Multiple Imputation Through XGBoost

Yongshi Deng and Thomas Lumley

Department of Statistics, University of Auckland, Auckland, New Zealand

ABSTRACT
The use of multiple imputation (MI) is becoming increasingly popular for addressing missing data. Although some conventional MI approaches have been well studied and have shown empirical validity, they have limitations when processing large datasets with complex data structures. Their imputation performances usually rely on the proper specification of imputation models, and this requires expert knowledge of the inherent relations among variables. Moreover, these standard approaches tend to be computationally inefficient for medium and large datasets. In this article, we propose a scalable MI framework \textit{mixgb}, which is based on XGBoost, subsampling, and predictive mean matching. Our approach leverages the power of XGBoost, a fast implementation of gradient boosted trees, to automatically capture interactions and nonlinear relations while achieving high computational efficiency. In addition, we incorporate subsampling and predictive mean matching to reduce bias and to better account for appropriate imputation variability. The proposed framework is implemented in an \textit{R} package \textit{mixgb}. Supplementary materials for this article are available online.

1. Introduction
Multiple imputation (MI), first introduced by Rubin (1978), has received increasing recognition in dealing with missing data because it can reduce bias and represent the uncertainty of missing values. Given an incomplete dataset, each missing value is replaced by a set of $M > 1$ plausible values instead of a single value. Analyses can then be performed separately on $M$ complete imputed datasets, and the results can be combined to yield valid inference. Rubin (1987) argued that with proper multiple imputation, the pooled estimates and variance would be statistically valid. A number of frameworks have been developed to implement MI. However, few automated procedures for large-scale imputation have been devised.

A major flaw of traditional MI implementations is that they fail to automatically capture complex relations among variables. If such relations exist, popular software packages such as \textit{mi} (Su et al. 2011) and \textit{mice} with default settings (van Buuren and Groothuis-Oudshoorn 2011) will produce unsatisfactory results, unless users manually specify any potential nonlinear effects in the imputation model for each incomplete variable. Indeed, van Buuren and Groothuis-Oudshoorn (2011) indicated that it is crucial to include all interactions of interest in the imputation model in order to achieve optimal results. However, it appears that researchers often use \textit{mice} in an automated manner (e.g., Wendt et al. 2021; Awada et al. 2021). Tree-based algorithms provide a potential solution to solve this problem. Stekhoven and Bühlmann (2012) proposed a nonparametric method for imputing missing values based on random forests (Breiman 2001) and implemented it in an \textit{R} package called \textit{missForest}. Doove, van Buuren, and Dusseldorp (2014) implemented classification and regression trees (CART) (Breiman et al. 1984) and random forests within the \textit{mice} framework (\textit{mice-cart} and \textit{mice-rf}) and showed that they can better preserve nonlinear effects, compared to the standard implementation of \textit{mice}.

Another drawback of existing MI frameworks is the excessive computation time for large datasets. Recent advances in technology have made the collection and analysis of large datasets feasible. However, current MI methods, including those that can capture complex data structures such as \textit{mice-cart} and \textit{mice-rf}, are more suited for small datasets and often struggle with medium and large datasets. This problem can become unmanageable when there is a moderate number of missing values present across many variables. Recently, \textit{mice} has included \textit{ranger} (Wright and Ziegler 2017), a fast implementation of random forests, as one of its imputation methods. However, its imputation performance and computational time have not been investigated.

The purpose of this article is to present a fast and automated MI procedure \textit{mixgb}, which is based on XGBoost (Chen and Guestrin 2016), subsampling, and predictive mean matching (Little 1988), with a focus on yielding statistically valid results. XGBoost, a fast tree boosting algorithm, has been a frequent winner in Kaggle data competitions (Chen and Guestrin 2016) and has gained immense popularity due to its speed and accuracy. XGBoost’s ability to efficiently capture complex structures of large datasets means it has great potential for automated MI. With subsampling and predictive mean matching, our proposed method can better incorporate the variability of missing data and enhance imputation quality.
This article is structured as follows. Section 2 describes the proposed MI through XGBoost framework (mixgb) in detail. Section 3.1 provides an overview of the simulation study that we used to evaluate imputation performance. The evaluation criteria for MI implementations are presented in Section 3.2. Simulation results are given in Section 3.3, demonstrating the imputation quality of mixgb. Section 4 demonstrates the advantage of mixgb over other implementations in terms of computational efficiency, and Section 5 provides an example using a real dataset. Finally, discussions based on empirical results are presented in Section 6. We have implemented the proposed method in our R package mixgb (Deng 2023), which is available at https://github.com/agnesdeng/mixgb and on CRAN.

2. Framework

Using XGBoost to impute missing data has attracted growing interest in recent years. However, previous work has focused on the prediction accuracy of missing values without adequately accounting for the uncertainty of missing data, resulting in an underestimation of imputation variability (e.g., Zhang et al. 2019). To address this problem, we propose using XGBoost with subsampling and predictive mean matching (PMM) for MI.

