Imputation procedures in surveys using nonparametric and machine learning methods: an empirical comparison

Mehdi Dagdoug[(1)], Camelia Goga[(1)] and David Haziza[(2)]
(1) Université de Bourgogne Franche-Comté,
Laboratoire de Mathématiques de Besançon, Besançon, FRANCE
(2) Université de Montréal, Département de mathématiques et de statistique,
Montréal, CANADA

November 30, 2020

Abstract

Nonparametric and machine learning methods are flexible methods for obtaining accurate predictions. Nowadays, data sets with a large number of predictors and complex structures are fairly common. In the presence of item nonresponse, nonparametric and machine learning procedures may thus provide a useful alternative to traditional imputation procedures for deriving a set of imputed values used next for the estimation of study parameters defined as solution of population estimating equation. In this paper, we conduct an extensive empirical investigation that compares a number of imputation procedures in terms of bias and efficiency in a wide variety of settings, including high-dimensional data sets. The results suggest that a number of machine learning procedures perform very well in terms of bias and efficiency.

Key words: Additive models; Bayesian additive regression trees (BART); CART; Cubist algorithm; Ensemble Methods; Nearest Neighbour; Item nonresponse; Random forest; Support vector regression (SVR); Survey data; Statistical learning; Tree boosting.

1 Introduction

In the last decade, the interest in machine learning methods has been growing in national statistical offices (NSO). These data-driven methods provide flexible tools for obtaining accurate

Mehdi Dagdoug's research was supported by grants of the region of Franche-Comté and Médiamétrie.

predictions. The increasing availability of data sources (e.g., big data sources and satellite information) provides a rich pool of potential predictors that may be used to obtain predictions at different stages of a survey. These stages include the nonresponse treatment stage (e.g., propensity score weighting and imputation) and the estimation stage (e.g., model-assisted estimation and small area estimation). The imputation stage is the focus of the current paper.

Item nonresponse refers to the presence of missing values for some, but not all, survey variables. Frequent causes of item nonresponse include refusal to answer a sensitive question (e.g., income) and edit failures. The most common way of treating item nonresponse in NSOs is to replace a missing value with a single imputed value, constructed on the basis of a set of p explanatory variables, $\mathbf{X} = (X_1, \ldots, X_p)$, available for both respondents and nonrespondents. A variety of imputation procedures are available, ranging from simple (e.g., mean, historical and ratio imputation) to more complex (e.g., nonparametric procedures); e.g., see Chen and Haziza (2019) for an overview of imputation procedures in surveys. Every imputation procedure makes some (implicit of explicit) assumptions about the distribution of the variable Y requiring imputation. This set of assumptions is often referred to as an imputation model. At the imputation stage, it is therefore important to identify and include in the model all the appropriate explanatory variables that are predictive of the variable requiring imputation and determine a suitable model describing the relationship between Y and the set of explanatory variables $\mathbf{X}$.

We distinguish parametric imputation procedures from nonparametric imputation procedures. In parametric imputation, the shape of the relationship between Y and $\mathbf{X}$ is predetermined; e.g., linear and generalized linear regression models. However, point estimators based on parametric imputation procedures may suffer from bias if the functional form is misspecified or if the vector $\mathbf{X}$ fails to include interactions or predictors accounting for curvature. In contrast, with nonparametric methods, the shape of the relationship between Y and $\mathbf{X}$ is left unspecified. These methods have the ability to capture nonlinear trends in the data and tend to be robust to the non-inclusion of interactions or predictors accounting for curvature.

Commonly used nonparametric methods include kernel smoothing, local polynomial regression and spline-based regression models. While these methods provide some robustness against model misspecification, they tend to breakdown when the number predictors is large,

a problem known as the curse of dimensionality. To mitigate this problem, one may employ additive models (Hastie and Tibshirani, 1986). However, when the dimension of $\mathbf{X}$ is very large, these models tend to fail and machine learning methods may provide an interesting alternative. The class of machine learning methods, that includes tree-based models such as random forests and boosting methods, provide more flexible approaches able to adapt to complex non-linear and non-additive relationships between the survey variable requiring imputation and a set of predictors. These methods may also prove useful in the case of large data sets exhibiting a large number of observations on a large number of variables. Many machine learning procedures are relatively computationally efficient and can produce accurate predictions by offering the user a kind of automatic variable selection that may prove useful in a high-dimensional setting.

However, both a theoretical treatment and an empirical comparison of machine learning imputation procedures in the context of missing survey data are currently lacking. In this paper, we aim to fill the latter gap by conducting an extensive simulation study that investigates the performance of several nonparametric and machine learning procedures in terms of bias and efficiency. To that end, we generated several finite populations with relationships between Y and $\mathbf{X}$, ranging from simple to complex and generated the missing values according to several nonresponse mechanisms. We also considered both a low-dimensional and high dimensional settings. The simulation setup and the models are described in Section 4. We restricted our attention to population totals (Section 4) and population quantiles (Section 5) as the target parameters. The following procedures were included in our comparisons: the score method (Little, 1986; Haziza and Beaumont, 2007), K nearest-neighbour (Chen and Shao, 2000), additive models based on B-spline regression, regression trees (Breiman et al., 1984), random forests (Breiman, 2001), tree-based boosting methods (Friedman, 2001) including XGBoost (Chen and Guestrin, 2016) and Bayesian additive regression trees (Chipman et al., 2010), the cubist algorithm (Quinlan et al., 1992; Quinlan, 1993) and support vector regression (Vapnik, 1998, 2000). In Section 3, we describe these models and the corresponding imputation procedures.

In recent years, machine learning procedures have received some attention in a survey sampling context. In the ideal situation of 100% response, the theoretical properties of model-assisted estimation procedures based on regression trees (McConville and Toth, 2019)

and random forests (Dagdoug et al., 2020a) have been recently established. Dagdoug et al. (2020b) studied the theoretical properties of point and variance estimators based on random forests in a context of imputation for item nonresponse and data integration; see also Tipton et al. (2013); De Moliner and Goga (2018) for applications of random forests in surveys. A number of empirical investigations have been conducted to assess the performance of machine learning procedures in a context of propensity score estimation for unit nonresponse; e.g., Lohr et al. (2015), Gelein (2017) and Kern et al. (2019).

The machine learning procedures described in Section 3 slightly differ from their traditional implementation because of the inclusion of the sampling weights in the construction of imputed values. However, it should be noted that most of the machine learning software packages for obtaining predicted values assume simple random sampling and cannot handle unequal weights. Modifying machine learning algorithms to account for unequal weights may prove challenging. When the design features (e.g., sampling weights, stratum indicators, etc.) are related to the survey variable requiring imputation, failing to incorporate them in the models may lead to biased estimators. To cope with this issue, we suggest to include all the appropriate design variables in the specification of the model. Standard machine learning software packages may then be safely used for creating a set of imputed values. In Section 4, we use Poisson sampling with inclusion probabilities proportional to a size variable X to select repeated samples from the finite population. The size variable X being related to the variable requiring imputation, including the X-variable in the specified models led to satisfactory results.

2 Preliminaries

Consider a finite population $U = \{1, 2, ..., N\}$ of size N. Let Y denote a survey variable and y_i be the y-values attached to unit i, $i = 1, \cdots, N$. We are interested in estimating (i) the finite population total of the y-values, $t_y = \sum_{i \in U} y_i$ and (ii) the finite population quantile of order γ defined as $\mathcal{Q}_\gamma := \inf \{t \in \mathbb{R}; F_N(t) \geqslant \gamma\}$, where

$$F_N(t) = \sum_{i \in U} \mathbb{1}\left(y_i \leqslant t\right)/N$$

denotes the finite population distribution function.

From U, we select a sample S, of size n, according to a sampling design $\mathcal{P}(S = s)$ with first-order inclusion probabilities $\pi_i = Pr(i \in S)$.

A complete data estimator of t_y is the well-known Horvitz-Thompson estimator

$$\widehat{t}_\pi = \sum_{i \in S} \frac{y_i}{\pi_i}, \tag{1}$$

which is design-unbiased for t_y provided that $\pi_i > 0$ for all $i \in U$. A complete data estimator of the finite population quantile $\mathcal{Q}_\gamma$ is given by

$$\widehat{\mathcal{Q}}_\gamma := \inf \left\{ t \in \mathbb{R}; \widehat{F}(t) \geqslant \gamma \right\}, \tag{2}$$

where

$$\widehat{F}(t) = \frac{1}{\widehat{N}} \sum_{i \in S} \frac{\mathbb{1}(y_i \leqslant t)}{\pi_i} \tag{3}$$

with $\widehat{N} = \sum_{i \in S} 1/\pi_i$ denoting the Horvitz-Thompson estimator of the population size N. Under mild regularity conditions (Wang and Opsomer, 2011), the complete data estimator $\widehat{\mathcal{Q}}_\gamma$ is design-consistent for $\mathcal{Q}_\gamma$.

In practice, the Y-variable may be prone to missing values. Let r_i be a response indicator such that $r_i = 1$ if y_i is observed and $r_i = 0$, otherwise. Let $S_r = \{i \in S;\ r_i = 1\}$ denote the set of respondents, of size n_r, and $S_m = \{i \in S;\ r_i = 0\}$ the set of nonrespondents, of size n_m, such that $S_r \cup S_m = S$ and $n_r + n_m = n$. Available to the imputer is the data $(y_i, \mathbf{x}_i)$ for $i \in S_r$ as well as the values of the vector $\mathbf{x}_i$ for $i \in S_m$.

Let $\widehat{y}_i$ be the imputed value used to replace the missing value y_i and

$$\widetilde{y}_i = r_i y_i + (1 - r_i)\, \widehat{y}_i$$

be the ith value of the Y-variable after imputation. Point estimators of t_y and $\mathcal{Q}_\gamma$ after imputation, often referred to as imputed estimators, are readily obtained from the complete data estimators (1) and (2) by replacing y_i with $\widetilde{y}_i$. This leads to

$$\widehat{t}_{imp} = \sum_{i \in S} \frac{\widetilde{y}_i}{\pi_i} \tag{4}$$

and

$$\widehat{\mathcal{Q}}_{\gamma,imp} = \inf \left\{ t \in \mathbb{R}; \widehat{F}_{imp}(t) \geqslant \gamma \right\}, \tag{5}$$

where

$$\widehat{F}_{imp}(t) = \frac{1}{\widehat{N}} \sum_{i \in S} \frac{\mathbb{1}\left(\widetilde{y}_i \leqslant t\right)}{\pi_i} \tag{6}$$

denotes the imputed estimator of $F_N(t)$.

Remark 2.1. *The population total t_y, the distribution function $F_N(t)$ and the quantile of order γ, $\mathcal{Q}_\gamma$, may all be obtained as the solution of the following census estimating equation (Binder, 1983; Chen and Haziza, 2019):*

$$U_N(\theta_N) = \sum_{i \in U} u(y_i; \theta_N) = 0, \tag{7}$$

where θ_N is a generic notation denoting a finite population parameter and $u(y_i;\theta)$ is a function of θ_N. We assume that a solution to (7) exists and is unique. For instance, the population total t_y can be obtained as a solution of (7) with $u(y_i;\theta_N) = y_i - n^{-1}\pi_i\theta_N$; the finite population distribution function $F_N(t)$ can be obtained as a solution of (7) with $u(y_i;\theta_N) = \mathbb{1}\left(y_i \leqslant t\right) - \theta_N$. Finally, the quantile $\mathcal{Q}_\gamma$ of order γ can be obtained as a solution of (7) with $u(y_i;\theta_N) = \mathbb{1}\left(y_i \leqslant \theta_N\right) - \gamma$. Other finite population parameters can be obtained as a solution of (7); e.g., see Chen and Haziza (2019). The imputed estimators $\widehat{t}_{imp}$, $\widehat{\mathcal{Q}}_{\gamma,imp}$ and $\widehat{F}_{imp}(t)$ given respectively by (4)-(6) can be obtained by solving the following sample estimating equation:

$$\widehat{U}_{imp}(\widehat{\theta}_{imp}) = \sum_{i \in S} \frac{1}{\pi_i} u(\widetilde{y}_i; \widehat{\theta}_{imp}) = 0,$$

where $\widehat{\theta}_{imp}$ denotes an imputed estimator of θ_N.

To construct the imputed values $\widehat{y}_i$, we postulate the following imputation model ξ:

$$\begin{aligned} &\mathbb{E}_\xi(y_i|\mathbf{x}_i) = f(\mathbf{x}_i), \\ &\mathbb{V}_\xi\left(y_i|\mathbf{x}_i\right) = \sigma_i^2, \\ &\mathbb{C}ov_\xi\left(y_i, y_j|\mathbf{x}_i, \mathbf{x}_j\right) = 0 \quad \text{for} \qquad i \neq j, \end{aligned} \tag{8}$$

where f is an unknown function. Often, the variance structure σ_i^2 is assumed to have the form $\sigma_i^2 = \sigma^2 a_i$, where $a_i > 0$ is a known coefficient attached to unit i and σ^2 is an unknown parameter.

We assume that the data are Missing At Random (Rubin, 1976):

$$f(y_i|\mathbf{x}_i, r_i = 1) = f(y_i|\mathbf{x}_i, r_i = 0). \tag{9}$$

That is, we assume that the distribution of Y given $\mathbf{x}$ is the same for both respondents and nonrespondents. If Condition (9) holds, the imputed values can be safely generated from $f(y_i|\mathbf{x}_i, r_i = 1)$, which can be estimated from the observed data. In the context of imputation, the properties of point estimators are evaluated with respect to the joint distribution induced by the imputation, the sampling design and the unknown nonresponse mechanism. This framework is often referred to as the ξpq-framework (Chen and Haziza, 2019). Note that our simulation setup in Section 4 is consistent with the ξpq-framework as the simulation process involves (i) generating repeated finite populations; (ii) selecting a sample from each of population and (iii) generating a set of response indicators in each sample.

Deterministic imputation consists of replacing the missing y_i by $\widehat{y}_i = \widehat{f}(\mathbf{x}_i)$, where $\widehat{f}$ is an estimator of the unknown regression function f based on the responding units $i \in S_r$. However, deterministic imputation methods tend to distort the distribution of the survey variable Y requiring imputation, potentially leading to biased estimators of quantiles (Haziza, 2009; Chen and Haziza, 2019). To cope with this issue, one can recourse to random imputation that consists of adding an appropriate amount of random noise to the deterministic value $\widehat{f}(\mathbf{x}_i)$. More specifically, let $e_j := \widehat{\sigma}_j^{-1}\{y_j - \widehat{f}(\mathbf{x}_j)\}$ for $j \in S_r$, where $\widehat{\sigma}_j$ of an estimator of σ_j (see Remark 2.2 below). We define the standardized residual

$$\widetilde{e}_j = e_j - \frac{\sum_{\ell \in S_r} w_\ell e_\ell}{\sum_{\ell \in S_r} w_\ell}, \quad j \in S_r.$$

In the case of random imputation, the missing y_i is replaced by

$$\widehat{y}_i = \widehat{f}(\mathbf{x}_i) + \widehat{\sigma}_i \widehat{e}_i, \tag{10}$$

where $\widehat{e}_i$ is selected at random from the set of standardized residuals $\{\widetilde{e}_j\}_{j \in S_r}$ with probability $w_j / \sum_{\ell \in S_r} w_\ell$.

Remark 2.2. *To obtain an estimator $\widehat{\sigma}_i$ of σ_i, one can postulate a model $\mathbb{E}(\epsilon_i^2 \mid \mathbf{x}_i) = m(\mathbf{x}_i)$, where m is an unknown function. An estimator $\widehat{\sigma}_i^2$ of σ_i^2 is obtained by fitting a parametric*

or a nonparametric procedure with the square residuals e_i^2 as the response and $\mathbf{x}_i$ as the set of predictors.

