training data sets for each of the approaches considered. As expected, listwise deletion generates the estimators with more bias for all the mechanisms. In terms of bias, we observe that the algorithms that impute the missing values before the construction of the random forest tend to generate less biased results, while MIA consistently tends to generate the second more biased estimators. For the MCAR case, all the methods considered tend to be unbiased. On the other hand, in the case of the DEPY data-missing mechanism we observe the biggest variability between the methods, and it becomes more evident how the methods that perform some imputation previous to the construction of the random forest tend to generate less biased estimators. We see that all the methods generate the same kind of bias. That is, all of them underestimate or all of them overestimate the regression function, just changing the magnitude of this bias. It is worthy to observe that we can obtain unbiased estimators even for missing not at random mechanism, in fact for the LOG case all the methods tend to generate unbiased estimators.

For the proportion of missing values considered, we observe little variability in the average MSE and the average bias per method and per data-missing mechanism. However, important differences appears when the rate of missing values increases, with the variability in the average MSE and the average bias increasing with it (see Tables 1 to 4).

[1]Codes to reproduce our results can be found in: `https://github.com/IrvingGomez/RandomForestsSimulations`, while examples on the use of these codes can be found in `https://github.com/IrvingGomez/Random_forests_with_missing_values`.

Figure 2. Average MSE of the testing data set for each approach and each missing-data mechanism.

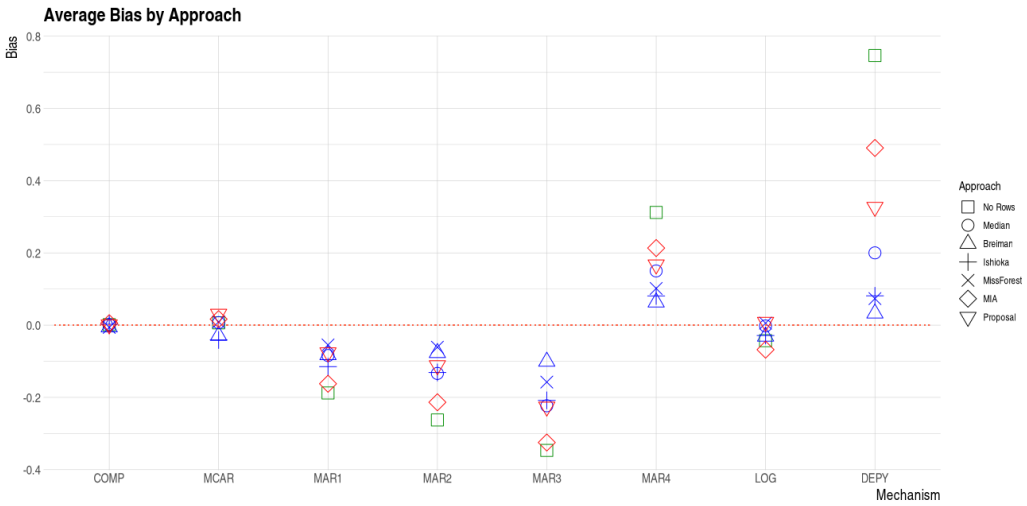

Figure 3. Average bias of the testing data set for each approach and each missing-data mechanism.

5.2. Increasing the rate of missingness

Using the 100 training data sets with no missing values, we calculate the importance of the variables with the `R` package `randomForests`, by percentage of increase in mean squared error and by increase in node purity [5,22]. Figures 4 and 5 show the violin plots for these measurements of importance. We can observe a consistently order for the variables with missing values in both measures of importance, where $\mathbf{X}^{(4)}$ is considered more important than $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(3)}$. Hence, we decided to change the fraction of

missingness in $\mathbf{X}^{(4)}$ to vary between 5%, 10%, 20%, 40%, 60%, 80%, 90% and 95%, without changing the percentage of missingness in $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(3)}$. In this part of the study we do not consider anymore listwise deletion.