Our R package mixgb offers settings that allow users to use subsampling with different ratios and choose the type of PMM. In this article, we use mixgb to denote using XGBoost with only PMM, and mixgb-sub to denote using XGBoost with PMM and subsampling. Unlike the R package mice, our imputation framework is noniterative. However, users can set the number of iterations in our package mixgb to be greater than one. In Section 3, the imputation performance of both mixgb and mixgb-sub was examined using a single iteration, whereas mice was evaluated with five iterations.

Our framework mixgb-sub works as follows: Given a dataset $Y_{raw}$ with $p$ incomplete variables, we first sort the variables by the number of missing values in ascending order. We then conduct an initial imputation of the missing values to obtain a complete sorted dataset $Y$. We let $Y_{obs}$ and $Y_{mis}$ be the observed values and the imputed (originally missing) values for an incomplete variable $Y_i$. We also let $Y_{obs}^*$ denote the corresponding data in all variables other than $Y_i$ for entries where $Y_i$ was observed. Similarly, $Y_{mis}^*$ denotes the data for variables other than $Y_i$ for entries where $Y_i$ was originally missing before the initial imputation.

We use XGBoost models with subsampling to better account for the uncertainty of missing values. This is analogous to sampling parameters of a parametric model from their posterior distributions. When the subsampling ratio is less than one, a different subset of the data is used in each boosting round. Therefore, the set of data $Y^*$ used in each of the $M$ imputations will differ. For each incomplete variable, we fit an XGBoost model with subsampling and use it to obtain predictions for $Y_{mis}^*$, which we denote by $\hat{Y}_{mis}^*$.

However, for continuous data, simply imputing missing values with point predictions $\hat{Y}_{mis}^*$ is prone to underestimating imputation variability. This problem can be mitigated by PMM, which was first proposed by Rubin (1986) and extended to MI by Little (1988). PMM works by matching the predicted value of each missing entry to a set of $K$ donors that have the closest predicted values among the observed entries. One donor is then randomly selected, and its observed value is used to impute the missing value. We write $\hat{\beta}_i$ for the estimates of model $f^* : Y_{mis}^* \sim Y_{obs}^*$ using subsampling for each imputation. Four types of PMM are summarized in Table 1 based on van Buuren (2018).

We have implemented Type 0, Type 1, and Type 2 PMM in our R package mixgb. Type 0 is improper as it ignores the sampling variability in $\hat{\beta}_i$, so we offer this option only for research purposes. Type 3 needs two distinct draws of estimates for each imputation, which requires training $2M$ imputation models. We consider it computationally inefficient, hence, it is not included in the current version. For Type 1 PMM, the predicted values for the donors are obtained by fitting a model to the entire dataset, whereas the predicted values for the recipients are produced by fitting an imputation model with subsampling for each imputation. For Type 2 PMM, the same model is used to obtain the predicted values for the donors and recipients. By default, our framework uses Type 2 PMM for continuous data and no PMM for categorical data. The detailed algorithm mixgb-sub is described in Algorithm 1 and illustrated in Figure 1. Without subsampling, mixgb works in a similar way. However, the entire dataset is used for each imputation. Note that omitting subsampling will reduce the variability between imputations and the primary source of between-imputation variance will be due to the differences between the PMM donors. If the number of PMM donors is set to one and subsampling is not used, then each of the $M$ imputations will be identical.

Figure 2 illustrates the need for using PMM for imputing continuous data with mixgb. We created a dataset of 1000 observations using $y_i = 5x_i + \epsilon_i$, where $x_i \sim \text{Normal}(0, 1)$ and $\epsilon_i \sim \text{Normal}(0, 1)$ for $i = 1, 2, \ldots, 1000$. We then generated 50% missing data in the variable $x$ under the missing completely at random (MCAR) mechanism. We refer to simulated observations chosen to be missing as "masked true". As shown in Figure 2, the variability within an imputed dataset using mixgb without PMM was considerably less than the variance of the observed data and the variance of the masked true data. In

| PMM   | Donors’ predicted values | Recipients’ predicted values | Differences |
|-------|--------------------------|-----------------------------|-------------|
| Type 0 | $\hat{Y}_{obs}^* = \hat{Y}_{obs}^* \hat{\beta}_i$ | $\hat{Y}_{mis}^* = \hat{Y}_{mis}^* \hat{\beta}_i$ | $\hat{\beta}_i$ are the same across $M$ imputations |
| Type 1 | $\hat{Y}_{obs}^* = \hat{Y}_{obs}^* \hat{\beta}_i$ | $\hat{Y}_{mis}^* = \hat{Y}_{mis}^* \hat{\beta}_i$ | donors use $\hat{\beta}_i$ and recipients use $\hat{\beta}_i^*$ for each imputation |
| Type 2 | $\hat{Y}_{obs}^* = \hat{Y}_{obs}^* \hat{\beta}_i$ | $\hat{Y}_{mis}^* = \hat{Y}_{mis}^* \hat{\beta}_i^*$ | donors and recipients both use $\hat{\beta}_i^*$ for each imputation |
| Type 3 | $\hat{Y}_{obs}^* = \hat{Y}_{obs}^* \hat{\beta}_i$ | $\hat{Y}_{mis}^* = \hat{Y}_{mis}^* \hat{\beta}_i^*$ | $\hat{\beta}_i$ and $\hat{\beta}_i^*$ are two different draws for each imputation |
contrast, both Type 1 and Type 2 PMM had similar within-variance to the masked true data.