In Section 3, except for the parametric imputation procedure discussed in Section 3.1, all the other procedures (Section 3.2-3.9) are nonparametric. In Section 4, these procedures are compared empirically in terms of bias and efficiency under a variety of settings.

3 A description of imputation methods

3.1 Parametric regression imputation

Parametric regression assumes that the first moment (8) is given by

$$\mathbb{E}_\xi(y_i|\mathbf{x}_i) = f(\mathbf{x}_i, \boldsymbol{\beta}), \tag{11}$$

where $\boldsymbol{\beta}$ is a vector of coefficients to be estimated and $f(\cdot)$ is a predetermined function. An estimator $\widehat{\boldsymbol{\beta}}$ of $\boldsymbol{\beta}$ is obtained by solving the following estimating equations based on the responding units:

$$\sum_{i \in S_r} \frac{w_i}{\sigma_i^2} \{y_i - f(\mathbf{x}_i, \boldsymbol{\beta})\} \frac{\partial f(\mathbf{x}_i, \boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = 0, \tag{12}$$

where $w_i > 0$ is a weight attached to element i. Common choices for w_i include $w_i = 1$ and $w_i = \pi_i^{-1}$ (Chen and Haziza, 2019). The imputed value $\widehat{y}_i$ under deterministic parametric regression imputation is given by

$$\widehat{y}_i = f(\mathbf{x}_i, \widehat{\boldsymbol{\beta}}), \quad i \in S_m. \tag{13}$$

A special case of (13) is $f(\mathbf{x}_i, \boldsymbol{\beta}) = \mathbf{x}_i^\top \boldsymbol{\beta}$, which corresponds to the customary linear regression model. In this case, the imputed value (13) reduces to

$$\widehat{y}_i = \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}, \quad i \in S_m, \tag{14}$$

where

$$\widehat{\boldsymbol{\beta}} = \left(\sum_{j \in S_r} w_j \sigma_j^{-2} \mathbf{x}_j \mathbf{x}_j^\top \right)^{-1} \sum_{j \in S_r} w_j \sigma_j^{-2} \mathbf{x}_j y_j. \tag{15}$$

The imputed value $\widehat{y}_i$ given by (14) can be written as a weighted sum of the respondent y-values:

$$\widehat{y}_i = \sum_{j \in S_r} w'_{ij} y_j, \quad i \in S_m, \tag{16}$$

where $w'_{ij} = \mathbf{x}_i^\top \left(\sum_{j' \in S_r} w_{j'} \sigma_{j'}^{-2} \mathbf{x}_{j'} \mathbf{x}_{j'}^\top \right)^{-1} w_j \sigma_j^{-2} \mathbf{x}_j$. If the intercept is among the X-variables, then $\sum_{j \in S_r} w'_{ij} = 1$ for all $i \in S_m$. A random counterpart of (13) is given by (10).

Another important special case of (13) is the logistic regression model,

$$f(\mathbf{x}_i, \boldsymbol{\beta}) = \exp(\mathbf{x}_i^\top \boldsymbol{\beta}) / (1 + \exp(\mathbf{x}_i^\top \boldsymbol{\beta})),$$

which can be used for modeling binary variables. An estimator of $\boldsymbol{\beta}$ is obtained by solving (12), which requires a numerical algorithm such as the Newton-Raphson procedure. To eliminate the possibility of an impossible imputed value, a missing value to a $0-1$ variable is typically imputed by $\widehat{y}_i$, where $\widehat{y}_i$ is a realization of a Bernoulli variable with parameter $f(\mathbf{x}_i, \widehat{\boldsymbol{\beta}})$.

Under deterministic or random parametric regression imputation, the imputed estimator $\widehat{t}_{imp}$ is consistent for t_y provided that the first moment of the imputation model (8) is correctly specified. However, this type of imputation may lead to biased estimators of quantiles. In contrast, the use of a random parametric regression imputation procedure tend to preserve the distribution of the variable requiring imputation, leading to valid estimators; see Chen and Haziza (2019) for a discussion.

3.2 Imputation classes : the score method

The score method (Little, 1986; Haziza and Beaumont, 2007) consists of partitioning the sample S into H (say) imputation classes and imputing the missing values within each class independently from one class to another. It can be implemented as follows:

Step 1: For all $i \in S$, compute the preliminary values $\widehat{y}_i^{LR} = \mathbf{x}_i^\top \widehat{\boldsymbol{\beta}}$, where $\widehat{\boldsymbol{\beta}}$ is given by (15).

Step 2: Compute the empirical quantiles $q_1, q_2, \ldots, q_{H-1}$ of order $1/H, 2/H, \ldots, (H-1)/H$ of the $\widehat{y}^{LR}$-values.

Step 3: Split the sample S into H classes, $C_1, \ldots, C_h, \ldots, C_H$, such that

$$C_h = \left\{ i \in S : \widehat{y}_i^{LR} \in [q_{h-1}; q_h) \right\}, \quad h = 1, \ldots, H,$$

with $q_0 = -\infty$ and $q_H = +\infty$.

It is common practice to use either mean imputation or random hot-deck imputation within classes. For mean imputation, the imputed value for missing y_i in the hth imputation class is given by

$$\widehat{y}_i = \frac{\sum_{j \in S_r \cap C_h} w_j y_j}{\sum_{j \in S_r \cap C_h} w_j} = \sum_{j \in S_r \cap C_h} w'_{ij} y_j, \quad i \in S_m \cap C_h,$$

where $w'_{ij} = w_j / \sum_{j' \in S_r \cap C_h} w_{j'}$ are the same for all $i \in S_m \cap C_h$ and $\sum_{j \in S_r \cap C_h} w'_{ij} = 1$ for all $i \in S_m \cap C_h$. For random hot-deck imputation, the imputed value is given by $\widehat{y}_i = y_j$, where the donor $j \in S_r \cap C_h$ is selected at random from the set of donors belonging to the hth imputation class with probability $w_j / \sum_{j' \in S_r \cap C_h} w_{j'}$. Note that random hot-deck imputation within classes can be viewed as mean imputation within classes with added residuals.

3.3 K-nearest neighbours imputation

K-nearest neighbour (KNN) imputation is one of the simplest and widely used nonparametric imputation procedures. No explicit assumption is made about the regression function f relating Y and $\mathbf{X}$. KNN imputation consists of replacing the missing value of a recipient by the weighted average of the y-values of its K closest respondents in terms of the X-variables.

Nearest-neighbour (NN) imputation corresponds to the limiting case of KNN obtained with $K = 1$. NN is a donor imputation belonging to the class of hot-deck procedures (Chen and Shao, 2000) since a missing value is replaced by an actual respondent y-value from the same file. NN imputation is especially useful for imputing categorical or discrete Y-variables; e.g., see Chen and Shao (2000), Beaumont and Bocci (2009) and Yang and Kim (2019).

Let $\mathcal{N}_K(i)$ be the set of K responding units closest to $\mathbf{x}_i$. Any distance function in $\mathbb{R}^p$ may be used to measure the closeness between two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$. In the simulation study presented in Section 4, we used the customary Euclidean distance. The KNN imputed value for missing y_i is given by

$$\widehat{y}_i = \frac{\sum_{j \in \mathcal{N}_K(i) \cap S_r} w_j y_j}{\sum_{j \in \mathcal{N}_K(i) \cap S_r} w_j}, \quad i \in S_m.$$

The imputed value $\widehat{y}_i$ obtained with KNN can be written as a weighted sum of the respondent y-values:

$$\widehat{y}_i = \sum_{j \in S_r} w'_{ij} y_j, \quad i \in S_m,$$

where $w'_{ij} = w_j \mathbb{1}(j \in \mathcal{N}_K(i)) / \sum_{j' \in \mathcal{N}_K(i) \cap S_r} w_{j'}$ for $j \in S_r$ with $\sum_{j \in S_r} w'_{ij} = 1$. KNN imputation is a locally weighted procedure since the respondents j lying not close enough to unit i with respect to the X-variables are assigned a weight equal to 0; i.e., $w'_{ij} = 0$. The indicator function in the expression of w'_{ij} can be replaced by a one-dimensional continuous kernel smoother $\mathcal{K}_h$, whose role is to control the size of the weight through a tuning parameter h : the units j lying farther from unit i will be assigned a smaller weight than units lying close to it (Hastie et al., 2011).

The imputed estimator under KNN imputation tends to be inefficient when the dimension p of $\mathbf{x}$ is large. Indeed, as p increases, it becomes more difficult to find enough respondents around the point at which we aim to make a prediction. This phenomenon is known as the curse of dimensionality (Hastie et al., 2011, Chap. 1) for a more in-depth discussion ok the KNN procedure. Also, it suffers from a model bias which is of order $(K/n)^{1/p}$. Nearest-neigbour imputation for missing survey data has been considered in Chen and Shao (2000), Beaumont and Bocci (2009) and Yang and Kim (2019).

3.4 B-splines and additive model nonparametric regression

Spline regression is a flexible nonparametric method for fitting non-linear functions $f(\cdot)$. It can be viewed as a simple extension of linear models. For simplicity, we start with a univariate X-variable supported on the interval $[0; 1]$. A spline function of order v with κ equidistant interior knots, $0 = \xi_0 < \xi_1 < ... < \xi_\kappa < \xi_{\kappa+1} = 1$, is a piecewise polynomial of degree $v - 1$ between knots and smoothly connected at the knots. These spline functions span a linear space of dimension of $q = v + \kappa$ with a basis function given by the B-splines functions:

$$B_\ell(x) = (\xi_\ell - \xi_{\ell - v}) \sum_{l=0}^{v} (\xi_{\ell - l} - x)_+^{v-1} / \Pi_{r=0, r \neq l}^{v} (\xi_{\ell - l} - \xi_{\ell - r}), \quad \ell = 1, \ldots, q,$$

where $(\xi_{\ell-l} - x)_+^{v-1} = (\xi_{\ell-l} - x)^{v-1}$ if $\xi_{\ell-l} \geq x$ and equal to zero, otherwise; see (Schumaker, 1981; Dierckx, 1993). The B-spline basis is appealing because the basis functions are strictly

local: each function $B_\ell(\cdot)$ has the knots $\xi_{\ell-v},\ldots,\xi_\ell$ with $\xi_r = \xi_{\min(\max(r,0),\kappa+1)}$ for $r = \ell-v,\ldots,\ell$ (Zhou et al., 1998), which means that its support consists of a small, fixed, finite number of intervals between knots. The unknown function $f(\cdot)$ is then approximated by $\widehat{f}(\cdot)$, a linear combination of basis functions $\{B_\ell\}_{\ell=1}^q$ with coefficients determined by a least squares criterion computed on the data $(y_i,x_i)_{i\in S_r}$ (Goga et al., 2019). The missing value y_i is then imputed by $\widehat{y}_i = \widehat{f}(x_i)$, where

$$\widehat{f}(x_i) = \sum_{\ell=1}^{q} \widehat{\beta}_\ell B_\ell(x_i) = \mathbf{b}_i^\top \widehat{\boldsymbol{\beta}}, \qquad x_i \in [0;1], \tag{17}$$

with $\mathbf{b}_i = (B_\ell(x_i))_{\ell=1}^q$ denoting the vector of B-spline basis functions, and $\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_\ell)_{\ell=1}^q$ minimizes

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}\in\mathbf{R}^q} \sum_{j\in S_r} w_j \left(y_j - \sum_{\ell=1}^{q} \beta_\ell B_\ell(x_j)\right)^2 = \left(\sum_{j\in S_r} w_j \mathbf{b}_j \mathbf{b}_j^\top\right)^{-1} \sum_{j\in S_r} w_j \mathbf{b}_j y_j; \tag{18}$$

see Goga et al. (2019). The expression of $\widehat{\boldsymbol{\beta}}$ is similar to that obtained with linear regression imputation given by (15) but unlike (15), the estimator (18) uses the B-spline functions $B_1,\ldots,B_q$, whose number can vary as a function of the number of knots κ and the order v of the B-spline functions. The degree v of the piecewise polynomial does not seem to have a great impact on the model fits if a large enough number of interior knots is used (Ruppert et al., 2003). This is why quadratic or cubic splines are mostly used in practice and an adequate number of interior knots will allow to obtain flexible fits that capture local non-linear trends in the data. Knots are usually placed at the X-quantiles and their number may have a great effect on the model fits: a large value of κ will lead to overfitting, in which case a penalization criterion may be used in (18), while a small value of κ may lead to underfitting. Ruppert et al. (2003) give a practical rule for choosing the number κ of interior knots :

$$\kappa = \min\left(\frac{1}{4} \times \text{number of unique } x_i, 35\right).$$

The imputed value (17) with B-spline regression can be also written as a weighted sum of the respondent y-values similar to (16), $\widehat{y}_i = \sum_{j\in S_r} w'_{ij} y_j$ for all $i \in S_m$ with weights

now given by $w'_{ij} = \mathbf{b}_i^\top \left(\sum_{j' \in S_r} w_{j'} \mathbf{b}_{j'} \mathbf{b}_{j'}^\top\right)^{-1} w_j \mathbf{b}_j$. These weights do not depend on the y-values as in linear regression imputation and $\sum_{j \in S_r} w'_{ij} = 1$ since $\sum_{j=1}^{q} B_j(x) = 1$ for all $x \in [0;1]$. Unlike linear regression imputation, the weights w'_{ij} are now local due to the B-spline functions ensuring more flexibility to model local nonlinear trends in the data.

We now turn to the multivariate case. For ease of presentation, we confine to the case of two predictors, X_1 and X_2. Additive models provide a simple way to model nonlinear trend in the data (Hastie and Tibshirani, 1986) and extend the standard linear model by allowing non-linear functions between the response variable Y and each of the explanatory variables, while maintaining additivity. In the case of two predictors, the relationship between Y and X_1, X_2 is expressed as a linear combination of unknown smooth functions f_1 and f_2:

$$y_i = \alpha + f_1(x_{i1}) + f_2(x_{i2}) + \epsilon_i, \tag{19}$$

where the ϵ_i's are independent errors with mean equal to zero. The model (19) is restricted to be additive and does not account for the potential interactions among the predictors. Accounting for interactions between X_1 and X_2 would require the additional predictor $X_1 X_2$ to be included in the model, leading to

$$y = f_1(x_1) + f_2(x_2) + f_3(x_1, x_2) + \xi,$$

where f_3 is a low-dimensional interaction function fitted by using two-dimensional smoothers, such as local regression or two-dimensional splines. This is beyond the scope of this article. When the number of predictors is large, the number of potential interactions may be considerable, making the implementation of this procedure challenging. In such situations, random forests and boosting, discussed in sections 3.6 and 3.7, provide more flexible approaches. But, as pointed out by James et al. (2015), additive models provide a useful compromise between linear and fully nonparametric models.