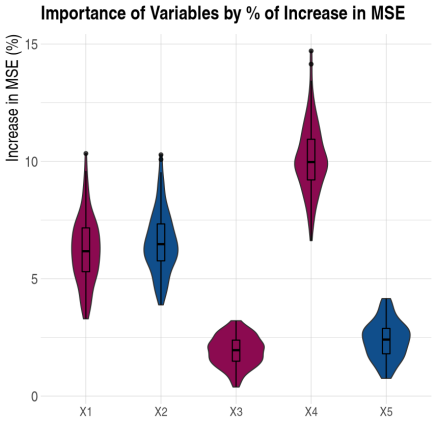

Figure 4. Importance variable accordingly to percentage increase in MSE. Variables with missing values are in wine, while variables with complete values are in blue.

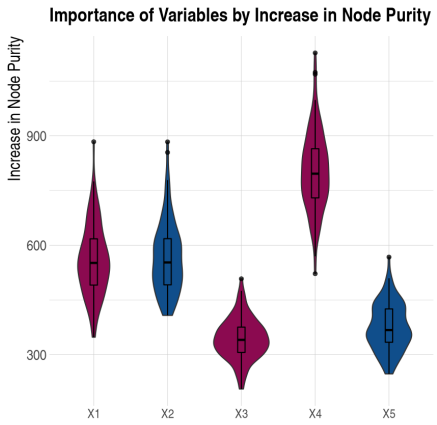

Figure 5. Importance variable accordingly to increase in node purity. Variables with missing values are in wine, while variables with complete values are in blue.

Figure 6 presents the average MSE varying the percentage of missing data and considering the MAR1 mechanism, while Tables 1 and 2 show the average MSE with its standard error. Analogous figures and tables for the MSE for all the missing-data mechanisms can be found in the supplementary material at `https://irvinggomez.com/publication/supplementary_random_forests_simulation/Supplementary_RandomForestsSimulation.pdf`.

Important differences between the performance of the methods become clear with the increasing percentage of missingness. Especially when this value is over 60%, there is an order on the performance of the methods. In those cases, missForest and our proposal represent the methods with less MSE. Moreover, we can see the advantage of searching for the best assignation when the percentage becomes really large with our proposal outperforming all the other algorithms. While all the methods present a similar MSE when the percentage of missing values is smaller than 40%. These same phenomena are observed for the rest of the data-missing mechanisms (see the suplemmentary material). We observe constantly throughout the different data-missing mechanisms that the methods with the worst performance are median-imputation and Breiman's approach, while missForest and our proposal being the methods with the lowest MSE, and MIA and Ishioka's approach lying somewhere between these two groups. Once again, these differences become clear when the percentage of missing values is beyond 60%. Furthermore, we can see in Tables 1 and 2 some deterioration in all the methods when the percentage exceeds 60%, with our approach showing the less MSE and less deterioration in terms of the standard error.

We consistently observe that our approach and missForest outperform the other methods, regardless of the percentage of missing values, while there is no clear advantage for the rest of the algorithms over the others. However, some differences between the missing-data mechanisms are also reflected when we increase the percentage of missing values. For the MCAR case, missForest and our approach present a MSE be-

tween 7.95 and 8.19 when the percentage of missingness is 90%. On the other hand, for the same percentage of missing values and the same algorithms we observe a deterioration in terms of the MSE which varies between 9.05 and 12.86 when we consider the DEPY scenario.

Similarly, Figure 7 presents the average bias varying the percentage of missing data and considering the MAR1 mechanism, while Tables 3 and 4 show the average bias with its standard error. When we include the bias in the study, the differences between methods and missing-data mechanisms become more evident. We observe that this missing-data mechanism introduced a bias in the methods. Analogous figures and tables for the bias for all the missing-data mechanisms can be found in the supplementary material. We observe consistently throughout all the data-missing mechanisms that MIA and median-imputation generate the most biased estimators while Breiman's approach tends to generates the less biased. These differences between methods become clear once the rate of missing data exceeds the 60%.