\[ Y_i = \text{norm1}_i + \text{norm2}_i + \text{norm3}_i + \text{norm5}_i + \text{norm7}_i + (\text{bin1}_i = 1) - (\text{ord1}_i = 1) - 2 \cdot (\text{ord1}_i = 2) + \text{norm1}^2_i + \text{norm2}_i \times \text{norm3}_i - 3 \cdot \text{norm5}_i \times (\text{bin1}_i = 1) - 2 \cdot \text{norm7}_i \times (\text{ord1}_i = 1) + \text{norm7}_i \times (\text{ord1}_i = 2) + \epsilon_i, \]

where the $\epsilon_i$ was from a standard normal distribution. The variables norm1, ..., norm8 were drawn from standard normal distributions, bin1 was drawn from a binary distribution with $p = 0.5$, and ord1 was drawn from a binomial distribution with $n = 2$ and $p = 0.5$. The distributions of norm1, norm2, norm3, and norm4 had pairwise correlations of 0.5. The binary variable bin1 was correlated with both norm5 and norm6 with correlation $\rho \approx 0.55$, while norm5 and norm6 had correlation of 0.7. The ordinal variable ord1 was correlated with both norm7 and norm8 with correlation $\rho \approx 0.65$, while norm7 and norm8 had correlation of 0.7. Note that norm4, norm6, and norm8 were included in the dataset as ancillary variables, but they were not used to generate $Y$.

**Missing Data Mechanism.** All variables, except for $Y$ and the three ancillary variables norm4, norm6, and norm8, were made missing via an MAR mechanism depending on $Y$ and one of the three ancillary variables. Let $Z_i = Y_i + \text{norm4}_i$. Suppose that $R_i = 0$ when norm1 is missing. We let

\[
P(R_i = 0) = \begin{cases} 
0.6 & \text{if } Z_i \text{ is in the top third of } Y + \text{norm4}, \\
0.1 & \text{if } Z_i \text{ is in the middle third of } Y + \text{norm4}, \\
0.6 & \text{if } Z_i \text{ is in the bottom third of } Y + \text{norm4}.
\end{cases}
\]

Similarly, missing values in norm2 and norm3 were generated using the same mechanism based on $Y + \text{norm4}$. Missing values in norm5 and bin1 were generated using the same method, except with $Y + \text{norm6}$ in place of $Y + \text{norm4}$. Missing values in norm7 and ord1 were similarly generated based on $Y + \text{norm8}$.

**MI Methods.** We assessed the imputation quality of the following implementations: default settings within mice (mice-default), classification and regression trees within mice (mice-cart), fast implementation of random forests within mice (mice-ranger), XGBoost with PMM (mixgb), and XGBoost with PMM and subsampling (mixgb-sub).

We used the default settings for methods from the R package mice, including the number of iterations maxit $= 5$, whereas the default number of iterations in mixgb and mixgb-sub is one. We believe that our method has good performance even when the number of iterations is smaller than that of mice. The performance of all methods was evaluated over 1000 simulation runs. Our evaluation was based on empirical bias, within-imputation variance, between-imputation variance and coverage.

**3.2. Evaluation Criteria**

Suppose $\beta$ is a $k \times 1$ column vector, representing the true coefficients of data generation model (3.1). Let $\hat{\beta}$ be the estimate of $\beta$ for the full simulated dataset and let $U$ be estimated variance-covariance matrix of $\hat{\beta}$. To investigate the imputation performance of MI methods, we used the following criteria based on Rubin (1987) and Brand et al. (2003):

**Empirical Bias.** For each simulation run, we obtained an MI

\[
\bar{\beta}_M = \frac{1}{M} \sum_{m=1}^{M} \hat{\beta}_m
\]

and $\hat{\beta}_m$ is the estimate of the $m$th
imputation. We are interested in the empirical bias $E_h[\beta_M] - \hat{\beta}$, where $E_h$ denotes averaging over 1000 simulation runs.

**Within-imputation Variance.** The within-variance component of the combined MI estimate is defined as $U_M = \frac{1}{M} \sum_{m=1}^{M} U_m^*$, where $U_m^*$ is the variance-covariance matrix of $\beta_m^*$. We compared the average of within-imputation variance over 1000 runs $\text{Var}_W := E_h[U_M]$ to $\text{Var}_W^{\text{target}} := U$, the estimated variance-covariance of $\hat{\beta}$.

**Between-imputation Variance.** The between-imputation variance is given by $B_M = \frac{1}{M-1} \sum_{m=1}^{M} (\beta_m^* - \overline{\beta}_M)(\beta_m^* - \overline{\beta}_M)^T$. The average adjusted between-imputation variance over 1000 simulations is $\text{Var}_B := \frac{1}{M} \sum_{m=1}^{M} (\beta_m^* - \overline{\beta}_M)(\beta_m^* - \overline{\beta}_M)^T$, where the factor $(1 + M^{-1})$ is used to adjust for $M$ being finite. We compared $\text{Var}_B$ to $\text{Var}_B^{\text{target}} := \text{Var}(\overline{\beta}_M)$, the empirical variance of the $\overline{\beta}_M$ over 1000 simulation runs.