The unknown functions f_1 and f_2 in (19) can be estimated by using two B-spline basis $\mathcal{B}_1 = \{B_{11}, \ldots, B_{1q_1}\}$ and $\mathcal{B}_2 = \{B_{21}, \ldots, B_{2q_2}\}$, which leads to $\widehat{f}_1(x_{i1}) = \sum_{\ell=1}^{q_1} \widehat{\beta}_{1\ell} B_{1\ell}(x_{i1})$ and $\widehat{f}_2(x_{i2}) = \sum_{\ell=1}^{q_2} \widehat{\beta}_{2\ell} B_{2\ell}(x_{i2})$, where $\widehat{\beta}_{1\ell}$ and $\widehat{\beta}_{2\ell}$ are determined, as before, by a least square criterion. To ensure the identifiability of α, additional constraints such as $\sum_{i=1}^{n_r} \widehat{f}_1(x_{i1}) = \sum_{i=1}^{n_r} \widehat{f}_2(x_{i2}) = 0$ are usually imposed. With these constraints, the estimators $(\widehat{\alpha}, \widehat{\boldsymbol{\beta}}_1, \widehat{\boldsymbol{\beta}}_2)$ are

simply obtained as a regression coefficient estimator, for $\widehat{\boldsymbol{\beta}}_1 = (\widehat{\beta}_{1\ell})_{\ell=1}^{q_1}$ and $\widehat{\boldsymbol{\beta}}_2 = (\widehat{\beta}_{2\ell})_{\ell=1}^{q_2}$. The imputed value for missing y_i is given by

$$\widehat{y}_i = \widehat{\alpha} + \widehat{f}_1(x_{i1}) + \widehat{f}_2(x_{i2}), \quad i \in S_m. \tag{20}$$

In practice, a backfitting algorithm is used to compute $f_1(\cdot)$ and $f_2(\cdot)$ iteratively (Hastie et al., 2011). However, when the number p of explanatory variables is large, the algorithm may not converge and additive models tend to breakdown. Finally, random versions of (17) and (20) are obtained by adding random residuals as in (10).

3.5 Regression trees

Regression trees through the CART algorithm have been initially suggested by Breiman (1984). Tree-based methods are simple to use in practice for both continuous and categorical variables and useful for interpretation. They form a class of algorithms which recursively split the p-dimensional predictor space, the set of possible values for the X-variables, into distinct and non-overlapping regions of $\mathbb{R}^p$. The prediction $\widehat{f}_{tree}(\mathbf{x}_i)$ at point $\mathbf{x}_i$ corresponds to the average of the respondent y-values falling in the same region as unit i. When the number of X-variables is not too large, the splitting algorithm is quite fast, otherwise it may be time-consuming.

Following Creel and Krotki (2006), we slightly adapt the original CART algorithm as well as the estimation procedure of $f(\cdot)$. The CART algorithm recursively searches for the splitting variable and the splitting position (i.e., the coordinates on the predictor space where to split) leading to the greatest possible reduction in the residual mean of squares before and after splitting. More specifically, let A be a region or node and let $\#(A)$ the number of units belonging to A. A split in A consists of finding a pair (ℓ, z), where ℓ is the variable coordinates taking value between 1 and p, and z is the position of the split along the ℓth coordinate, within the limits of A. Let $\mathcal{C}_A$ be the set of all possible pairs (ℓ, z) in A. The splitting process is performed by searching for the best split (ℓ^*, z^*) in the sense that

$$(\ell^*, z^*) \quad = \quad \underset{(\ell,z)\in\mathcal{C}_A}{\arg\max}\, L(\ell, z) \tag{21}$$

with

$$L(\ell, z) \quad = \quad \frac{1}{\#(A)} \sum_{i \in S_r} \mathbb{1}(\mathbf{x}_i \in A) \left\{ (y_i - \bar{y}_A)^2 - (y_i - \bar{y}_{A_L} \mathbb{1}(X_{i\ell} < z) - \bar{y}_{A_R} \mathbb{1}(X_{i\ell} \geqslant z))^2 \right\}, \tag{22}$$

where X_{ij} is the measure of jth variable X_j for the ith individual, $A_L = \{\mathbf{X} \in A; \mathbf{X}_\ell < z\}$, $A_R = \{\mathbf{X} \in A; \mathbf{X}_\ell \geqslant z\}$ and $\mathbf{X}_\ell$ the ℓth coordinate of X; $\bar{y}_A$ is the average of y_i for those units i such that $\mathbf{x}_i \in A$. In (21), $\mathbb{1}(\mathbf{x}_i \in A) = 1$ if $\mathbf{x}_i \in A$, and $\mathbb{1}(\mathbf{x}_i \in A) = 0$, otherwise. From (21), the best split (ℓ^*, z^*) is the one that produces a tree with the smallest residuals sum of squares (James et al., 2015, Chap. 8); that is, we seek (ℓ^*, z^*) that minimizes

$$(\ell^*, z^*) = \arg \min_{(\ell,z) \in \mathcal{C}_A} \left\{ \sum_{i \in S_r : \mathbf{x}_i \in A} (y_i - \bar{y}_{A_L})^2 \mathbb{1}(X_{i\ell} < z) + \sum_{i \in S_r : \mathbf{x}_i \in A} (y_i - \bar{y}_{A_R})^2 \mathbb{1}(X_{i\ell} \geqslant z) \right\}.$$

The missing y_i is replaced by $\widehat{y}_i = \widehat{f}_{tree}(\mathbf{x}_i)$, which corresponds to the weighted average of the respondent y-values falling into the same region as $i \in S_m$:

$$\widehat{y}_i = \sum_{j \in S_r} \frac{w_j \mathbb{1}(\mathbf{x}_j \in A(\mathbf{x}_i))}{\sum_{j' \in S_r} w_{j'} \mathbb{1}(\mathbf{x}_{j'} \in A(\mathbf{x}_i))} y_j, \quad i \in S_m, \tag{23}$$

where $A(\mathbf{x}_i)$ is the region from $\mathbb{R}^p$ containing the point $\mathbf{x}_i$. With tree-based methods, the imputed value $\widehat{y}_i$ can also be expressed as

$$\widehat{y}_i = \sum_{j \in S_r} w'_{ij} y_j, \quad i \in S_m, \tag{24}$$

where $w'_{ij} = w_j \mathbb{1}(\mathbf{x}_j \in A(\mathbf{x}_i)) / \sum_{j' \in S_r} w_{j'} \mathbb{1}(\mathbf{x}_{j'} \in A(\mathbf{x}_i))$ with $\sum_{j \in S_r} w'_{ij} = 1$. With regression trees and tree-based methods in general, the non-overlapping A-regions obtained by means of the CART algorithm depend on the respondent data $\{(y_i, \mathbf{x}_i)\}_{i \in S_r}$; i.e., the same set of X-variables with a different set of respondents will lead to different non-overlapping A-regions. The resulting imputed estimator is similar to a post-stratified estimator based on adaptative post-strata.

Regression trees are simple to interpret and often exhibit a small model bias. However, they tend to overfit the data if each A-region contains too few elements. To cope with this issue, regression trees may be pruned, meaning that superfluous splits (with respect to a penalized version of (21)) are removed from the tree. Pruning a regression tree tends to

reduce its model variance at the expense of increasing the model bias; see Hastie et al. (2011). A random version of (24) is obtained by adding random residuals as in (10). Bagging and boosting methods may be used to improve the efficiency of tree-based procedures. This is discussed next.

3.6 Random forests

Random forest (Breiman, 2001) is an ensemble method which achieves better accuracy than tree-regression methods by creating a large number of different regression trees and combining them to produce more accurate predictions than a single model would. Random forests are especially efficient in complex settings such as small sample sizes, high-dimensional predictor space and complex relationships (Hamza and Larocque (2005), Díaz-Uriarte and de Andrés (2006), among others). Since the article of Breiman (2001), random forests have been extensively used in various fields such as medicine (Fraiwan et al., 2012), time series analysis (Kane et al., 2014), agriculture (Grimm et al., 2008), to cite just a few. Recently, their theoretical properties have been established by Scornet et al. (2015).

There exist a number of random forest algorithms (see Biau and Scornet (2016) for discussion). A widely used algorithm proceeds as follows (Dagdoug et al., 2020b):

Step 1: Consider B bootstrap data sets $D_1, D_2, ..., D_B$, obtained by selecting with replacement n_r pairs $(y_i, \mathbf{x}_i)$ from $D = \{(y_i, \mathbf{x}_i)\}_{i \in S_r}$.

Step 2: In each bootstrap data set D_b for $b = 1, \ldots, B$, fit a regression tree and determine the prediction $\widehat{f}_{tree}^{(b)}$ for the unknown f in (8) as described in section 3.5. For each regression tree, only p' variables randomly chosen among the p variables are considered in the search for the best split in (21).

Step 3: The imputed value for missing y_i is obtained by averaging the predictions at the point $\mathbf{x}_i$ of the B regression tree predictions:

$$\widehat{y}_i = \frac{1}{B} \sum_{b=1}^{B} \widehat{f}_{tree}^{(b)}(\mathbf{x}_i), \quad i \in S_m, \tag{25}$$

where $\widehat{f}_{tree}^{(b)}(\mathbf{x}_i)$ is the prediction for the unknown f in (8) computed at $\mathbf{x}_i$ and obtained with the bth regression tree as described in Section 3.5. More specifically, from (23),

the prediction $\widehat{f}_{tree}^{(b)}(\mathbf{x}_i)$ corresponds to the weighted average of y-values for $j \in S_r$ falling in the same region $A^{(b)}(\mathbf{x}_i)$ containing $i \in S_m$.

A random version of (25) is obtained by adding random residuals as in (10). Although random forests are based on fully-grown trees, the accuracy of the predictions is improved by considering bootstrap of units and model aggregation, a procedure called *bagging* and used in statistical learning for reducing the variability. The number B of regression trees should be large enough to ensure a good performance without harming the processing time; see Scornet (2017). The second improvement brought by random forest is the random selection at each split of p' predictors, achieving decorrelated trees. The value of p' is typically chosen as $p' \simeq \sqrt{p}$ (Hastie et al., 2011). In random forest algorithms, a stopping criterion is usually specified so that the algorithm stops once a certain condition (e.g., on the minimum number of units in each final nodes) is met.

3.7 Least square tree-boosting and other tree-boosting methods

As in bagging, boosting (Friedman, 2001) is a procedure that can be applied to any statistical learning methods for improving the accuracy of model predictions and is typically used with tree-based methods. While bagging involves the selection of bootstrap samples to create many different predictions, boosting is an iterative method that starts with a weak fit (or learner) and improves it at each step of the algorithm by predicting the residuals of prior models and adding them together to make the final prediction.

To understand how boosting works, consider a regression tree with non-overlapping regions $A_1, \ldots, A_J$, expressed as

$$T(x, \Theta) = \sum_{j=1}^{J} \gamma_j \mathbb{1}(\mathbf{x}_i \in A_j). \tag{26}$$

The parameter $\Theta = \{\gamma_j, A_j\}_{j=1}^{J}$ is obtained by minimizing

$$\widehat{\Theta} = \arg\min_{\Theta} \sum_{j=1}^{J} \sum_{i:\mathbf{x}_i \in A_j} \mathcal{L}(y_i, \gamma_j) = \arg\min_{\Theta} \sum_{i \in S_r} \mathcal{L}(y_i, T(\mathbf{x}_i, \Theta)), \tag{27}$$

where $\mathcal{L}$ denotes a loss function; e.g., the quadratic loss function. With the latter, given a region A_j, estimating the constant γ_j is usually straightforward as $\widehat{\gamma}_j = \overline{y}_j$ the average the y-

values belonging to A_j. However, finding the regions $\{A_j\}_{j=1}^J$ and solving (27) in a traditional way may prove challenging and computationally intensive as it requires optimizing over all the parameters jointly. To overcome this difficulty, one may use a greedy top-down recursive partitioning algorithm to find $\{A_j\}_{j=1}^J$ as described in Section 3.5. Alternatively, one may split the optimization problem (27) into many simple subproblems that can be solved rapidly. Boosting uses the latter and considers that the unknown f has the following additive form:

$$f(x) = \sum_{m=1}^{M} T(x, \Theta_m), \tag{28}$$

where $T(x, \Theta_m)$ for $m = 1, \ldots, M$ are trees determined iteratively by using a forward stage-wise procedure (Hastie et al., 2011): at each step, a new tree is added to the expansion without modifying the coefficients and parameters of trees already added. Each added tree, usually referred to as a weak-learner, has a small size and slowly improves the estimation of f in areas where it does not perform well. For the quadratic loss function, after accounting for the survey weights, the algorithm becomes:

Step 1: Initialize the algorithm with a constant value: $\widehat{f}_0(\mathbf{x}_i) = 0$ and

$$\widehat{\gamma}_0 = \underset{\gamma \in \mathbb{R}}{\arg\min} \sum_{i \in S_r} w_i (y_i - \gamma)^2 = \frac{1}{\sum_{i \in S_r} w_i} \sum_{i \in S_r} w_i y_i.$$

Step 2: For $m = 1$ to M:

(a) Given the current model $\widehat{f}_{m-1}$, fit the regression tree that best predicts the residuals values $y_i - \widehat{f}_{m-1}(\mathbf{x}_i), i \in S_r$ and get the terminal regions $(A_{jm})_{j=1}^{J_m}$.

(b) Given the terminal regions A_{jm}, the optimal constants $\widehat{\gamma}_{jm}$ are found as follows:

$$\widehat{\gamma}_{jm} = \underset{\gamma_{jm}}{\arg\min} \sum_{i \in S_r : \mathbf{x}_i \in A_{jm}} w_i \mathcal{L}(y_i, \widehat{f}_{m-1}(\mathbf{x}_i) + \gamma_{jm}) = \underset{\gamma_{jm}}{\arg\min} \sum_{i \in S_r : \mathbf{x}_i \in A_{jm}} w_i (y_i - \widehat{f}_{m-1}(\mathbf{x}_i) - \gamma_{jm})^2$$

for $j = 1, \ldots, J_m$.

(c) Update $\widehat{f}_m(\mathbf{x}_i) = \widehat{f}_{m-1}(\mathbf{x}_i) + T(\mathbf{x}_i, \widehat{\Theta}_m)$ where $\widehat{\Theta}_m = \{A_{jm}, \widehat{\gamma}_{jm}\}_{j=1}^{J_m}$ and $T(\mathbf{x}_i, \widehat{\Theta}_m) = \sum_{j=1}^{J_m} \widehat{\gamma}_{jm} \mathbb{1}(\mathbf{x}_i \in A_{jm})$.

Step 3: Output $\widehat{f}_M(\mathbf{x}_i)$ and get the imputed value

$$\widehat{y}_i = \widehat{f}_M(\mathbf{x}_i). \tag{29}$$

A random version of (29) is obtained by adding random residuals as in (10). The number M of trees should not be too large and, for better performances, Hastie et al. (2011) recommend to consider the same number of splits $J_m = J$ at each iteration. The value of J reflects the level of dominant interactions between the X-variables. The value $J = 2$ (one split) produces boosted models with only main effects without interactions, whereas the value $J = 3$ allows for two-variable interactions. Empirical studies suggest that $J = 6$ generally leads to good results. As in ridge regression, shrinkage is used with tree boosting. In this case, Step 2. (c) of the above algorithm is replaced by a penalized version:

$$\widehat{f}_m(\mathbf{x}_i) = \widehat{f}_{m-1}(\mathbf{x}_i) + \nu T(\mathbf{x}_i, \widehat{\Theta}_m),$$

where the parameter $\nu \in (0, 1)$, called learning rate, is used to penalized large trees; usually $\nu = 0.1$ or 0.01. Both M and ν control the performance of the model prediction.