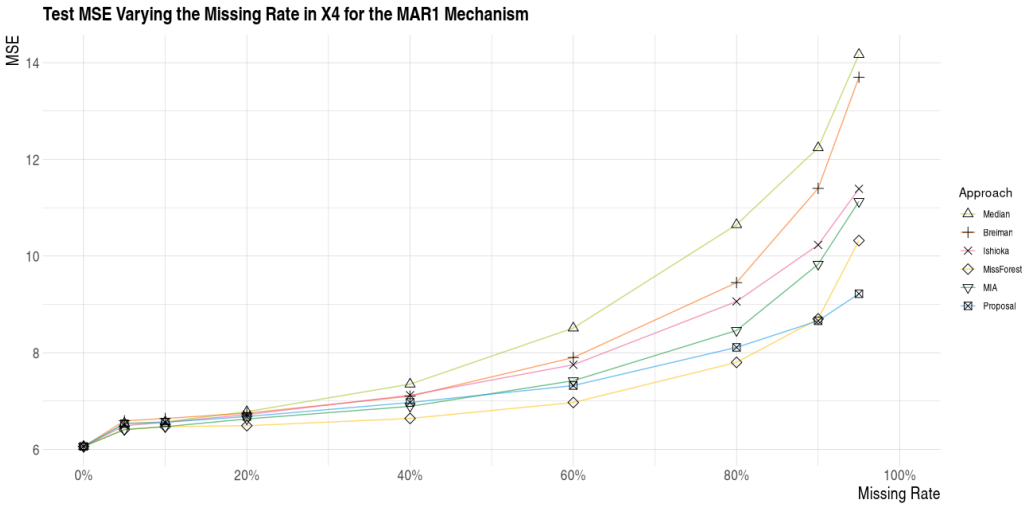

Figure 6. Average MSE for the testing data set for each percentage of missingness, considering the MAR1 mechanism.

	0%	5%	10%	20%	40%
Median	6.06 ± 0.06	6.54 ± 0.06	6.57 ± 0.05	6.78 ± 0.06	7.35 ± 0.07
Breiman	6.06 ± 0.06	6.59 ± 0.06	6.64 ± 0.06	6.75 ± 0.06	7.10 ± 0.06
Ishioka	6.06 ± 0.06	6.49 ± 0.06	6.56 ± 0.06	6.72 ± 0.06	7.12 ± 0.07
MissForest	6.06 ± 0.06	6.41 ± 0.06	6.47 ± 0.06	6.49 ± 0.06	6.64 ± 0.06
MIA	6.06 ± 0.06	6.41 ± 0.06	6.47 ± 0.06	6.63 ± 0.06	6.89 ± 0.07
Proposal	6.06 ± 0.06	6.53 ± 0.06	6.56 ± 0.06	6.68 ± 0.06	6.97 ± 0.06

Table 1. Average mean squared error and its standard error for the different methods, considering the MAR1 case.

	60%	80%	90%	95%
Median	8.51 ± 0.09	10.65 ± 0.15	12.24 ± 0.21	14.17 ± 0.28
Breiman	7.90 ± 0.10	9.45 ± 0.13	11.40 ± 0.26	13.70 ± 0.34
Ishioka	7.75 ± 0.08	9.06 ± 0.12	10.23 ± 0.13	11.39 ± 0.20
MissForest	6.97 ± 0.06	7.80 ± 0.09	8.70 ± 0.14	10.32 ± 0.35
MIA	7.42 ± 0.08	8.46 ± 0.11	9.83 ± 0.14	11.13 ± 0.20
Proposal	7.32 ± 0.07	8.11 ± 0.08	8.66 ± 0.08	9.22 ± 0.11

Table 2. (Cont.) Average mean squared error and its standard error for the different methods, considering the MAR1 case.