**Coverage.** Coverage is the proportion of replications where the 95% confidence interval around the combined MI estimate $\overline{\beta}_M$ contains $\hat{\beta}$. The 95% confidence interval for $\hat{\beta}$ is calculated as $\overline{\beta}_M \pm t_{\nu,0.025} \sqrt{\frac{1}{1 + M^{-1}} B_M}$, where $t_{\nu,0.025}$ is the 97.5th percentile of a Student-t distribution with the degrees of freedom $\nu$ as defined in Rubin (1987).

### 3.3 Results

Simulation results over 1000 runs are summarized in Table 2. We visualized the simulation results of biases and imputation variances in Figures 3 and 4, respectively. The average computational time for each simulation run was 9 seconds for mice-default, 68 sec for mice-cart, 72 sec for mice-ranger, 11 sec for mixgb, and 12 sec for mixgb-sub.

To evaluate the biases of the imputation methods, we compared the estimates of coefficients obtained from different MI implementations against the empirical true estimates and
Table 2. Summary of simulation results.

| Method | Bias  | \( \frac{\text{Var}_T}{1000} \) | \( \frac{\text{Var}_T}{1000} \) | \( \frac{\text{Var}_W}{1000} \) | \( \frac{\text{Var}_W}{1000} \) | \( \frac{\text{Var}_T}{1000} \) | \( \frac{\text{Var}_T}{1000} \) | CI coverage% |
|--------|-------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|---------------|
| (Intercept) | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |
| norm1 | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |
| norm2 | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |
| norm3 | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |
| norm5 | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |
| norm7 | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |
| bin11 | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |
| ord11 | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |
| ord12 | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |
| I(norm1) | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |
| norm2:norm3 | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |
| norm5:bin11 | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |
| norm7:ord11 | | | | | | | | |
| mice-default | | | | | | | | |
| mice-cart | | | | | | | | |
| mice-ranger | | | | | | | | |
| mixgb | | | | | | | | |
| mixgb-sub | | | | | | | | |

(Continued)
Table 2. (Continued)

| Method          | Bias $\times 1000$ | $\text{Var}_T$ $\times 1000$ | $\text{Var}_W$ $\times 1000$ | $\text{Var}_B$ $\times 1000$ | $\text{Var}_{\text{target}}$ $\times 1000$ | CI coverage% |
|-----------------|---------------------|-------------------------------|-----------------------------|-------------------------------|---------------------------------------------|--------------|
| norm7:ord12     | -0.49               | 14.00                         | 7.31                        | 1.25                          | 6.70                                        | 2.86         |
| mice-default    | -0.29               | 11.04                         | 5.99                        | 1.25                          | 7.00                                        | 4.74         |
| mice-cart       | -0.33               | 14.10                         | 5.66                        | 1.25                          | 8.51                                        | 4.41         |
| mice-ranger     | -0.28               | 9.40                          | 6.68                        | 1.25                          | 4.77                                        | 5.43         |
| mixgb           | -0.28               | 9.88                          | 6.60                        | 1.25                          | 5.31                                        | 5.35         |
| mixgb-sub       | 0.003               | 0.03                          | 0.05                        | 0.001                         | 0                                           | 0.03         |
| Maximum Monte Carlo SE | 0.003 | —                             | —                           | —                             | —                                           | 2            |

NOTE: $\text{Var}_T = \text{Var}_W + \text{Var}_B$, $\text{Var}_{\text{target}} = \text{Var}_{\text{W, target}} + \text{Var}_{\text{B, target}}$.

**Figure 3.** Visualization of the simulation results on coefficient estimates of MI methods and complete case analysis (1000 runs). Vertical solid lines represent the empirical true values. It shows that both mixgb and mixgb-sub obtained less biased estimates than mice-default and complete case analysis, and as good as or better than mice-cart and mice-ranger.

**Figure 4.** Relative ratio of imputation variance and target imputation variance of each estimate using MI methods over 1000 simulation runs. The top panel compares performance on within-imputation variability, and the bottom panel shows performance on between-imputation variability. Relative ratios that are closer to zero signify better estimates in terms of imputation variance.
estimates found using complete cases. These estimates are displayed in Figure 3. As can be seen, complete case estimates had the largest biases for most coefficients compared to MI. Among MI methods, mice-default performed worse than tree-based methods, especially for interaction terms. This is consistent with the results obtained by Doove, van Buuren, and Dusseldorp (2014). Tree-based methods had rather similar performances. Out of 14 coefficients, mixgb-sub obtained 8 least biased estimates, mixgb obtained three least biased estimates and mice-cart also had 3. In this study, mixgb-sub had slightly smaller bias than mixgb for most coefficients, but the differences between these two methods were small.