3.7.1 XGBoost

Chen and Guestrin (2016) suggested a scalable end-to-end tree boosting system called XGBoost which is extremely fast. Here, we adapt the algorithm in order to account for the survey weights. Consider again a tree with formal expression given in (26). This tree learning algorithm consists of minimizing the following objective function at the m-th iteration:

$$\widehat{\Theta}_m = \arg\min_{\Theta_m} \{ \sum_{i \in S_r} w_i \mathcal{L}(y_i, \widehat{f}_{m-1}(\mathbf{x}_i) + T(\mathbf{x}_i, \Theta_m)) \} + \Omega(T(x, \Theta_m)), \tag{30}$$

where the penalty function $\Omega(T(x, \Theta_m)) = \gamma J + \frac{\lambda}{2} \sum_{j=1}^{J} \gamma_j^2$ penalizes large trees in order to avoid overfitting. The search problem is optimized by using a second-order Taylor approximation of $\mathcal{L}$, and ignoring the constant term, the new optimization problem reduces to:

$$\widehat{\Theta}_m = \arg\min_{\Theta_m} \sum_{j=1}^{J} \left[\gamma_j \sum_{i \in S_r : \mathbf{x}_i \in A_j} w_i g_i + \frac{1}{2} \gamma_j^2 (\sum_{i \in S_r : \mathbf{x}_i \in A_j} w_i h_i + \lambda) \right] + \gamma J, \tag{31}$$

where g_i and h_i are the first and second-order derivatives of the loss function computed at $\widehat{f}_{m-1}(\mathbf{x}_i)$. With the quadratic loss function, $g_i = 2(\widehat{f}_{m-1}(\mathbf{x}_i) - y_i)$ and $h_i = 2$. The new objective function from (31) is a second-order polynomial with respect to γ_j, so the optimal γ_j is easily obtained as $\gamma_j^* = -(\sum_{i \in S_r : \mathbf{x}_i \in A_j} w_i g_i)/(\sum_{i \in S_r : \mathbf{x}_i \in A_j} w_i h_i + \lambda)$, leading to the optimal value of the objective function as $-(1/2)\sum_{j=1}^{J}(\sum_{i \in S_r : \mathbf{x}_i \in A_j} w_i g_i)^2/(\sum_{i \in S_r : \mathbf{x}_i \in A_j} w_i h_i + \lambda) + \gamma J$. This value is then used next as a decision criterion in a greedy top-down recursive algorithm to find the optimal regions A_j of the m-th tree to be added.

3.7.2 Bayesian additive regression trees (BART)

Bayesian additive regression trees (Chipman et al., 2010, BART) is similar to boosting in the sense that the unknown regression function f has an additive form as in (28). While boosting is completely nonparametric, BART makes a Gaussian assumption on the model errors:

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right),$$

where $f(x) = \sum_{m=1}^{M} T(x, \Theta_m) = \sum_{m=1}^{M} T_m(x, \Gamma_m)$ is assumed to be a sum of tree functions and $\Gamma_m = \{\gamma_j, \gamma_2, \ldots, \gamma_{Jm}\}$ is the set of parameter values associated with the J_m terminal nodes in each tree $T(x, \Theta_m)$.

As stated in Chipman et al. (2010), although similar in spirit to gradient boosting, BART differs from boosting algorithms both by the way it weakens the individual trees by relying on a Bayesian framework, but also on how it performs the iterative fitting. More specifically, a prior is specified for the parameters of the model $(T_1, \Gamma_1), (T_2, \Gamma_2), \ldots, (T_m, \Gamma_m)$ and σ^2. The prior of T_m can be decomposed into three components :

1. The probability that a node at depth J is a terminal node is given by $\alpha\,(1+J)^{-\beta}$ for $\alpha \in (0;1)\,,\ \beta \geq 0$.

2. The distribution on the splitting variable assignments in each interior node is uniform.

3. The distribution of the splitting value conditional on the chosen splitting variable is also uniform.

Borrowing the illustrative example of Chipman et al. (2010), with the parameters $\alpha = 0.95$ and $\beta = 2$, trees with $1, 2, 3, 4, 5$ terminal nodes receive prior probabilities of

0.05, 0.55, 0.28, 0.09 and 0.03, respectively. Therefore, as in boosting, the BART model tends to favor trees with a small number of terminal nodes. However, the process of restricting the depth of regression trees (or equivalently the number of terminal nodes) in BART is different from the one used in boosting. For boosting, the depth of the trees is fixed by the user and is similar for all trees used in the forest. For BART, the user specifies a probability for the trees to have a certain number of terminal nodes. As a result, the number of terminal nodes is random rather tan fixed. Therefore, it is likely that trees have only a small number of terminal nodes with the BART model, but this number can vary depending on the data at hand. For γ_j, a conjugate prior is chosen to make computations simpler; e.g., $p(\gamma_{jm}|T_m)$ is assumed to be $\mathcal{N}(\gamma_\gamma, \sigma^2_\gamma)$. Similarly, a conjugate prior is chosen for σ^2, e.g., the inverse chi-square distribution. To generate the posterior distribution, the authors suggest the use of a Gibbs sampler. For general guidelines about the choices of these parameters, see Chipman et al. (2010). The imputed value for missing y_i is obtained as with the general boosting algorithm given in Section 3.7, where the prediction of each regression tree is the weighted average of the values in the terminal node containing $\mathbf{x}_i$.

3.8 Cubist algorithm

Cubist is an updated implementation of the M5 algorithm introduced by Quinlan et al. (1992) and Quinlan (1993). It is an algorithm based on regression trees and linear models, among other ingredients. Initially, Cubist was only available under a commercial license. In 2011, the code was released as open-source. The algorithm proceeds as follows (Kuhn and Johnson, 2013, Chap. 8):

Step 1: Create a partition $\mathcal{P} = \{A_1, A_2, ..., A_T\}$ of $\mathbb{R}^p$. To do so, let $\mathcal{C}_A$ be the set of all possible splits in a node A of cardinality ℓ, that is, the set of all possible pairs (position, variable). Then, the split is performed using the following criterion:

$$L'(z,j) = \underset{(z,j)\in\mathcal{C}_A}{\arg\max} \sqrt{\sum_{i\in S_r}\left(y_i - \left(\frac{1}{n_r}\sum_{j'\in S_r} y_{j'}\right)\right)^2} - \sum_{h=1}^{\ell}\frac{n_h}{n_r}\sqrt{\sum_{i:\mathbf{x}_i\in D_h}\left(y_i - \left(\frac{1}{n_r}\sum_{j':\mathbf{x}_i\in D_h} y_{j'}\right)\right)^2},$$

where $D_1, \ldots, D_\ell$ denote the ℓ non-terminal nodes after each of the $\ell - 1$ previous splits and n_h denotes the cardinal of elements in the node D_h.

Step 2: In each node, a linear model is fitted between the survey variable Y and the auxiliary variables that have been used to split the tree. More specifically, consider the jth terminal node A_j. Then, there exists a path from the first node to the current node A_j in the graph formed by the tree. This path uses p'_j variables among the set $\{X_1, X_2, ..., X_p\}$. For instance, assume that a partition of 5 elements is created by the tree shown in Figure 1. Then, the linear model in the node A_1 is fitted using the variables that created the path in red, that is, X_1, X_4 and X_6, and so $p'_1 = 3$ for this node. The linear model fitted in the node A_4 uses only one variable, X_1, (the green path), so $p'_4 = 1$. The coefficients $\boldsymbol{\beta}_j \in \mathbb{R}^{p'_j}$ of the linear model in the node A_j are

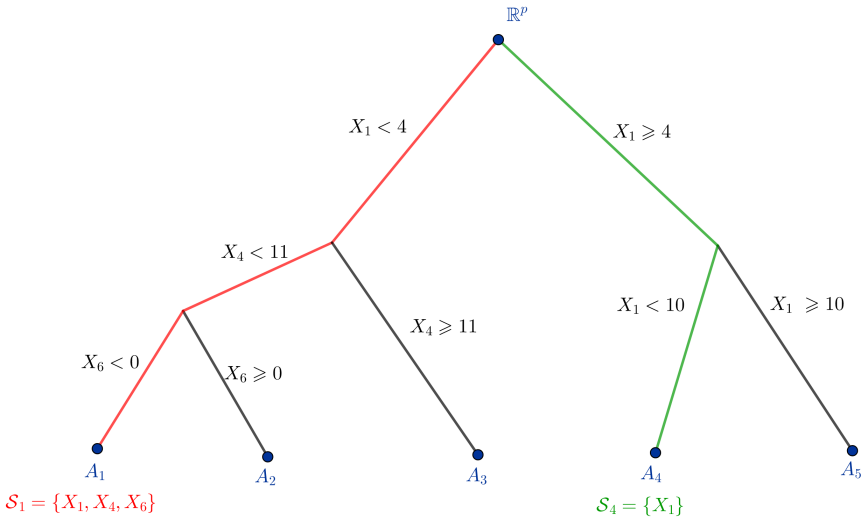

Figure 1: Example of a graph induced by a tree algorithm.

estimated using the customary weighted least squares criterion:

$$\widehat{\boldsymbol{\beta}}_j = \underset{\boldsymbol{\beta}_j \in \mathbb{R}^{p'_j}}{\arg\min} \sum_{i \in S_r} w_i \left\{ y_i - \boldsymbol{\beta}_j^\top \mathbf{x}_i^{(j)} \right\}^2 \mathbb{1}\left(\mathbf{x}_i \in A_j\right),$$

where $\mathbf{x}_i^{(j)}$ is the vector containing the measurements of the p'_j variables for unit i.

Step 3: In each node, a backward elimination procedure is performed using the adjusted error rate (AER) criterion. For instance, in the jth terminal node, we have

$$AER(A_j) = \frac{\#(A_j) + p^*}{\#(A_j) - p^*} \sum_{i \in S_r : \mathbf{x}_i \in A_j} |y_i - \widehat{y}_i|,$$

where p^* denotes the number of variables used in the current model which predicts $\widehat{y}_i$ for a prediction at the point $\mathbf{x}_i$. Each variable in the initial model is dropped and the AER is recomputed. Terms are dropped from the model as long as the AER decreases.

Step 4: Once the tree is fully grown, it is pruned by removing unnecessary splits. Starting at the terminal nodes, the AER is computed with and without the node. Whenever the node does not result in a decrease of the AER, it is pruned. This process is performed until no more node can be removed.

Step 5: To avoid over-fitting, a smoothing procedure is performed. Let $\widehat{y}_{i(j)}$ be the predicted value obtained by fitting the linear model in the jth child node and $\widehat{y}_{i(p)}$ be the predicted value obtained from the direct parent node. These predictions are combined as

$$\widehat{y}_i = a y_{i(j)} + (1-a)\widehat{y}_{i(p)},$$

where

$$a = \frac{\widehat{V}(\mathbf{e}_{(p)}) - \widehat{Cov}(\mathbf{e}_{(j)}, \mathbf{e}_{(p)})}{\widehat{V}(\mathbf{e}_{(j)} - \mathbf{e}_{(p)})}$$

with $e_{i(j)} = y_i - \widehat{y}_{i(j)}$ denoting the ith coordinate of the vector $\mathbf{e}_{(j)}$, $e_{i(p)} = y_i - \widehat{y}_{i(p)}$ denoting the ith coordinate of the vector $\mathbf{e}_{(p)}$ and $\widehat{V}(\cdot)$ and $\widehat{Cov}(\cdot,\cdot)$ denoting the empirical model variance and covariance, respectively.

Step 6: Cubist can be used as an ensemble model. Once the Cubist algorithm is fitted, the subsequent iterations of the algorithm use the previously trained algorithm to define an adjusted response $y_i^{(m)}$ so that the next iteration of the algorithm uses

$$y_i^{(m)} = y_i - (y_i^{(m-1)} - y_i),$$

where $y_i^{(m)}$ is the value of the adjusted response y_i for the mth iteration of the Cubist algorithm.

Step 7: The final imputed value for missing y_i is derived using a K nearest-neighbour rule:

$$\widehat{y}_i = \frac{1}{K}\sum_{k=1}^{K} \frac{1}{0.5 + d_k}(t_k + \widehat{y}^{(k)} - \widehat{t}_k), \tag{32}$$

where d_k denotes the distance between $\mathbf{x}_i$ and the kth neighbor, t_k denotes the outcome of the kth neighbor and $\widehat{t}_k$ its predicted value.

A random version version of (32) is obtained by adding random residuals as in (10).

3.9 Support vector regression

Support vector machines (Vapnik, 1998, 2000; Cortes and Vapnik, 1995; Smola and Schölkopf, 2004) belong to the class of supervised learning algorithms and may be used for regression analysis. We start by considering the linear regression model

$$f(\mathbf{x}_i) = \beta_0 + \mathbf{x}_i^T \boldsymbol{\beta}, \quad \beta_0 \in \mathbb{R}, \quad \boldsymbol{\beta} \in \mathbb{R}^p,$$

before discussing the case of nonlinear relationships. In the customary regression framework, the goal is to minimize the residuals sum of squares. In Support Vector Regression (SVR), the goal is to minimize a function of the residuals plus a L^2-penalization on the regression coefficient:

$$\mathcal{S} = \sum_{i \in S_r} V_\epsilon(y_i - f(\mathbf{x}_i)) + \frac{\lambda}{2}||\boldsymbol{\beta}||^2, \tag{33}$$

where V_ϵ is the so-called ϵ-insensitive error measure defined as $V_\epsilon(x) = 0$ if $|x| < \epsilon$ and $|x| - \epsilon$ otherwise (Vapnik, 2000) for $\epsilon > 0$; ε can be viewed as the allowed tolerance for fitting; see Figure 1 in Smola and Schölkopf (2004). The optimization problem (33) may not have solution and supplementary tolerances ξ_i, ξ_i^* (called also "the slack variables") on the individual fitted errors are considered (Smola and Schölkopf, 2004). There exist several ways for incorporating weights in the optimization problem, leading to different weighted support vector regression solutions. We consider the method suggested by Lee et al. (2005) and Han and Clemmensen (2014):

$$\underset{\beta}{\text{minimize}} \quad \frac{1}{2}||\boldsymbol{\beta}||^2 + C \sum_{i \in S_r} \widetilde{w}_i \, (\xi_i + \xi_i^*) \tag{34}$$

and

$$\begin{aligned} \text{subject to} \quad & y_i - \beta_0 - \mathbf{x}_i^T \boldsymbol{\beta} \leqslant \epsilon + \xi_i, \\ & \beta_0 + \mathbf{x}_i^T \boldsymbol{\beta} - y_i \leqslant \epsilon + \xi_i^*. \\ & \xi_i, \xi_i^* > 0, \end{aligned} \tag{35}$$

where $C > 0$ is the tuning parameter that provides a trade-off between the smoothness of the fitted function and the deviation from the training data and $\widetilde{w}_i = w_i / \sum_{j \in S_r} w_j \in (0, 1)$ denotes the normalized sampling weight associated with unit i. As a result, the $\widetilde{w}_i$'s are all smaller than one. As argued by Han and Clemmensen (2014), incorporating weights in the objective function as in (34) has the effect of shrinking the estimators $\widehat{\beta}_j$ to different extents. The solution of (33) and (35) is given by $\widehat{\boldsymbol{\beta}} = \sum_{i \in S_r} (\widehat{\alpha}_i - \widehat{\alpha}_i^*) \mathbf{x}_i$, which leads to

$$\widehat{f}(\mathbf{x}) = \sum_{i \in S_r} (\widehat{\alpha}_i - \widehat{\alpha}_i^*) < \mathbf{x}_i, \mathbf{x} > + \beta_0, \tag{36}$$

where $< \cdot, \cdot >$ is an inner product and $\widehat{\alpha}_i > 0$ and $\widehat{\alpha}_i^* > 0$ denote the Lagrange multipliers verifying the quadratic programming problem:

$$\min_{\alpha_i, \alpha_i^*} \epsilon \sum_{i \in S_r} (\alpha_i + \alpha_i^*) - \sum_{i \in S_r} y_i (\alpha_i - \alpha_i^*) + \frac{1}{2} \sum_{i,j \in S_r} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) < \mathbf{x}_i, \mathbf{x}_j >$$

subject to $0 \leq \alpha_i, \alpha_i^* \leq C_i := C \times \widetilde{w}_i$, $\sum_{i \in S_r} (\alpha_i - \alpha_i^*) = 0$ and $\alpha_i \alpha_i^* = 0$. As a result, only a subset of the solution values $(\widehat{\alpha}_i - \widehat{\alpha}_i^*)$ are nonzero and the associated data values are called the support vectors. The solution $\widehat{\boldsymbol{\beta}}$ is written as a linear combination of these support vectors. Moreover, the prediction $\widehat{f}(\mathbf{x})$ uses only the support vectors and the inner products between $\mathbf{x}$ and $\mathbf{x}_i$ without requiring the computation of $\widehat{\boldsymbol{\beta}}$. This property is useful for extending the method to handle nonlinear relationships.