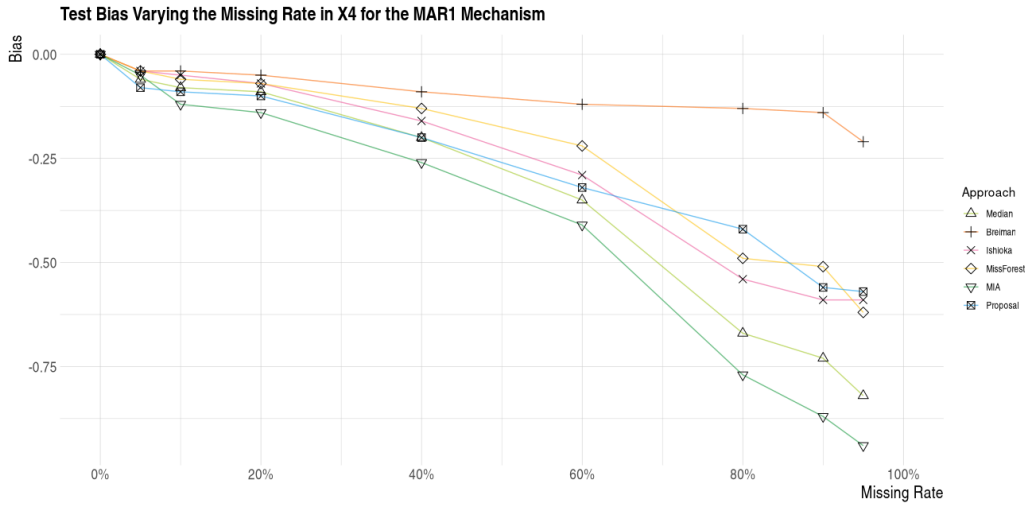

Figure 7. Average bias for the testing data set for each percentage of missingness, considering the MAR1 mechanism.

	0%	5%	10%	20%	40%
Median	0.00 ± 0.02	-0.06 ± 0.02	-0.08 ± 0.02	-0.09 ± 0.02	-0.20 ± 0.03
Breiman	0.00 ± 0.02	-0.04 ± 0.02	-0.04 ± 0.02	-0.05 ± 0.03	-0.09 ± 0.03
Ishioka	0.00 ± 0.02	-0.04 ± 0.02	-0.05 ± 0.02	-0.07 ± 0.02	-0.16 ± 0.03
MissForest	0.00 ± 0.02	-0.05 ± 0.02	-0.06 ± 0.02	-0.07 ± 0.02	-0.13 ± 0.02
MIA	0.00 ± 0.02	-0.08 ± 0.02	-0.12 ± 0.02	-0.14 ± 0.03	-0.26 ± 0.03
Proposal	0.00 ± 0.02	-0.08 ± 0.02	-0.09 ± 0.02	-0.10 ± 0.02	-0.20 ± 0.03

Table 3. Average bias and its standard error for the different methods, considering the MAR1 case.

5.3. Prediction of new observations with missing entries

A challenge in machine learning is to compute a solution or a prediction to a certain problem when the given information is somewhat incomplete. In our specific context, our proposal do not have to rely on imputations of the missing entries to be able to calculate a prediction for a new data point. The fact that most of the existing techniques use imputation to construct the random forest tend to provide a prediction that is highly dependent on those imputations. In the following, we explain a way to perform the prediction phase without having to impute the missing entries of the new

	60%	80%	90%	95%
Median	-0.35 ± 0.03	-0.67 ± 0.03	-0.73 ± 0.05	-0.82 ± 0.04
Breiman	-0.12 ± 0.03	-0.13 ± 0.03	-0.14 ± 0.03	-0.21 ± 0.05
Ishioka	-0.29 ± 0.03	-0.54 ± 0.03	-0.59 ± 0.05	-0.59 ± 0.05
MissForest	-0.22 ± 0.03	-0.49 ± 0.04	-0.51 ± 0.05	-0.62 ± 0.07
MIA	-0.41 ± 0.03	-0.77 ± 0.03	-0.87 ± 0.05	-0.94 ± 0.05
Proposal	-0.32 ± 0.03	-0.42 ± 0.04	-0.56 ± 0.03	-0.57 ± 0.03

Table 4. (Cont.) Average bias and its standard error for the different methods, considering the MAR1 case.

data point.