We evaluated the performance of imputation variance, as discussed in Section 3.2, by plotting the relative ratios \(\frac{\text{Var}_W}{\text{Var}_W^{\text{target}}} - 1\) and \(\frac{\text{Var}_B}{\text{Var}_B^{\text{target}}} - 1\) in Figure 4. A relative ratio closer to zero indicates a better variance estimate. The top panel shows that all imputation methods obtained larger within-imputation variance than \(\text{Var}_W^{\text{target}}\). This was expected as we generated a large proportion of missing values in most variables in the dataset. Additionally, mice-default had larger within-imputation variances than other methods, followed by mice-ranger, while mice-cart tended to have the smallest within-imputation variances.

In terms of between-imputation variance, we can see that mixgb without subsampling was lower than the target variance for all coefficients. This confirms our expectation that when subsampling is not used, mixgb will underestimate the between-imputation variance because it does not incorporate the uncertainty of model parameters. With subsampling, mixgb-sub was closer to the target variance, and it had the best performance overall. On the other hand, mice-default, mice-cart, and mice-ranger obtained higher between-imputation variances than the target values. Among these, mice-ranger had the largest relative ratio in 11 of 14 estimates, indicating that it might have overestimated the between-imputation variances.

As can be seen in Table 2, all methods had zero coverage for terms norm5, norm7, ord12, norm5:bin11, and norm7:ord11. This was not surprising given our simulation settings. The coverage rates of the coefficients norm1, norm2, norm3 and norm4 were higher than those of other coefficients for all methods. Overall, mice-default had the lowest coverage rates for all interaction terms. Tree-based methods had similar coverage rates for most coefficients but mice-cart and mice-ranger performed considerably worse than mixgb and mixgb-sub for bin11 and ord11. The coverage rates of mixgb-sub were slightly better than those of mixgb.

4. Computational Time

This section compares the computational time for different MI implementations using real and simulated datasets. As both XGBoost and ranger provide multithreading, we utilized all 20 available threads for mixgb-cpu, mixgbsub-cpu and mice-ranger. For mixgb-gpu and mixgbsub-gpu, only a single CPU thread was used with GPU support. Since XGBoost with GPU uses tree method gpu_hist, we used tree method hist for the CPU version in order to eliminate alternative explanations for the difference in computational time between them. The default number of iterations maxit is five for the mice package, whereas in our package mixgb, the default maxit is one. In this comparison, we set maxit to one for all methods. In practice, mice uses a higher value of maxit and it will be slower. Apart from these, we used default settings for all methods. All code used in this section is available in the supplementary materials.

First, we chose three real-world datasets of varying sizes to gain a sense of the imputation time for different MI implementations. For each dataset, 30% missing data was created under the MCAR mechanism in two variables. Table 3 displays the average elapsed time to obtain five imputed datasets over ten replications.

| Dataset       | mice-default | mice-cart | mice-ranger | mixgb-cpu | mixgbsub-cpu | mixgb-gpu |
|---------------|--------------|-----------|-------------|-----------|--------------|-----------|
| Credit        | 3.7          | 18.7      | 8.2         | 3.0       | 3.9          | 3.7       |
| s.e.          | (0.03)       | (0.1)     | (0.02)      | (0.03)    | (0.09)       | (0.03)    |
| Allstate      | 2104.7       | 3651.9    | 537.9       | 62.8      | 60.7         | 66.0      |
| s.e.          | (11.5)       | (10.0)    | (1.4)       | (0.7)     | (0.5)        | (0.7)     |
| Higgs1M       | 112.2        | 3587.0    | 397.3       | 98.7      | 55.6         | 125.9     |
| s.e.          | (4.8)        | (5.2)     | (10.8)      | (4.5)     | (2.0)        | (4.1)     |

Table 3: Average computational time and standard error (s.e) for imputing real datasets using various MI implementations.

**NOTE:** Results are averaged over 10 repetitions.
Allstate and Higgs1M, but CPU versions were faster for imputing the smaller dataset Credit. Subsampling with mixgb took more time when just the CPU was used. However, subsampling required significantly less time when the GPU was used.

To further investigate the computational time of different MI implementations, we ran an experiment on simulated data. We generated datasets with sample sizes ranging from $10^3$ to $10^6$, with the number of features being 11, 31, and 51. Standard normal data were created for continuous data. For categorical data, all explanatory variables had three levels and the response variable was binary. For each dataset, 50% missing data in three variables were created via the MCAR mechanism. We did not evaluate mice-cart here as it would have required excessive computation time even for medium-sized datasets.

The log average time to obtain five imputations for continuous data and categorical data is shown in Figures 5 and 6, respectively, with all results averaged over ten repetitions. Overall, mice-default was the fastest for continuous datasets, but it performed poorly with increasing numbers of observations and features for categorical data. On the other hand, mice-ranger was reasonably fast for datasets with relatively small sample sizes; however, as the sample size increased, it scaled badly for both continuous and categorical data. To a certain extent, mice-ranger handled larger numbers of features more effectively than larger number of instances. Surprisingly, when working with the $10^6 \times 11$ dataset, mice-ranger took considerably longer to run compared to other methods. We will discuss this outlier later. All four versions of mixgb had good scalability across varying numbers of features and instances, and outperformed mice-default and mice-ranger for large categorical datasets, where the GPU versions of mixgb were the fastest methods. When using multithreading CPU, mixgb-cpu and mixgbsub-cpu were faster than their GPU counterpart for smaller datasets, but mixgb with GPU support performed better for large sample sizes. Our experiment also showed that using subsampling with CPU increased imputation time, especially when the sample size was large. However, the additional time required for subsampling was minor with GPU support.