We now consider the case of a nonlinear and unknown function f. We approximate f in a basis of functions $\{\phi_m\}_{m=1}^M$ as follows:

$$f(x) = \sum_{m=1}^{M} \beta_m \phi_m(x) + \beta_0$$

and β_0 and $\boldsymbol{\beta} = (\beta_m)_{m=1}^M$ minimize (34) and

$$\begin{aligned} \text{subject to} \quad & y_i - \beta_0 - \sum_{m=1}^{M} \beta_m \phi_m(x_i) \leqslant \epsilon + \xi_i, \\ & \beta_0 + \sum_{m=1}^{M} \beta_m \phi_m(x_i) - y_i \leqslant \epsilon + \xi_i^*. \\ & \xi_i, \xi_i^* > 0. \end{aligned} \tag{37}$$

A similar derivation as before leads to $\widehat{\boldsymbol{\beta}} = \sum_{i \in S_r} (\widehat{\alpha}_i - \widehat{\alpha}_i^*)\, \phi(\mathbf{x}_i)$ for $\phi(\mathbf{x}_i) = (\phi_m(\mathbf{x}_i))_{m=1}^M$ and

$$\widehat{f}(\mathbf{x}) = \sum_{i \in S_r} (\widehat{\alpha}_i - \widehat{\alpha}_i^*)\, \mathcal{K}(\mathbf{x}_i, \mathbf{x}) + \beta_0,$$

where $\mathcal{K}(\mathbf{x}_i, \mathbf{x}) =< \phi(\mathbf{x}_i), \phi(\mathbf{x}) >= \sum_{m=1}^M \phi_m(\mathbf{x}_i)\phi_m(\mathbf{x})$ is a positive definite kernel (Smola and Schölkopf, 2004). The computation of $\widehat{f}(\mathbf{x})$ involves $\phi(\mathbf{x})$ only through inner products and using a kernel function makes the computation of $\widehat{f}(\mathbf{x})$ possible without requiring $\phi(\mathbf{x})$. All is needed is the knowledge of $\mathcal{K}$. Using $\mathcal{K}$, it is possible to solve the optimization problem in a higher-dimensional space without having to compute any product in this space. Common choices of $\mathcal{K}(\cdot, \cdot)$ include the Gaussian kernel $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-||\mathbf{x}_i - \mathbf{x}_j||^2\right)$ and the polynomial kernel $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \left(1 + \mathbf{x}_i^\top \mathbf{x}_j\right)^q, \quad q = 2, 3, \ldots$. The imputed value for the missing y_i is given by

$$\widehat{y}_i = \sum_{j \in S_r} \left(\widehat{\alpha}_j - \widehat{\alpha}_j^*\right) \mathcal{K}(\mathbf{x}_j, \mathbf{x}_i) + \widehat{\beta}_0. \tag{38}$$

A random version version of (38) is obtained by adding random residuals as in (10). The reader is referred to Smola and Schölkopf (2004) for a discussion on how to estimate β_0.

4 Simulation study: the case of population totals

We conducted an extensive simulation study to investigate the performance of the imputation procedures described in Section 3 in terms of bias and efficiency.

4.1 The setup

For each scenario, we repeated $R = 5,000$ iterations of the following process:

(i) A finite population of size $N = 10,000$ was generated. The population consisted of a survey variable Y and a set of predictors $X_1, \ldots, X_p$.

(ii) From the finite population generated in Step (i), a sample, of size n, was selected according to a given probability sampling design.

(iii) In each sample, nonresponse to item Y was generated according to a given nonresponse mechanism.

(iv) The missing values in each sample were imputed using several imputation procedures.

We now give a more in-depth discussion of each of the steps (i)-(iv).

We first generated five predictors $X_1, \ldots, X_5$, according to the following distributions: X_1 followed a normal distribution, $X_1 \sim \mathcal{N}(0,1)$; X_2 followed a Beta distribution, $X_2 \sim \text{Beta}(3,1)$; X_3 followed a Gamma distribution, $X_3 \sim 2 \times \text{Gamma}(3,2)$; X_4 followed a Bernoulli distribution, $X_4 \sim \mathcal{B}(0.7)$; and X_5 followed a multinomial distribution, $X_5 \sim \text{Mult}(0.4, 0.3, 0.3)$. The predictors X_1-X_3 were continuous, whereas the predictors X_4 and X_5 were discrete. The predictors X_1-X_3 were standardized so as to have a zero mean and a variance equal to one. Given the predictors X_1-X_5, we generated the continuous survey variables $Y_1, \ldots, Y_8$, according to the following models:

- $Y_1 = 2 + 2X_1 + X_2 + 2X_3 + \mathcal{N}(0,1)$;
- $Y_2 = 2 + 2X_1 + X_2 + 2X_3 + \text{Pareto}(1,4)$;
- $Y_3 = 2 + X_1 + X_2^2 + X_3 + \mathcal{N}(0,1)$;
- $Y_4 = 2 + 2X_1 + X_2 + 3X_3X_4 + 1.5\mathbb{1}(X_5 = 1) - 2\mathbb{1}(X_5 = 2) + \mathcal{N}(0,1)$;
- $Y_5 = 2 + 5X_1^3 + 4X_2^2 + X_3X_4 + 1.5\mathbb{1}(X_5 = 1) - 2\mathbb{1}(X_5 = 2) + \mathcal{N}(0,1)$;
- $Y_6 = 2 + (2X_1 + X_2 + 2X_3)^2 + \mathcal{N}(0,1) + \text{Beta}(3,1)$;
- $Y_7 = 2 + (2X_1 + X_2 + 3X_3X_4 + 1.5\mathbb{1}(X_5 = 1) - 2\mathbb{1}(X_5 = 2))^2 + \mathcal{N}(0,1)$;
- $Y_8 = 4\cos(X_1) + \mathcal{N}(0,1)$;

and the binary survey variables as follows:

- $Y_9 = \mathbb{1}(S_1 > 1/2)$, where

$$S_1 = 0.1 + 0.79 \exp\{1 + 0.5\,(0.75 + 2X1 + 2X_2 + 2X_3 - X_4 - X_3X_4 + 1.5\mathbb{1}(X_5 = 1) - 2\mathbb{1}(X_5 = 2))\}^{-1};$$

- $Y_{10} = \mathbb{1}(S_2 > 1/2)$, where

$$S_2 = 0.55 \times Q + 0.02 - 0.01X_2^3$$

with

$$Q = \exp\{1 + 0.4 \times (6.5 + 2X_1 + 2X_2 + 2X_3 - X_4 - X_3X_4 + 1.5\mathbb{1}(X_5 = 1) - 2\mathbb{1}(X_5 = 2))\}^{-1}. \quad (39)$$

For the survey variables Y_2 and Y_6, note that we have generated errors for non-normal distribution to assess the robustness of the BART procedure that assumes a Gaussian distribution for the errors.

From each population, we selected samples, of (expected) size $n = 1,000$, according to two sampling designs: (a) simple random sampling without replacement and (b) Poisson sampling with probability proportional to the values of the variable X_5; i.e., $\pi_i = 1,000 \times (x_{5i}/\sum_{i\in U} x_{5i})$ for all $i \in U$. Simple random sampling without replacement was used for estimating the finite population total of the continuous survey variables Y_1-Y_6 and Y_8 and the binary variables Y_9 and Y_{10}, whereas Poisson sampling was used for estimating the totals of the survey variables Y_4 and Y_7.

In each sample, nonresponse to the survey variable Y_ℓ, $\ell = 1, \ldots, 10$, was generated according to four nonresponse mechanisms. That is, the response indicators r_i were generated from a Bernoulli distribution with probability p_{gi}, $g = 1, \ldots, 4$, where

$$\text{(NR1)}: \; p_{1i} = 0.1 + 0.79 \exp\{1 + 0.5\,(0.75 + 2x_{i1} + 2x_{i2} + 2x_{i3} - x_{i4} - x_{i3}x_{i4} + 1.5\mathbb{1}(x_{i5} = 1) - 2\mathbb{1}(x_{i5} = 2))\}^{-1};$$

$$\text{(NR2)}: \; p_{2i} = 0.5;$$

$$\text{(NR3)}: \; p_{3i} = 0.55 \times q_i + 0.02 - 0.01x_{i2}^3;$$

$$\text{(NR4)}: \; p_{4i} = 0.5 \times q_i + 0.13 - 0.1\,(\sin(x_{i1}) + \cos(x_{i2}));$$

where q_i is the ith value of Q given by (39). In (NR1)-(NR4), the model parameters were set so as to obtain a response rate of about 50% in each sample.

In each sample, the missing values were imputed according to eleven imputation procedures described in section 3. Some of the imputation procedures required the specification of some parameters (e.g., regularization parameter, depth of a regression tree, choice of a kernel, etc.). We have included several configurations to assess the impact of these parameters on the performance of these procedures. Based on the different configurations, we ended up with twenty-seven imputation procedures. More specifically, we included the following procedures:

Procedure 1: "LR" : Deterministic linear regression imputation; see Section 3.1.

Procedure 2: "MWCα" : Mean imputation within classes, where the number of units in each class was set to $\alpha \in \{50, 100, 250, 500\}$; see Section 3.2.

Procedure 3: "HDWCα" : Random hot-deck imputation within classes, where the number of units in each class was set to $\alpha \in \{50, 100, 250\}$; see Section 3.2.

Procedure 4: "KNN" : K-Nearest-Neighbours imputation with $K = 1$ and $K = 5$ nearest neighbours and the euclidian distance and implemented with the R-package `caret`; see Section 3.3.

Procedure 5: "AMSα" : Additive models based on cubic B-splines with α equidistant interiors knots placed at the x-quantiles, where $\alpha \in \{5, 10\}$ and implemented with the R-package `mgcv`; see Section 3.4.

Procedure 6: "CART" : Imputation through regression trees with the CART algorithm and implemented with the R-package `rpart`; see Section 3.5.

Procedure 7: "RF1" : Imputation through random forest with $B = 1000$ trees, one observation per terminal node and 1 predictor considered for the search in each split. "RF2": Random forest with $B = 1000$ trees, 5 observations per terminal node and $\sqrt{p}$ predictors considered for each split, where p is the number of X-variables used in the imputation model, in our case $p = 5$. "RF3" : Random forest with $B = 1000$ trees, 10 observations per terminal node and $\sqrt{p}$ predictors considered for each split. Simulations were implemented with the R-package `ranger`; see Section 3.6.

Procedure 8: "XGB1": XGBoost algorithm with $M = 50$ trees each one with $J = 3$ final splits and a learning rate of 0.1. "XGB2": XGBoost algorithm with $M = 100$ trees with $J = 6$ and a learning rate of 0.05. "XGB3": XGBoost algorithm with $M = 250$ trees with $J = 10$ and a learning rate of 0.01. Simulations were implemented with the R-package `xgboost`; see Section 3.7.1.

Procedure 9: "BART" : Imputation through Bayesian additive regression trees. Simulations were implemented with the R-package `bartMachine`; see Section 3.7.2.

Procedure 10: "CUBIST1": Cubist with one model. "CUBIST2" : Cubist with five models. "CUBIST3" : Cubist with 5 models and unbiased estimation. Simulations were implemented with the R-package `Cubist`; see Section 3.8.

Procedure 11: "SVR1": Support vector regression imputation with a Gaussian kernel and the ν objective function. "SVR2": Support vector regression imputation with a polynomial kernel of degree 3 and the ϵ-insensitive objective function. "SVR3": Support vector regression imputation with a Gaussian kernel and the ϵ-insensitive objective function. "SVR4": Support vector regression imputation with a linear kernel and the ϵ-insensitive objective function. Simulations were implemented with the R-package `e1071`; see Section 3.9.

The imputation procedures used in our simulations were based on an imputation model that included the predictors $X_1, \ldots, X_5$, without any interaction terms. Except for random hot-deck imputation (Procedure 3) and nearest-neighbour imputation (Procedure 4 with $K = 1$), for the binary variables Y_9 and Y_{10}, note that we have generated zeroes and ones from independent Bernoulli distributions with parameter $\widehat{y}_i$, where $\widehat{y}_i$ denotes the predicted value associated with unit i. Whenever $\widehat{y}_i < 0$, we set it to $\widehat{y}_i = 0$. Similarly, when $\widehat{y}_i > 1$, we set it to $\widehat{y}_i = 1$.

As a measure of bias of the imputed estimator $\widehat{t}_{imp}$ given by (4), we computed the Monte Carlo percent relative bias defined as

$$RB_{MC}(\widehat{t}_{imp}) = 100 \times \frac{1}{R} \sum_{r=1}^{R} \frac{(\widehat{t}_{imp}^{(r)} - t_y)}{t_y}, \tag{40}$$

where $\widehat{t}_{imp}^{(r)}$ denotes the imputed estimator $\widehat{t}_{imp}$ at the rth iteration, $r = 1, \ldots, 5,000$.

As a measure of efficiency, we computed the relative of efficiency, using the complete data estimator $\widehat{t}_{\pi}$ given by (1), as the reference. That is,

$$RE_{MC}(\widehat{t}_{imp}) = 100 \times \frac{MSE_{MC}(\widehat{t}_{imp})}{MSE_{MC}(\widehat{t}_{\pi})}, \tag{41}$$

where $MSE_{MC}(\widehat{t}_{imp}) = R^{-1} \sum_{r=1}^{R} (\widehat{t}_{imp}^{(r)} - t_y)^2$ and $MSE_{MC}(\widehat{t}_{\pi})$ is defined similarly.

4.2 Simulation results

In Section 4.2.1, we discuss the simulation results pertaining to the continuous survey variables $Y_1, \ldots, Y_6$ and Y_8, with simple random sampling without replacement. The results for Poisson sampling used in the case of Y_4 and Y_7 are discussed in Section 4.2.2. Finally, the case of the binary variables Y_9 and Y_{10}, whose totals were estimated with simple random sampling without replacement, is discussed in Section 4.2.3.

4.2.1 Continuous survey variables and simple random sampling without replacement

For simple random sampling without replacement, for each of the twenty-seven imputation procedures, we had seven survey variables and four nonresponse mechanisms, leading to $27 \times 4 \times 27 = 756$ sets of simulation results. For ease of presentation, we present the results in tabular and graphic forms. The displayed statistical analyses were obtained from $4 \times 7 = 28$ scenarios obtained by crossing all the nonresponse models and the survey variables.