At this stage, we assume that the training phase is finished and that one have access to the random forest with the assignations associated to each tree. The prediction is computed tree by tree and is averaged over the different trees in the random forest. A tree prediction is performed looking for a (pseudo-random) final cell in the tree that is the most likely to contain the query point $\mathbf{X}$. This is done in a recursive manner by going down in the tree following a procedure that we describe now. Assume that we are in the cell A which has a cut (h, z) that splits it into A_L and A_R and that the assignation vector associated to that cut is given by w. If the direction h is observed then, as usual, $\mathbf{X}$ is assigned to the left if $\mathbf{X}^{(h)} < z$, otherwise it is assigned to the right. Let us now assume that the direction h is missing, in this case we need to look at the vector w to assign the query point. Once again, let N be the number of points with a missing value in the direction h let N_L (resp. $N_R = N - N_L$) be the number of such points assigned to the left (right) node. At this step, if $N = 0$ (when no missing values where observed in the direction h in the cell during the training phase) we stop the descent and predict the value of Y by its mean value in the cell A. Otherwise, $N \neq 0$ and we can compute $p_L = N_L/N$ (resp. $p_R = N_R/N$) the empirical probability for a missing observation to belong to the left (resp. right) node, given that it belongs to cell A. The next cell is, then, stochastically selected to the left or to the right with respective probabilities p_L and p_R. Our algorithm keeps track of the assignations w at each step so that computing the probabilities p_L and p_R is direct. We can think of the same procedure for the other random forest techniques that do perform a certain kind of assignation. For example, MIA algorithm is the most suited for a comparable treatment for incomplete data points in the testing phase. Nevertheless, the "assignation information" is not accessible in the existing codes for MIA. For the case of the algorithms that do perform imputation of the missing values, it is not clear how an imputation has to be done for the testing phase. As discussed in [1], these algorithms are really dependent on the way the imputation is performed and then no theoretical guaranties are known for their consistency. In the following, we give some simulations of this testing phase letting the proportion of missing values to vary.

In the training phase:

- For each data set and each of the 7 missing-data mechanisms (MCAR, MAR1, MAR2, MAR3, MAR4, LOG y DEPY) we introduce missing values in the variables $\mathbf{X}^{(1)}$, $\mathbf{X}^{(3)}$ and $\mathbf{X}^{(4)}$. The proportion of missing values are 20% for $\mathbf{X}^{(1)}$, 10% for $\mathbf{X}^{(3)}$ and 60% for the variable $\mathbf{X}^{(4)}$.
- The random forests are constructed following the lines of Section 4.2

For the testing phase:

- For each missing-data mechanism, we let the percentage of missing values for $\mathbf{X}^{(4)}$ to vary from 0% to 95% and let the other two proportions of missing values for $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(3)}$ unchanged (at 20% and 10%, resp.)
- The graphics in Figure 8 represent the MSE on those 2000 observations where 100 random forests are computed for each missing-data mechanism.

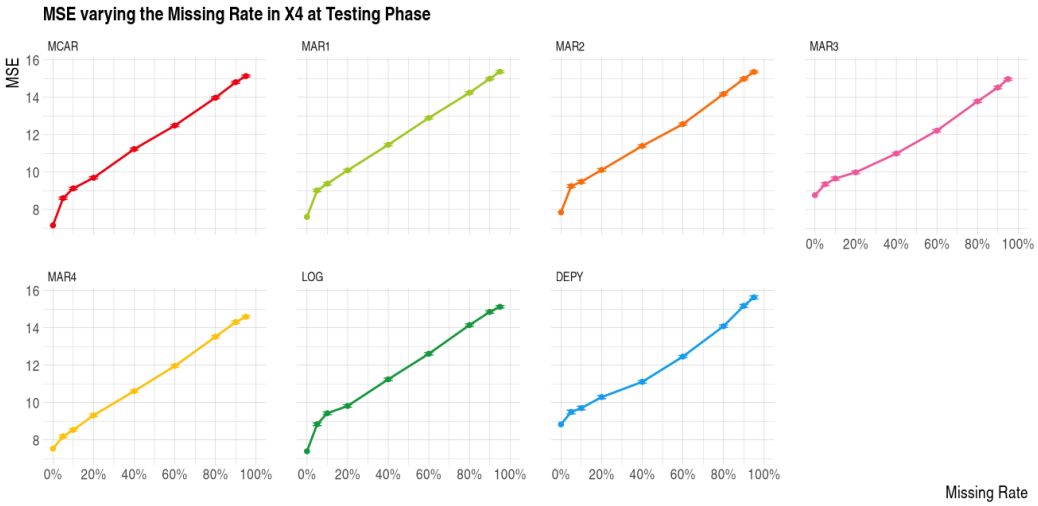

Figure 8. We compute the MSE over a testing set of 2000 data points, the proportion of missing values for $\mathbf{X}^{(4)}$ varies from 0% to 95%.