The computational time for mice-ranger on the dataset with $10^6$ observations and 11 features may seem counterintuitive. It was the only place where the runtime was larger with fewer features. We profiled the mice-ranger imputation process for the three datasets with $10^6$ observations. We found that the time taken to fit the imputation model using ranger increased with the number of features, as expected, and the resulting tree for 11 features was simpler and shallower. However, the number of observations at each terminal node was much larger. When mice-ranger obtains predicted values from a given tree, it randomly samples one observation from the corresponding terminal node. For the dataset with 11 features, each tree had approximately 300 times as many observations at each terminal node on average, compared to the datasets with 31 and 51 features, which had similar numbers of observations at each terminal node. Since all datasets had $5 \times 10^5$ missing data points in each incomplete variable, the increase in time required to sample from terminal nodes was significant and outweighed the shorter time to build the tree. This effect caused the unusual time.

Figure 5. Computational time for imputing continuous data.

Figure 6. Computational time for imputing categorical data.
5. Data Example

The NWTS (US National Wilms’ Tumor Study) dataset (D’Angio et al. 1989) contains outcome variables (e.g., relaps and dead) as well as histology results for 3915 patients. Since it provides both the central lab histology (histol) and local institution histology (instit), it has been extensively used as an example to illustrate two-phase sampling designs (e.g., Breslow and Chatterjee 1999; Breslow et al. 2009; Chen and Lumley 2020). Other variables include study (3 or 4), stage (I-IV), the age and year at diagnosis, the weight of tumor (specwgt), the diameter of tumor (tumdiam), time to relapse (trel) and time to death (tsur). There are no missing values in the original dataset. We created missing data in histol, tumdiam and stage by an MAR mechanism dependent on relaps, instit, and specwgt. Missing data in histol and stage depended on relaps and instit; missing data in tumdiam depended on relaps and specwgt. Full details can be found in the supplementary materials. Five imputed datasets were then generated using mice-default, mice-cart, mice-ranger, mixgb and mixgb-sub. We fitted the following analysis model based on Kulich and Lin (2004):

\[
h(t) = h_0(t) \times \exp(\beta_1 \times \text{histol} + \beta_2 \times \text{age}_1 + \beta_3 \times \text{age}_2 + \beta_4 \times \text{stage} + \beta_5 \times \text{tumdiam} + \beta_6 \times \text{instit} + \beta_7 \times \text{histol} \times \text{age}_1 + \beta_8 \times \text{histol} \times \text{stage} \times \text{tumdiam}),
\]

where \(\beta_2\) and \(\beta_3\) are separate slopes for age younger than one year (\(\text{age}_1\)) and age one year or older (\(\text{age}_2\)). The variable stage is a binary indicator. It is coded 1 to indicate stage III-IV and 0 to indicate stage I-II. Table 4 summarizes the coefficient estimates obtained from full data, complete cases and MI methods. All MI implementations produced better estimates than complete case analysis.

In general, we know neither the missing data mechanism nor the true values of the missing data. The distribution of imputed values can differ noticeably from that of observed values. However, we can resort to diagnostic tools to check whether the imputed values are plausible and whether the imputation variability is reasonable. Our R package mixgb includes some functions for visualizing multiply-imputed values. Figure 7 compares the marginal distribution of the observed data, the masked true values, and five sets of imputed values using mean imputation in the variable tumdiam. In this case, there was no variability among five imputations. If we impute missing values by randomly sampling from the observed values of a variable, then the marginal distribution of that variable will resemble that of the observed values, like what is shown in the top panel of Figure 8. The bottom panel of Figure 8, however, illustrates that random sampling imputation failed to capture the relationship between variables tumdiam and specwgt.

On the other hand, Figure 9 shows that using MI methods can preserve the relationship between variables. The plot also indicates that mice-ranger had larger within and between imputation variability. Other types of diagnostic plots, which are omitted here, show similar patterns across all methods.