For each imputation procedure, Table 1 and Table 2 display, respectively, some descriptive statistics regarding the Monte Carlo absolute percent relative bias (absolute value of RB) and the Monte Carlo relative efficiency (RE) of $\widehat{t}_{imp}$ calculated across the twenty-eight scenarios. The corresponding side-by-side boxplots obtained from the twenty-eight scenarios are given in Figures 2 and 3. In Tables 1 and 2, the imputation procedures are ordered from the best to the worst with respect to the median absolute percent RB (the median of the twenty-eight values of absolute RB) and the median percent RE (the median of the twenty-eight values of RE), respectively. Figure 4 shows the distribution of the imputed estimator for the best ten imputation procedures in terms of RE. Finally, Table 3 displays the best five imputation procedures for each Y-variable.

From Table 1 and Table 2, among the twenty-seven imputation procedures, the best methods were: CUBIST, XGboost, AMS and BART. The performance of CUBIST3 was especially impressive with a median RE of 115%, a value of Q_{95} equal to 158% and a maximum value of 211%. The methods XGboost, AMS and BART exhibited similar performances with values of median RE ranging from 122% and 129%. However, for some scenarios, these methods did not perform well. For instance, the procedure XGB2 showed a value of max RE of about 438%, whereas it was equal to 1728% for AM5. Results suggest that additive models

with 5 interiors knots perform better than those with 10 interior knots. The next group of imputation procedures includes SVR and RF, with values of median RE ranging from 141% and 151%. Again, for some scenarios, both methods displayed poor performances with values of max RE ranging from 322% to 1138%. The procedure CART was less efficient than RF2 and RF3. The procedure 1-NN did relatively well with a median RE equal to 194%. On the other hand, the procedure 5-NN was rather inefficient with a median RE of 229%, which suggests that KNN with survey data works well only with a small number of neighbour. Turning to mean and random hot-deck imputation within classes, the score method was outperformed by the aforementioned procedures. Among the different versions of MCW and HDWC, the procedure MWC50 (which corresponds to 20 classes) led to the best results. This is consistent with the results of Haziza and Beaumont (2007). As expected, the procedure HDWC50 was less efficient than MWC50 as random hot-deck imputation suffers from the imputation variance, arising from the random selection of donors within classes. Finally, for some scenarios, it is worth noting that some of the procedures were better than the complete data estimator. For instance, for SVR4, the minimum value of RE and the value of $Q_{0.05}$ were respectively equal to 82% and 89%, respectively (see Table 2). Finally, the results in Table 5 suggest that the best methods were CUBIST, XGBoost, additive models and BART, which is consistent with the discussion above.

For each of the best ten imputation procedures displayed Table 2, Figure 5 displays the distribution of $\widehat{t}_{imp}$ for each nonresponse mechanism. Figure 5 suggests that the nonresponse mechanism may have a considerable impact on the behavior of the imputed estimator. For instance, in our experiments, we note that most of the imputation procedures performed poorly in the case of the nonresponse mechanism (NR1). Notable exceptions were AMS5, BART and Cubist3. In particular, Cubist3 seemed to be insensitive to the nonresponse mechanism, which is a desirable feature.

Ranking	Model	Min	$Q_{0.05}$	$Q_{0.25}$	$Q_{0.5}$	$Q_{0.75}$	$Q_{0.95}$	Max
1	CUBIST3	0.0	0.0	0.0	0.1	0.9	2.8	3.5
2	AMS5	0.0	0.0	0.0	0.1	1.8	7.7	13.8
3	AMS10	0.0	0.0	0.0	0.1	1.8	7.6	13.5
4	CUBIST1	0.0	0.0	0.1	0.5	3.4	7.5	7.5
5	XGB1	0.0	0.0	0.2	0.6	1.8	4.2	5.4
6	MWC50	0.0	0.0	0.1	0.6	2.7	8.3	11.7
7	HDWC50	0.0	0.0	0.1	0.6	2.7	8.3	11.8
8	CUBIST2	0.0	0.0	0.1	0.6	3.6	7.5	7.5
9	BART	0.0	0.1	0.4	0.8	2.2	4.0	4.6
10	XGB2	0.1	0.2	0.4	0.9	2.8	5.4	10.1
11	LR	0.0	0.0	0.1	0.9	3.8	12.8	20.4
12	SVR3	0.1	0.1	0.4	1.0	3.2	7.1	13.5
13	MWC100	0.0	0.0	0.3	1.0	3.6	10.1	12.9
14	HDWC100	0.0	0.0	0.3	1.0	3.6	10.1	12.9
15	SVR1	0.0	0.1	0.4	1.2	3.4	7.4	14.0
16	RF3	0.0	0.2	0.5	1.3	3.8	16.6	20.7
17	RF2	0.0	0.1	0.4	1.4	4	15.6	18.6
18	MWC250	0.0	0.0	0.7	1.7	4.9	14.6	18.1
19	HDWC250	0.0	0.0	0.6	1.7	4.9	14.6	18.1
20	RF1	0.1	0.2	0.9	1.7	7.7	32.1	39.5
21	NN	0.0	0.1	1.0	2.1	5.2	8.0	9.4
22	MWC500	0.0	0.0	0.7	2.2	7.2	25.5	30.6
23	CART	0.0	0.1	0.1	2.4	4.9	17.4	28.0
24	X5NN	0.0	0.2	1.5	3	7.3	12.0	13.7
25	SVR2	0.1	0.2	1.0	3.7	11.7	19.9	27.0
26	XGB3	0.6	1.5	3.1	4.3	5.0	9.5	10.3
27	SVR4	0.0	0.0	2.4	5.3	7.8	22.2	33.3

Table 1: Monte Carlo percent absolute relative bias of the imputed estimator: Descriptive statistics over all the scenarios

Ranking	Model	Min	$Q_{0.05}$	$Q_{0.25}$	$Q_{0.5}$	$Q_{0.75}$	$Q_{0.95}$	Max
1	CUBIST3	102	102	111	115	125	158	211
2	BART	113	113	116	122	131	154	204
3	AMS5	100	101	111	123	147	378	1728
4	AMS10	100	101	112	123	167	1195	1749
5	XGB1	101	103	115	129	153	203	288
6	CUBIST2	102	103	119	133	187	360	365
7	XGB2	102	102	117	133	166	316	438
8	CUBIST1	103	105	120	136	182	360	365
9	SVR1	94	103	122	141	180	284	322
10	SVR3	95	106	122	143	181	269	299
11	RF3	115	118	131	149	192	919	1138
12	RF2	113	118	130	151	202	824	1025
13	CART	125	134	143	168	248	1498	2683
14	LR	110	111	114	169	315	823	3494
15	MWC50	113	114	122	171	205	308	583
16	HDWC50	120	120	128	189	240	332	600
17	MWC100	116	116	136	191	217	296	670
18	NN	101	111	125	194	378	486	526
19	XGB3	92	100	128	194	663	1082	1104
20	HDWC100	123	125	142	213	246	322	686
21	RF1	136	137	149	223	375	3656	3916
22	MWC250	128	130	159	229	279	383	1162
23	5NN	94	108	123	229	659	775	855
24	SVR2	97	102	151	242	1616	3849	6355
25	SVR4	82	89	117	258	1439	4301	8675
26	HDWC250	141	143	185	265	325	411	1184
27	MWC500	151	155	202	269	336	1783	3021

Table 2: Monte Carlo percent absolute relative efficiency of the imputed estimator: Descriptive statistics over all the scenarios

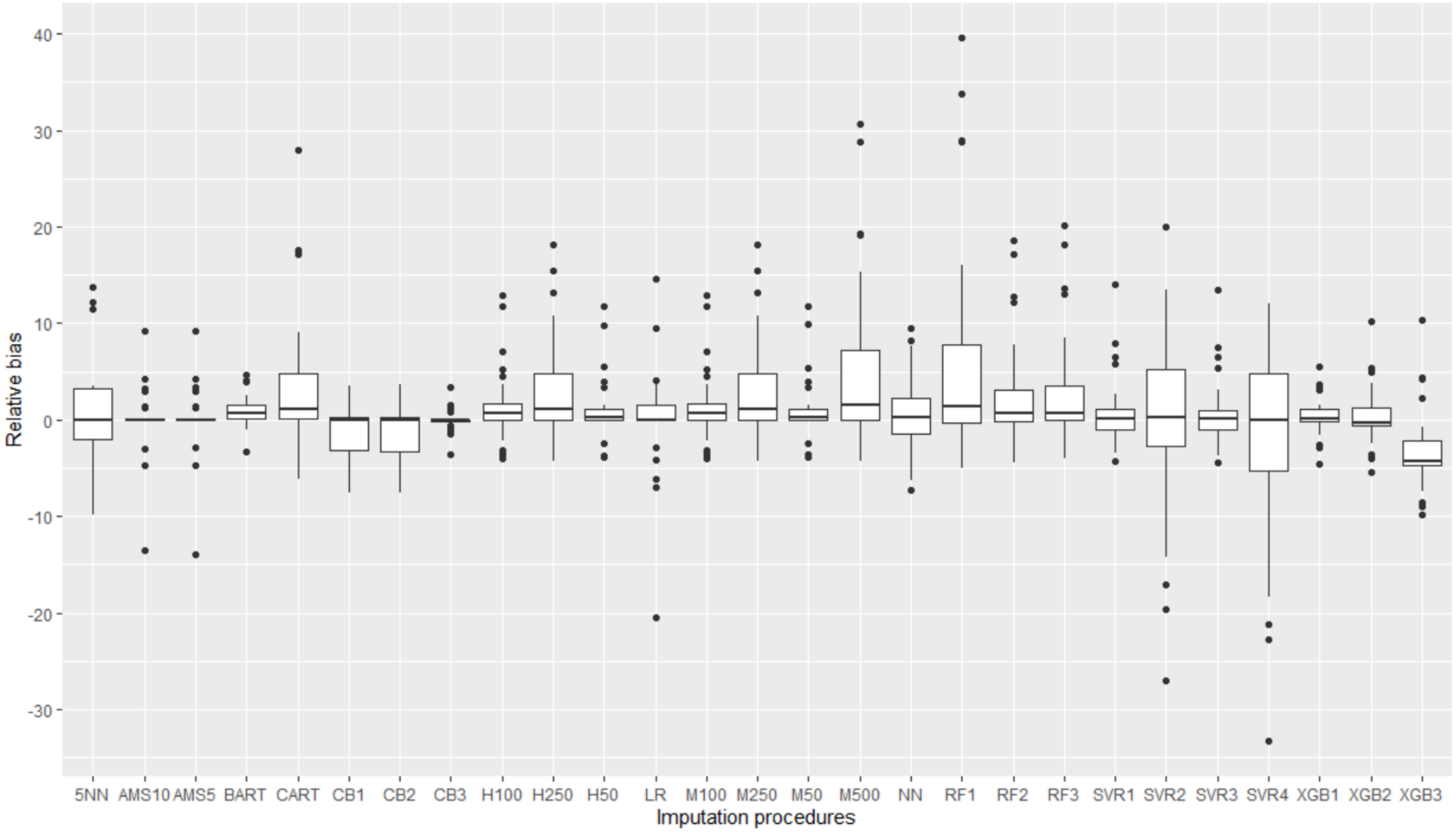

Figure 2: Monte Carlo percent relative bias across the scenarios.

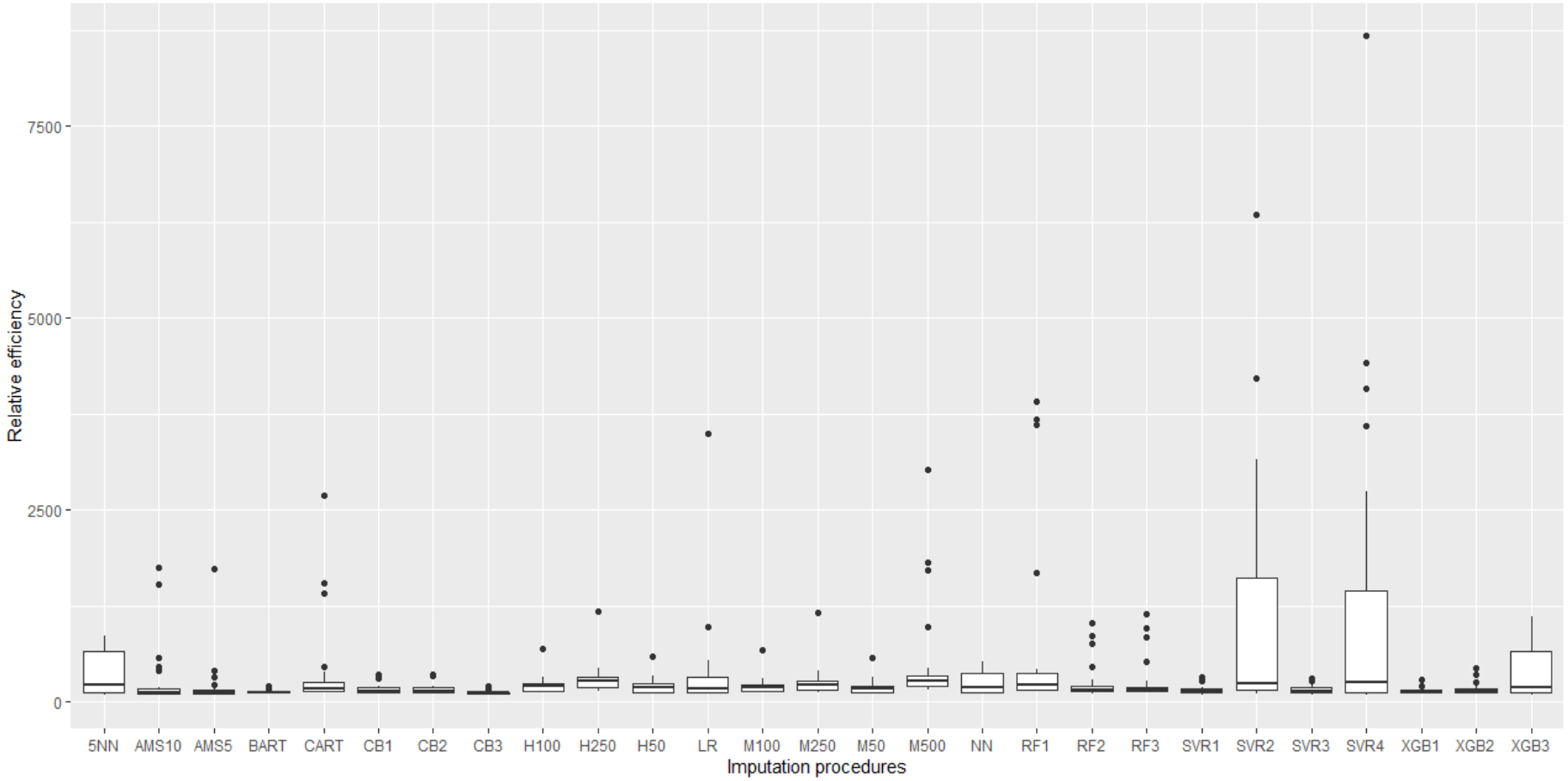

Figure 3: Monte Carlo percent relative efficiency across the scenarios.