Using this technique to assign a new observation with a pseudo-random approach during the prediction phase, we can observe a linear behavior in the estimation of the MSE trough all the data-missing mechanisms, which shows a robust approach to the mechanism that has generated the missing values. Moreover, we do not observe important differences between the values of the MSE throughout the mechanisms. We encourage the incorporation of this technique in other algorithms, or clear statements on how to deal with missing values during a prediction phase, allowing the comparison of the distinct algorithms beyond the training phase.

6. Discussion and conclusions

We developed a simulation study comparing a new proposal with other 6 distinct approaches based on random forests to perform regression with missing entries. Four of the algorithms impute the missing values to create completed data sets and then construct random forests in the usual way. Two algorithms belonging to this group rely on the computation of the proximity matrix [5,6] and improve the imputations iteratively. The third algorithm imputes the missing values with the median of the observations. The last algorithm of this group corresponds to missForest [7] which performs imputation through the implementation of random forests where the imputation of the missing values is treated as a regression problem by itself. We also considered MIA [9] which, similar to our proposed algorithm, handle missing values directly in the construction of the trees. Finally, as a simple benchmark we considered listwise deletion.

For the simulation study we considered the so-called "friedman1" regression function [19], which has been used in previous simulation studies [12,14,19,24,25], and considered 7 different mechanisms to introduce missing values in the data sets; one being missing completely at random, denoted as MCAR; five of these mechanisms being missing at random, denoted as MAR1, MAR2, MAR3, MAR4, and DEPY; and the last mechanism being missing not at random, denoted as LOG.

With no surprise, listwise deletion was the approach with the worst performance, this method should be avoided unless the percentage of observations with missing values is so low that they can be deleted without a severe harmful. For the rest of the algorithms, we observed differences between distinct techniques. These differences become more evident when the percentage of missing values increases, especially when it is over 60%, while these differences are diluted for small values of this percentage (less than 40%). Moreover, the behavior of the methods appear to be dependent on the missing-data mechanism. Intensive computer algorithms, as missForest and our approach, seem to perform particularly well for large percentage of missing values.

On the other hand, we see that simple techniques as median-imputation have similar performance to more complicated algorithms when the percentage of missing values is under 40% which makes its use sufficient in practice for few missing value contexts. Furthermore, when the percentage of missing values is low, we observed little variability in the average MSE and the average bias for the estimators obtained for all the methods considered and through all the data-missing mechanisms. However, when this percentage increases, especially over 60%, we observed a deterioration in the estimations, not only in the increase of the MSE and with more biased estimators, but also with more variability which is reflected in larger values for the standard error.

Since the patterns and differences between the methods and data-missing mechanisms become more evident with the increase of the missing values, we encourage the incorporation of a relatively high ratio of missingness for studies that focus on the analysis of algorithms to handle missing values. When this percentage becomes small (say 20% or less), even a simple technique as median-imputation can achieve similar results to more complicated algorithms, which become relevant with a high percentage of missing values.

We presented an extensive simulation study; introducing several data-missing mechanisms; varying the percentage of missingness from very low values (5%) to very large values (95%); and considering distinct methods, from simple approaches as listwise deletion and median-imputation to state-of-the-art algorithms that explode the computational power. However, we consider that more studies, both theoretical and empirical, are still needed. For sake of clarity and for a reasonably comprehensive study, we have limited the analysis in this article to simulated data, but also encourage the use of publicly available data sets with artificially added missing data.

Due to the variability find in the codes of the different techniques, and the calculation of some by-products (like the proximity matrix or measures of feature importance), that might increase both the execution time and the computation resources, we found challenging to analyze the algorithms through computational metrics. Therefore, we considered as an alternative to study and compare the algorithms through their theoretical complexity and the empirical analysis of simulated data, allowing the comparison of the results with the real regression function, and where we have access to the mechanisms that generated the missing values. Thus, we studied the complexity of our proposal in Section 3.2 and show in Section 3.3 how a bisection technique can be performed to find the best assignation of the missing values. With these simplifications, we proved that the complexity of the algorithm is comparable to

the MIA algorithm complexity up to a logarithmic factor, making possible to apply the algorithm for real applications.