### Table 4. Coefficient estimates of NWTS data using full data, complete cases and MI methods.

| Estimates | Methods                      | Full data | Complete cases | mice-default | mice-cart | mice-ranger | mixgb | mixgb-sub |
|-----------|------------------------------|-----------|----------------|--------------|-----------|-------------|-------|-----------|
| \(\beta_1\) | 4.10                         | 6.75      | 4.10           | 4.32         | 4.39      | 4.19        | 4.26  |
| \(\beta_2\) | -0.66                        | -1.21     | -0.67          | -0.59        | -0.62     | -0.63       | -0.58 |
| \(\beta_3\) | 0.10                         | 0.14      | 0.10           | 0.10         | 0.10      | 0.10        | 0.10  |
| \(\beta_4\) | 1.35                         | 2.59      | 1.20           | 1.46         | 1.08      | 1.10        | 1.10  |
| \(\beta_5\) | 0.07                         | 0.17      | 0.07           | 0.07         | 0.08      | 0.05        | 0.06  |
| \(\beta_6\) | -2.64                        | -4.59     | -2.65          | -2.95        | -2.98     | -2.71       | -2.79 |
| \(\beta_7\) | -0.06                        | -0.06     | -0.04          | -0.04        | -0.05     | -0.04       | -0.04 |
| \(\beta_8\) | -0.08                        | -0.13     | -0.07          | -0.09        | -0.06     | -0.05       | -0.06 |

**Figure 7.** The comparison of the marginal distributions of the observed values, the masked true values, and five sets of imputed values using mean imputation for variable tumdiam.
datasets but scaled well for larger datasets, with GPU versions being the fastest. Subsampling increased the imputation time notably when using only the CPU, especially for large sample sizes. Nevertheless, this increase was negligible when the GPU was used.

To the best of our knowledge, this is the first attempt to implement and evaluate MI using XGBoost with predictive mean matching and subsampling. We show that mixgb and mixgb-sub can achieve imputation performances as good as that of mice-cart, which was previously regarded as the best method.
for capturing complex data structures (Doove, van Buuren, and Dusseldorp 2014). We consider mice-ranger, mixgb, and mixgb-sub to be better alternatives for MI, since CART is susceptible to overfitting and does not generalize well in real-world data scenarios. Moreover, using mice-cart for large-scale imputations can be time-consuming. For smaller datasets, mice-ranger was faster than mixgb-cpu or mixgb-gpu; however, it was significantly slower for larger datasets. In addition, mice-ranger can take an excessive amount of time to run under certain circumstances, such as the case we described in Section 4. Our findings suggest that mixgb is a promising automated MI framework for large and complex datasets. In this study, we found mixgb-sub-gpu to be the best among all evaluated MI methods due to its speed and imputation performance.

Our results may be limited to simulation studies similar to the one presented in this article and may not generalize to other scenarios. Under our simulation settings, a large proportion of missing values were created via an MAR mechanism that made complete case analysis invalid. If the data is missing completely at random or missingness is only weakly associated with the available variables, then complete case analysis can often achieve better results than MI methods. It is also worth noting that the speed and quality of imputation generally depend on hyperparameter settings, such as the number of trees in mice-ranger (default 10), the number of boosting rounds in mixgb and mixgb-sub (default 100), and the number of threads used when a multicore processor is available. Future work should evaluate imputation performance with diverse datasets and a wider range of hyperparameter configurations.

Computation Details

The simulation study in Section 3 on imputation performance was run using R 4.0.2 (R Core Team 2022) on a Ubuntu 18.04 server with a 64-core Intel Xeon Gold 6142 CPU processor and 187GB of RAM, without the use of GPU. Simulation tasks for all methods were run in parallel. The evaluation of computational time in Section 4 was run using R 4.2.1 (R Core Team 2022) on a 64-bit Windows 10 PC with 64 GB of RAM, an Intel i7-12700K CPU, and an Nvidia GeForce RTX 3080 Ti GPU. Each imputation method was tested separately to avoid interference between tasks. R package mice (van Buuren and Groothuis-Oudshoorn 2011) version 3.14.0 and R package mixgb (Deng 2023) version 1.0.1 were used in this article. Computational time was recorded using the R package microbenchmark (Mersmann 2021) and visualization of simulation results used the R package ggplot2 (Wickham 2016).

Supplementary Materials

All supplementary materials are contained in a single archive supplement.zip. For more details, please refer to the readme.md file in the zip file.

Acknowledgments

The authors would like to express their sincere gratitude to the Editor, Associate Editor and two Reviewers for their insightful comments and suggestions.

Disclosure Statement

The authors report there are no competing interests to declare.

References

Awada, Z., Bouaoun, L., Nasr, R., Tifarli, A., Cuenin, C., Akika, R., Boustany, R.-M., Makoukji, J., Tamim, H., Zghib, N. K., and Ghantous, A. (2021), “LINE-1 Methylation Mediates the Inverse Association Between Body Mass Index and Breast Cancer Risk: A Pilot Study in the Lebanese Population,” Environmental Research, 197, 111094. [352]

Baldi, P., Sadowski, P., and Whiteson, D. (2014), “Searching for Exotic Particles in High-Energy Physics with Deep Learning,” Nature Communications, 5, 4306. [358]

Brand, J. P., van Buuren, S., Groothuis-Oudshoorn, K., and Gel savage, E. S. (2003), “A Toolkit in SAS for the Evaluation of Multiple Imputation Methods,” Statistica Neerlandica, 57, 36–45. [354]

Breiman, L. (2001), “Random Forests,” Machine Learning, 45, 5–32. [352]

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984), Classification and Regression Trees, Belmont, CA: Wadsworth. [352]

Breslow, N. E., and Chatterjee, N. (1999), “Design and Analysis of Two-Phase Studies with Binary Outcome Applied to Wilms Tumour Prognosis,” Journal of the Royal Statistical Society, Series C, 48, 457–468. [360]