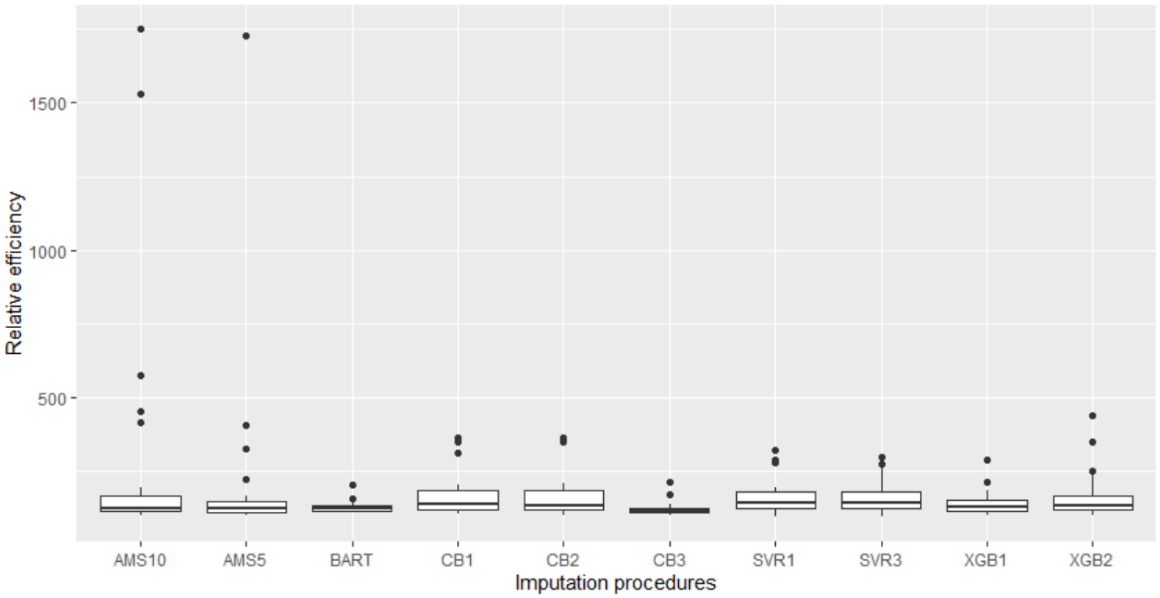

Figure 4: Monte Carlo percent relative efficiency across the scenarios: the best 10 procedures.

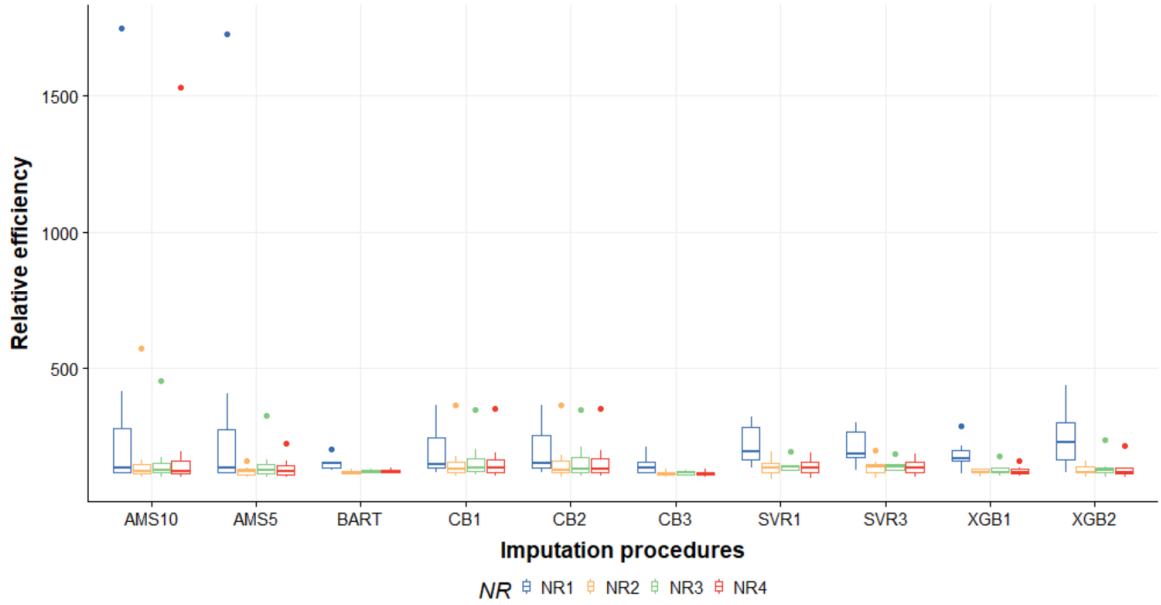

Figure 5: The effects of the nonresponse mechanism on the performance of the 10 best imputation procedures.

4.2.2 Continuous survey variables with Poisson sampling

Recall that Poisson sampling was used for estimating the population total of the survey variables Y_4 and Y_7. This led to $2 \times 4 \times 27 = 216$ sets of results. Due to the small number of scenarios ($2 \times 4 = 8$) for each of the survey variables Y_4 and Y_7, Tables 4 and 5 show the minimum, the median and the maximum Monte Carlo percent absolute RB and Monte Carlo

Ranking	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8
1	LR	CUBIST3	AMS5	BART	XGB3	CUBIST3	CUBIST3	CUBIST3
2	CUBIST3	LR	AMS10	CUBIST3	AMS5	BART	AMS5	AMS5
3	MW50	AMS5	BART	CUBIST1	AMS10	SVR3	AMS10	AMS10
4	AMS5	MWC50	CUBIST3	CUBIST2	XGB1	SVR1	MWC50	XGB1
5	AMS10	AMS10	CUBIST2	XGB1	XGB2	XGB1	BART	BART

Table 3: Best 5 imputation procedures for each survey variable.

percent RE only. The size variable X_5 used to obtain the first-order inclusion probabilities was included as a predictor in the imputation models. The results in Tables 4 and 5 were consistent with those obtained for simple random sampling without replacement. Again, the best methods were CUBIST3, BART and XGB1 in terms of either bias or efficiency.

Ranking	Model	Min	$Q_{0.5}$	Max
1	BART	0.1	0.9	3.0
2	CUBIST3	0.0	1	6.5
3	XGB1	0.0	2.4	5.2
4	CUBIST1	0.0	3.4	10.9
5	RF2	0.3	3.5	15.8
6	RF3	0.5	3.5	16.8
7	XGB2	0.4	3.9	8.6
8	AMS5	0.2	4.3	11.1
9	AMS10	0.2	4.3	10.7
10	CUBIST2	0.0	4.3	12.6
11	RF1	0.8	4.4	31.4
12	SVR3	0.1	4.4	6.7
13	LR	0.2	4.9	16.8
14	SVR1	0.1	4.9	7.1
15	MWC500	0.0	5.0	26.1
16	NN	0.0	5.0	7.3
17	MWC250	0.0	5.1	14.7
18	HDWC50	0.8	5.1	9.9
19	MWC50	0.0	5.2	9.9
20	MWC100	0.0	5.2	10.1
21	HDWC100	0.1	5.2	10.0
22	HDWC250	0.0	5.2	14.7
23	CART	0.2	5.6	24.6
24	5NN	1.3	7.1	11.7
25	XGB3	2.5	8.8	11.1
26	SVR2	1.0	11.7	22.6
27	SVR4	0.2	15.4	27.5

Table 4: Monte Carlo percent absolute relative bias of the imputed estimator: Descriptive statistics for Poisson sampling.

Ranking	Model	Min	$Q_{0.5}$	Max
1	BART	106	117	139
2	CUBIST3	111	118	239
3	XGB1	108	133	207
4	RF2	114	144	565
5	RF3	114	145	621
6	XGB2	110	156	246
7	SVR3	109	165	198
8	AMS5	124	168	486
9	SVR1	109	175	209
10	CUBIST1	114	175	469
11	NN	117	178	234
12	MWC50	125	188	396
13	MWC100	125	188	363
14	RF1	122	188	1868
15	LR	123	189	923
16	MWC250	128	190	525
17	CUBIST2	111	193	548
18	CART	133	198	1224
19	MWC500	133	198	1346
20	HDWC50	135	210	409
21	HDWC100	139	213	381
22	HDWC250	145	217	539
23	5NN	120	241	370
24	XGB3	116	272	441
25	AMS10	130	313	592
26	SVR2	142	493	1619
27	SVR4	141	769	2119

Table 5: Monte Carlo percent relative efficiency of the imputed estimator: Descriptive statistics for Poisson sampling.

4.2.3 Binary survey variables

In this section, we present the results pertaining to the binary variables Y_9 and Y_{10}. Again, for each imputation procedure, we obtained $2 \times 4 = 8$ sets of results. Tables 6 and 7 show the minimum, the median and the maximum Monte Carlo percent absolute RB and Monte Carlo percent RE, respectively.

The ranking for binary survey variables was slightly different from that obtained for the continuous survey variables. Nearest-neighbor (NN) imputation procedure was the best in terms of bias and efficiency. Recall that NN imputation did not rank among the best procedures for the continuous variables. NN imputation was followed by CUBIST, XGBOOST and BART.

Ranking	Model	Min	$Q_{0.5}$	Max
1	NN	136	144	428
2	XGB3	153	165	860
3	XGB2	156	167	827
4	CUBIST3	156	167	841
5	XGB1	156	171	932
6	BART	156	173	1052
7	5NN	152	174	1191
8	CUBIST2	163	179	873
9	CUBIST1	169	191	904
10	RF2	158	192	1572
11	RF3	162	198	1769
12	AMS5	169	219	2453
13	MWC100	160	221	1120
14	MWC50	159	222	1067
15	SVR1	171	222	3196
16	AMS10	165	223	2472
17	MWC50	159	223	1061
8	M100	159	225	1116
19	CART	176	229	1882
20	LR	164	230	2707
21	MWC250	172	244	1460
22	MWC250	173	246	1471
23	SVR3	191	280	2899
24	RF1	190	305	4666
25	M500	186	365	4977
26	SVR4	219	409	26429
27	SVR2	413	1839	17279

Table 6: Monte Carlo percent relative efficiency of the imputed estimator: Descriptive statistics for the binary survey variables.

4.3 High-dimensional setting

In this section, we investigate the performance of a subset of the imputation procedures considered in Section 4.1 in a high-dimensional setting. To that end, we used data from the Irish Commission for Energy Regulation (CER) Smart Metering Project conducted in 2009-2010 (CER, 2011) that focused on energy consumption and energy regulation[1]. About 6000 smart meters were installed in Irish residences and businesses. The customer's electrical consumption was collected every half an hour over a period of about two years.

We considered a subset of the original data set. We ended up with a population of $N = 6291$ smart meters (households and businesses) for a period of 14 consecutive days. For each population unit i (household or business), we had $2 \times 7 \times 48 = 672$ measurements denoted by $X_j = X(t_j), j = 1, \ldots 672$. Each of these 672 measurements represents the

[1]The data are available on request at: `https://www.ucd.ie/issda/data/commissionforenergyregulationcer/`.

Ranking	Model	Min	$Q_{0.5}$	Max
1	NN	0.0	0.5	3.6
2	CUBIST3	0.02	0.7	6.7
3	XGB3	0.03	0.8	7.7
4	BART	0.1	0.8	8.8
5	XGB1	0.14	0.9	7.9
6	XGB2	0.0	0.9	6.9
7	5NN	0.0	1.0	7.3
8	CUBIST2	0.2	1.0	7.0
9	CUBIST1	0.0	1.1	6.8
10	RF2	0.12	1.5	10.3
11	RF3	0.13	1.6	11.0
12	AMS5	0.04	1.6	11.9
13	AMS10	0.1	1.6	11.9
14	SVR1	0.3	1.7	12.0
15	LR	0.19	1.8	12.3
16	CART	0.18	1.8	11.4
17	MWC50	0.0	1.8	7.5
18	MWC100	0.0	1.8	7.7
19	HDWC50	0.03	1.8	7.5
20	HDWC100	0.01	1.8	7.7
21	MWC250	0.0	2.0	9.4
22	HDWC250	0.0	2.0	9.4
23	SVR3	0.43	2.3	11.5
24	RF1	0.08	2.7	19.0
25	SVR4	0.17	3.0	36.5
26	MWC500	0.0	3.2	16.4
27	SVR2	1.9	9.5	33.9

Table 7: Monte Carlo percent absolute relative bias of the imputed estimator: Descriptive statistics for the binary survey variables.

electricity consumption (in kW) at instant t_j. We denote by x_{ij} the value of X_j recorded by the smart meter i for $i = 1, \ldots, N$ at instant t_j. It should be noted that these variables were highly correlated among themselves with a condition number of the matrix $N^{-1}\mathbf{X}^T\mathbf{X}$ computed using all the data, of about 60.000.

We created four survey variables based on a subset of the auxiliary variables $X_1, \ldots, X_{672}$:

$$Y_1 = 400 + 2X_1 + X_2 + 2X_3 + \mathcal{N}(0, 1500);$$

$$Y_2 = 400 + X_1X_2 + 2X_3 + \mathcal{N}(0, 1500);$$

$$Y_3 = 500 + 2X_4 + 400\mathbb{1}_{\{X_5>156\}} - 400\mathbb{1}\left(X_5 \leqslant 156\right) + 1000\mathbb{1}\left(X_2 > 190\right)$$

$$+ 300\mathbb{1}\left(X_5 > 200\right) + \mathcal{N}(0, 1500);$$

$$Y_4 = 1 + \cos(2X_1 + X_2 + 2X_3)^2 + \epsilon_1,$$

where $\epsilon_1 \sim \mathcal{E}(2)$ and these error terms were centered so as to have a mean equal to zero. We were interested in estimating the population total of the survey variables Y_1-Y_4. Again, the simulation was based of $R = 5,000$ iterations of the process described in Section 4. Samples of size $n = 1000$ were selected according to simple random sampling without replacement. The missing values to the survey variables Y_1-Y_4 were generated according to

$$p_i = 0.1 + 0.89 \times \text{sigmoid}\left\{-0.83 + 0.001 \times (2x_{i1} + 2x_{i2} - 2.5x_{i3})\right\},$$

leading to an average response rate of about 50%.

Three high and very high dimensional settings were considered: in the first setting, the imputation models used the first 15 auxiliary variables $X_1, ..., X_{15}$, in the data set. In the second and third settings, the imputation models were based on the first 100 and 300 auxiliary variables $X_1, ..., X_{100}$, and $X_1, ..., X_{300}$, respectively.

To impute the missing values, we confined to a subset of the imputation procedures considered in Section 4.1: additive models, BART, CUBIST, XGBoost, random forests, nearest-neighbour imputation and support vector regression. Linear regression imputation and mean imputation within 20 classes were also considered. It is well known that the quality of predictions based on linear models tend to deteriorate substantially in the presence of a very large number of auxiliary variables. To cope with this issue, we also considered principal components analysis as a reduction-dimension method; see Cardot et al. (2017).

Table 8 shows the Monte Carlo percent relative bias (RB) and relative efficiency (RE) for $p = 15$ predictors. Table 9 shows the results for $p = 100$ and $p = 300$ predictors. For each scenario, the best imputation procedures are highlighted in bold. Note that the relative efficiency is now computed with respect to the mean square error of the imputed estimator based on the true imputation model. The additive models were considered in the first setting only ($p = 15$ variables) because their performance deteriorated rapidly as the number p of variables increased. For $p = 100$ and $p = 300$ the backfitting algorithm did not reach convergence in most scenarios.

From Tables 8 and 9, we note that CUBIST and XGBoost were the best method in the vast majority of the scenarios. These methods were followed by BART and random forests. As expected, additive models performed poorly, which illustrates the curse of dimensionality. It is worth pointing out that random forests performed better in the high-dimensional setting than they did in the low-dimension setting considered in section 4.1. Finally, the strategy

based on principal components analysis did relatively well in most scenarios.