Finally, in Section 5.3 we explain a process that allows the prediction of a new observation with missing entries. This procedure uses the assignation of the missing values during the training phase to estimate the probabilities of belonging to each cell. Hence, the new observation can be assigned stochastically to the cells of each tree using these probabilities. In the end, the prediction of the random forest is simply the average of the prediction of each tree. We observed through the simulation study that this technique seems to be robust to the missing-data mechanism and that it could be applied even for data sets with several missing values. Moreover, it can be applied with any other technique that assigns the missing observations, as long as the information of these assignations is kept. We also encourage to consider the prediction step and not only the training phase in studies dealing with missing values. Up to now, several approaches do not specify how to predict a new observation where only a part of the predictor variables are available, this might be challenging specially for algorithms that use the response to impute the missing entries. This step has a huge importance and should not be avoided, since the same mechanism that generated missing values during the training step could operate during the prediction phase.

References

[1] Gómez-Méndez I, Joly E. On the consistency of a random forest algorithm in the presence of missing entries. arXiv preprint arXiv:201105433. 2020;.

[2] Biau G, Scornet E. A random forest guided tour. Test. 2016;25(2):197–227.

[3] Troyanskaya O, Cantor M, Sherlock G, et al. Missing value estimation methods for dna microarrays. Bioinformatics. 2001;17(6):520–525.

[4] Van Buuren S, Brand JP, Groothuis-Oudshoorn CG, et al. Fully conditional specification in multivariate imputation. Journal of statistical computation and simulation. 2006; 76(12):1049–1064.

[5] Breiman L. Setting up, using, and understanding random forests v4.0 ; ???? Available from: `https://www.stat.berkeley.edu/~breiman/Using_random_forests_v4.0.pdf`.

[6] Ishioka T. Imputation of missing values for unsupervised data using the proximity in random forests. In: International Conference on Mobile, Hybrid, and On-line Learning. Nice; 2013. p. 30–36.

[7] Stekhoven DJ, Bühlmann P. Missforest—non-parametric missing value imputation for mixed-type data. Bioinformatics. 2011;28(1):112–118.

[8] Breiman L, Friedman JH, Stone C, et al. Classification and regression trees. Chapman and Hall/CRC; 1984.

[9] Twala B, Jones M, Hand DJ. Good methods for coping with missing data in decision trees. Pattern Recognition Letters. 2008;29(7):950–956.

[10] Feelders A. Handling missing data in trees: surrogate splits or statistical imputation? In: European Conference on Principles of Data Mining and Knowledge Discovery; Springer; 1999. p. 329–334.

[11] Farhangfar A, Kurgan L, Dy J. Impact of imputation of missing values on classification error for discrete data. Pattern Recognition. 2008;41(12):3692–3705.

[12] Rieger A, Hothorn T, Strobl C. Random forests with missing values in the covariates. Technical report. 2010;Available from: `http://epub.ub.uni-muenchen.de/11481`.

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

[14] Josse J, Prost N, Scornet E, et al. On the consistency of supervised learning with missing

values. arXiv preprint arXiv:190206931. 2019;.
[15] Schafer JL. Analysis of incomplete multivariate data. CRC press; 1997.
[16] Schafer JL, Olsen MK. Multiple imputation for multivariate missing-data problems: A data analyst's perspective. Multivariate behavioral research. 1998;33(4):545–571.
[17] Quinlan JR. C4.5: Programs for machine learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 1993.
[18] Hothorn T, Hornik K, Zeileis A. Unbiased recursive partitioning: A conditional inference framework. Journal of Computational and Graphical statistics. 2006;15(3):651–674.
[19] Friedman JH, et al. Multivariate adaptive regression splines. The annals of statistics. 1991;19(1):1–67.
[20] Chen T, Guestrin C. Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining; 2016. p. 785–794.
[21] Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society: Series B (Methodological). 1977; 39(1):1–22.
[22] Breiman L. Random forests. Machine learning. 2001;45(1):5–32.
[23] Rubin DB. Inference and missing data. Biometrika. 1976;63(3):581–592.
[24] Breiman L. Bagging predictors. Machine learning. 1996;24(2):123–140.
[25] Friedberg R, Tibshirani J, Athey S, et al. Local linear forests. Journal of Computational and Graphical Statistics. 2020;:1–15.