Breslow, N. E., Lumley, T., Ballantyne, C. M., Chambless, L. E., and Kulich, M. (2009), “Using the Whole Cohort in the Analysis of Case-Cohort Data,” American Journal of Epidemiology, 169, 1398–1405. [360]

Chen, T., and Guestrin, C. (2016), “XGBoost: A Scalable Tree Boosting System,” in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York: Association for Computing Machinery, pp. 785–794. [352,360]

Chen, T., and Lumley, T. (2020), “Optimal Multivariate Sampling for Regression Modeling in Two-Phase Designs,” Statistics in Medicine, 39, 4912–4921. [360]

D’Angio, G. J., Breslow, N., Beckwith, J. B., Evans, A., Baum, E., Delorimier, A., Fernbach, D., Harbovsky, E., Jones, B., Kelalis, P. et al. (1989), “Treatment of Wilms’ Tumor. Results of the Third National Wilms’ Tumor Study,” Cancer, 64, 349–360. [360]

Deng, Y. (2023), mixgb: Multiple Imputation Through XGBoost, R package version 1.0.1. [353,362]

Doove, L. L., van Buuren, S., and Dusseldorp, E. (2014), “Recursive Partitioning for Missing Data Imputation in the Presence of Interaction Effects,” Computational Statistics & Data Analysis, 72, 92–104. [352,358,360,362]

Kaggle (2016), Allstate Claims Severity Dataset, available at https://www.kaggle.com/competitions/allstate-claims-severity/data. [358]

Kelly, M., Longjoh, R., and Nottingham, K. (2023), The UCI Machine Learning Repository, available at https://archive.ics.uci.edu. [358]

Kulich, M., and Lin, D. Y. (2004), “Improving the Efficiency of Relative-Risk Estimation in Case-Cohort Studies,” Journal of the American Statistical Association, 99, 832–844. [360]

Little, R. J. (1988), “Missing-Data Adjustments in Large Surveys,” Journal of Business & Economic Statistics, 6, 287–296. [352,353,360]

Mersmann, O. (2021), microbenchmark: Accurate Timing Functions, R package version 1.4.9. [362]

R Core Team (2022), R: A Language and Environment for Statistical Computing, Vienna, Austria: R Foundation for Statistical Computing. [362]

Rubin, D. B. (1978), “Multiple Imputations in Sample Surveys-A Phenomenological Bayesian Approach to Nonresponse,” in Proceedings of the Survey Research Methods Section, American Statistical Association, pp. 20–34. [352]

(1986), “Statistical Matching Using File Concatenation with Adjusted Weights and Multiple Imputations,” Journal of Business & Economic Statistics, 4, 87–94. [353]

(1987), Multiple Imputation for Nonresponse in Surveys, Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics, New York: Wiley. [352,354,355]

ORCID

Yongbi Deng http://orcid.org/0000-0001-5845-859X

Thomas Lumley http://orcid.org/0000-0003-4255-5437
Stekhoven, D. J., and Bühlmann, P. (2012), "MissForest–Non-Parametric Missing Value Imputation for Mixed-Type Data," *Bioinformatics*, 28, 112–118. [352]

Su, Y. S., Gelman, A., Hill, J., and Yajima, M. (2011), "Multiple Imputation with Diagnostics (mi) in R: Opening Windows into the Black Box," *Journal of Statistical Software*, 45, 1–31. [352]

van Buuren, S. (2018), *Flexible Imputation of Missing Data* (2nd ed.), Boca Raton, FL: Chapman & Hall/CRC Press. [353]

van Buuren, S., and Groothuis-Oudshoorn, K. (2011), "Mice: Multivariate Imputation by Chained Equations in R," *Journal of Statistical Software*, 45, 1–67. [352,362]

Wendt, F. R., Pathak, G. A., Levey, D. F., Nuñez, Y. Z., Overstreet, C., Tyrrell, C., Adhikari, K., De Angelis, F., Tylee, D. S., Goswami, A., Krystal, J. H., Abdallah, C. G., Stein, M. B., Kranzler, H. R., Gelernter, J., and Polimanti, R. (2021), "Sex-Stratified Gene-by-Environment Genome-Wide Interaction Study of Trauma, Posttraumatic-Stress, and Suicidality," *Neurobiology of Stress*, 14, 100309. [352]

Wickham, H. (2016), *ggplot2: Elegant Graphics for Data Analysis*, New York: Springer-Verlag. [362]

Wright, M. N., and Ziegler, A. (2017), "Ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R," *Journal of Statistical Software*, 77, 1–17. [352]

Yeh, I.-C., and Lien, C.-h. (2009), "The Comparisons of Data Mining Techniques for the Predictive Accuracy of Probability of Default of Credit Card Clients," *Expert systems with applications*, 36, 2473–2480. [358]

Zhang, X., Yan, C., Gao, C., Malin, B., and Chen, Y. (2019), "XGBoost Imputation for Time Series Data," in *2019 IEEE International Conference on Healthcare Informatics (ICHI)*, pp. 1–3. [353]