Variable	Criterion	LR	MWC50	RF2	XGB1	NN	SVR3	AMS5	CB3	PCR1	PCR2	PCR3	BART
Y_1	RE	**100**	117	110	**103**	111	124	**101**	**100**	160	113	**100**	**101**
	RB	-0,18	1,7	1,7	0	-0,1	2,6	-0,0	-0,1	4,0	0,6	-0,5	0,3
Y_2	RE	184	176	**103**	**100**	**100**	295	7041	**101**	159	213	207	106
	RB	-44,3	15,7	3,8	0	0,7	19,2	9,5	-0,0	-47,0	-53,1	-48,5	2,1
Y_3	RE	190	135	**102**	108	128	134	403	109	188	178	210	**105**
	RB	4,6	2,1	0,1	-0,2	0,1	2,08	-0,0	1,2	4,6	4,3	5,2	0,0
Y_4	RE	125	126	143	147	188	195	130	**118**	**119**	121	123	131
	RB	-0,0	-0,0	0,5	0,2	-0,1	-1,3	0,0	-0,0	-0,11	-0,1	-0,0	0,0

Table 8: Relative biais (RB) and relative efficiency (RE) of imputation procedures with $p = 15$ auxiliary variables.

Variable	Dim	Criterion	LR	MWC50	RF2	XGB1	NN	SVR3	CB3	PCR1	PCR2	PCR3	BART
Y_1	p=100	RE	**102**	122	149	**103**	216	187	**100**	269	226	151	**105**
		RB	0,14	2,1	4,2	0,3	6,2	5,1	0	7,8	6,6	4,0	0,6
Y_2	p=100	RE	115	287	**109**	**100**	**100**	340	**100**	**100**	**108**	140	127
		RB	-23,8	34,3	7,5	0,1	3,3	26,1	-0,0	-31,0	-28,9	-32,5	5,8
Y_3	p=100	RE	158	185	**107**	**107**	354	162	**108**	236	224	196	129
		RB	3,2	3,9	1,1	-0,0	7,0	3,4	0,9	5,9	5,5	4,8	7,7
Y_4	p=100	RE	140	141	151	146	243	217	**122**	**120**	**120**	**121**	135
		RB	0,0	0,1	0,7	0,28	0,4	-1,5	-0,0	-0,0	-0,1	-0,1	-0,0
Y_1	p=300	RE	120	215	190	103	286	237	**100**	290	262	189	**110**
		RB	-0,2	1	5,7	0,6	7,05	6,7	0,06	8,3	7,7	5,7	1,3
Y_2	p=300	RE	**102**	1106	112	**100**	**100**	405	**100**	**91**	**85**	109	243
		RB	-6,3	89,1	9,5	0,1	4,01	35,	-0,0	-28,4	-25,3	-26,9	4,6
Y_3	p=300	RE	197	378	118	107	630	180	**108**	350	245	224	242
		RB	1,0	6,7	2,0	0,0	9,1	4,1	0,8	6,2	6,1	5,6	6,4
Y_4	p=300	RE	276	584	155	143	443	214	**124**	**120**	**120**	**121**	131
		RB	0,1	2,4	0,7	0,3	0,6	-1,5	0,06	-0,0	-0,1	-0,1	-0,0

Table 9: Relative biais (RB) and relative efficiency (RE) of imputation procedures with $p = 100$ and respectively, $p = 300$ auxiliary variables.

5 Simulation study: the case of population quantiles

In this section, we turn our attention to population quantiles. Except for nearest-neighbour imputation, we confined to the random versions of the imputation procedures described in Section 3. The target parameters were the quantiles of order $\gamma_1 = 0.25$, $\gamma_2 = 0.5$ and $\gamma_3 = 0.75$ that correspond to the first quartile, the median and the third quartile, respectively. We considered a subset of the scenarios described in Section 4.1. First, we confined to the case of

the survey variables Y_3 and Y_6 and the nonresponse mechanisms (NR1) and (NR3) described in Section 4.1, leading to $2 \times 2 = 4$ scenarios. Also, samples were selected according to simple random sampling without replacement only. In each sample, we computed the imputed estimator $\widehat{\mathcal{Q}}_{\gamma,imp}$ given by (5) for $\gamma_1 = 0.25$, $\gamma_2 = 0.5$ and $\gamma_3 = 0.75$. As in Section 4, we computed the Monte Carlo percent relative bias of $\widehat{\mathcal{Q}}_{\gamma,imp}$ and the relative efficiency, given respectively by (40) and (41) with $\widehat{t}_{imp}$ replaced with $\widehat{\mathcal{Q}}_{\gamma,imp}$, $\widehat{t}_{\pi}$ replaced with $\widehat{\mathcal{Q}}_{\gamma}$ and t_y replaced with Q_{γ}.

The results are presented in Figures 6-8. In each figure, the x-axis corresponds to the median of the Monte Carlo percent relative bias of $\widehat{\mathcal{Q}}_{\gamma,imp}$ computed across the 4 scenarios, whereas the y-axis corresponds to the median of the Monte Carlo relative efficiency. For the purpose of clarity, we have excluded from Figures 6-8 any imputation procedure whose median of the Monte Carlo percent relative bias lied outside the interval $[-20; 20]$ or whose median of the Monte Carlo relative efficiency was above 500.

From Figures 6-8, Cubist displayed a very good performance in terms of bias and efficiency for the three quantiles. The procedure XGBoost led to good results for $Q_{0.25}$ and $Q_{0.75}$ but performed poorly for $Q_{0.5}$. Similarly, BART performed very well for both $Q_{0.5}$ and $Q_{0.75}$ but exhibited a poor performance for $Q_{0.25}$. Support vector machine (SVR3) did relatively well for both $Q_{0.5}$ and $Q_{0.75}$ but was outperformed by Cubist and XGBoost for $Q_{0.25}$. Again, the Cubist algorithm seemed to be insensitive to the target parameter, the model that has generated the Y-variable and the nonresponse mechanism, at least in our experiments.

6 Final remarks

In this paper, we have conducted an extensive simulation study to compare several non-parametric and machine learning imputation procedures in terms of bias and efficiency. The imputation procedures were evaluated in the case of finite population totals of continuous and binary variables and for population quantiles under both simple random sampling without replacement and proportional-to-size Poisson sampling. The Cubist algorithm, BART and XGBoost performed very well in a wide variety of settings. In general, these methods seem to be highly robust to model misspecification and seem to have the ability to capture nonlinear trends in the data. Additive models based on B-splines performed well in the case of population totals when the number of explanatory variables was small but broke down

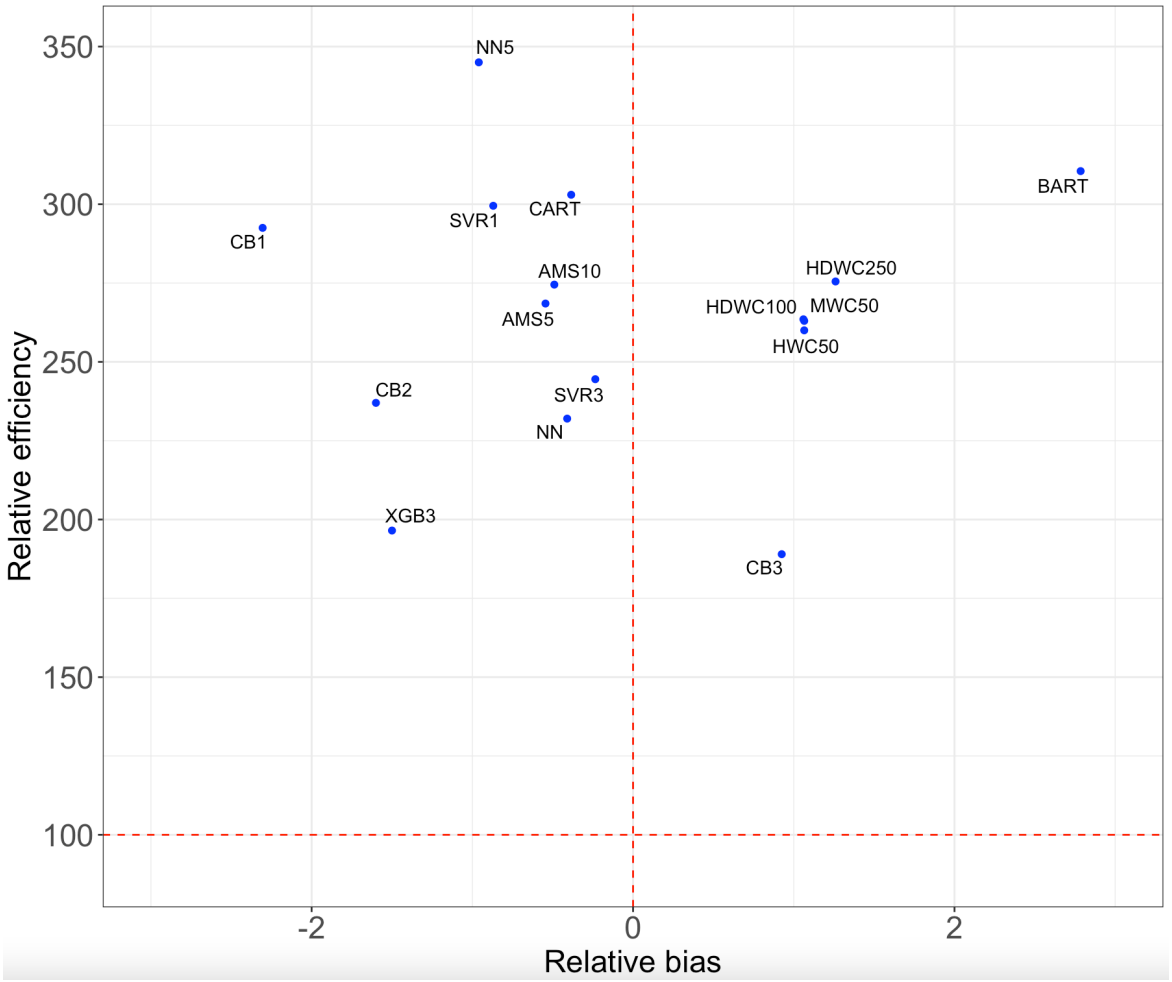

Figure 6: Median performances of the best imputed estimators for the estimation of $\mathcal{Q}_{0.25}$.

for large values of p. Finally, random forests performed relatively well in a high-dimensional setting. In practice, the choice of an imputation procedure is not clear-cut and depends on the data at hand. If one is reasonably confident about the correct specification of the first moment of the imputation model (that includes the correct specification of the functional form and the correct specification of the vector of explanatory variables), parametric imputation procedures are expected to do well in terms of bias and efficiency. In addition, parametric imputation is simpler to understand and the results are easier to interpret, in general. In the case of complex/nonlinear relationships and/or in a high-dimensional setting, our empirical investigations suggest that machine learning procedures outperform traditional imputation procedures as they tend to be robust against model misspecification. However, these procedures require the specification of some regularization parameters. For instance, for XGBoost, one must specify the learning rate, the maximal depth and the coefficient of penalization. In support vector regression, the cost function and the kernel function must be

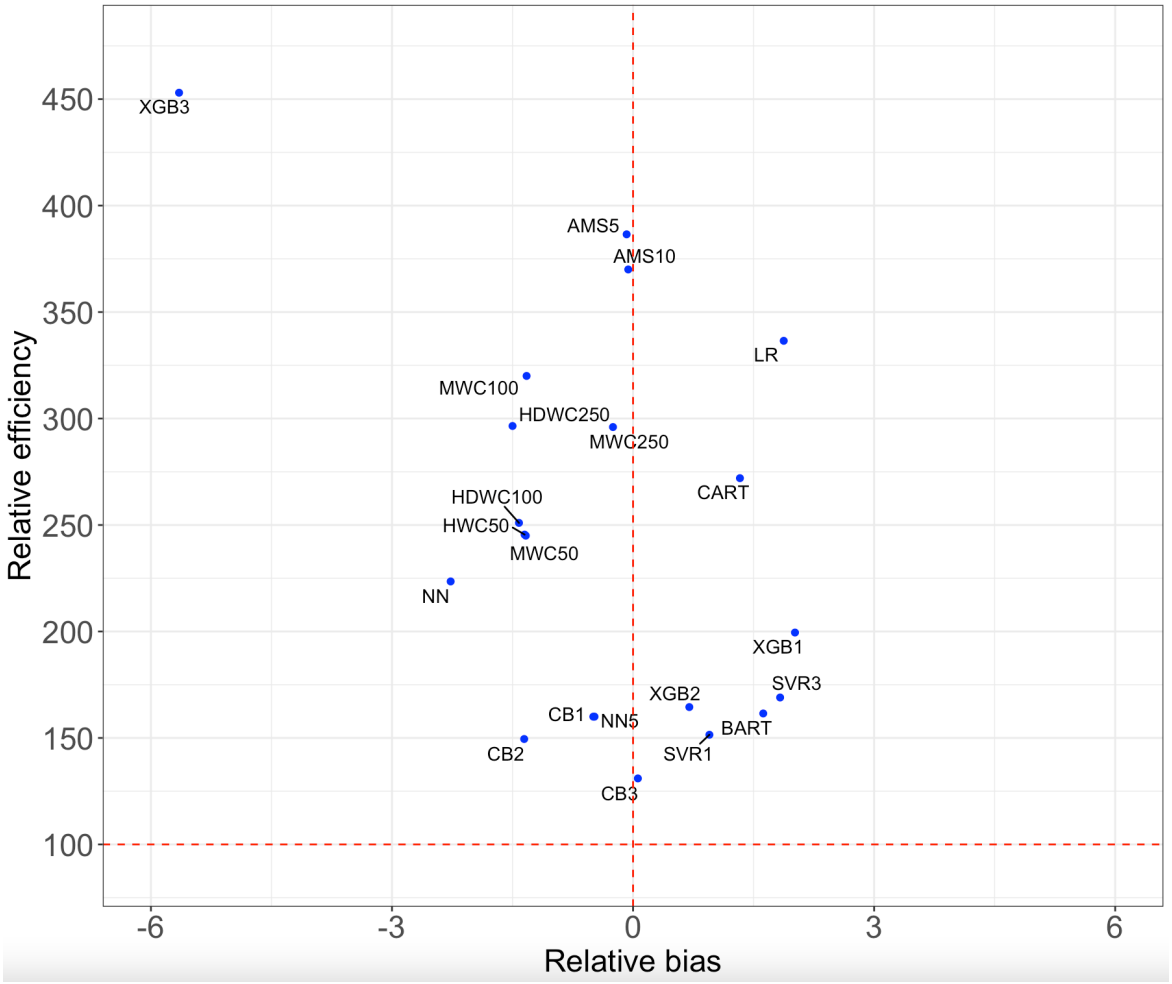

Figure 7: Median performances of the best imputed estimators for the estimation of $\mathcal{Q}_{0.5}$.

selected, among others. In practice, the value for some of these parameters are determined through a cross-validation procedure. To keep the processing time at a reasonable level, all the regularization parameters were predetermined in our experiments. Overall, it seems that Cubist is an excellent choice as it performed well in all the scenarios, unlike its main competitors (e.g., XGBoost, BART, random forest, etc.) whose performance varied from one scenario to another. From a computational point of view, most procedures were efficient. One notable exception is BART that proved to be highly computer intensive with an average processing time approximately twenty times larger than what was required for the other procedures.

Drawing inferences from survey data requires a variance estimate. It is well known that imputed values should not be treated as observed values. Otherwise, the resulting variance estimates tend to be much smaller, on average, than the true variance, especially if the non-response rates are appreciable. In the last three decades, a number of variance estimation procedures have been proposed for obtaining variance estimates that account for sampling,

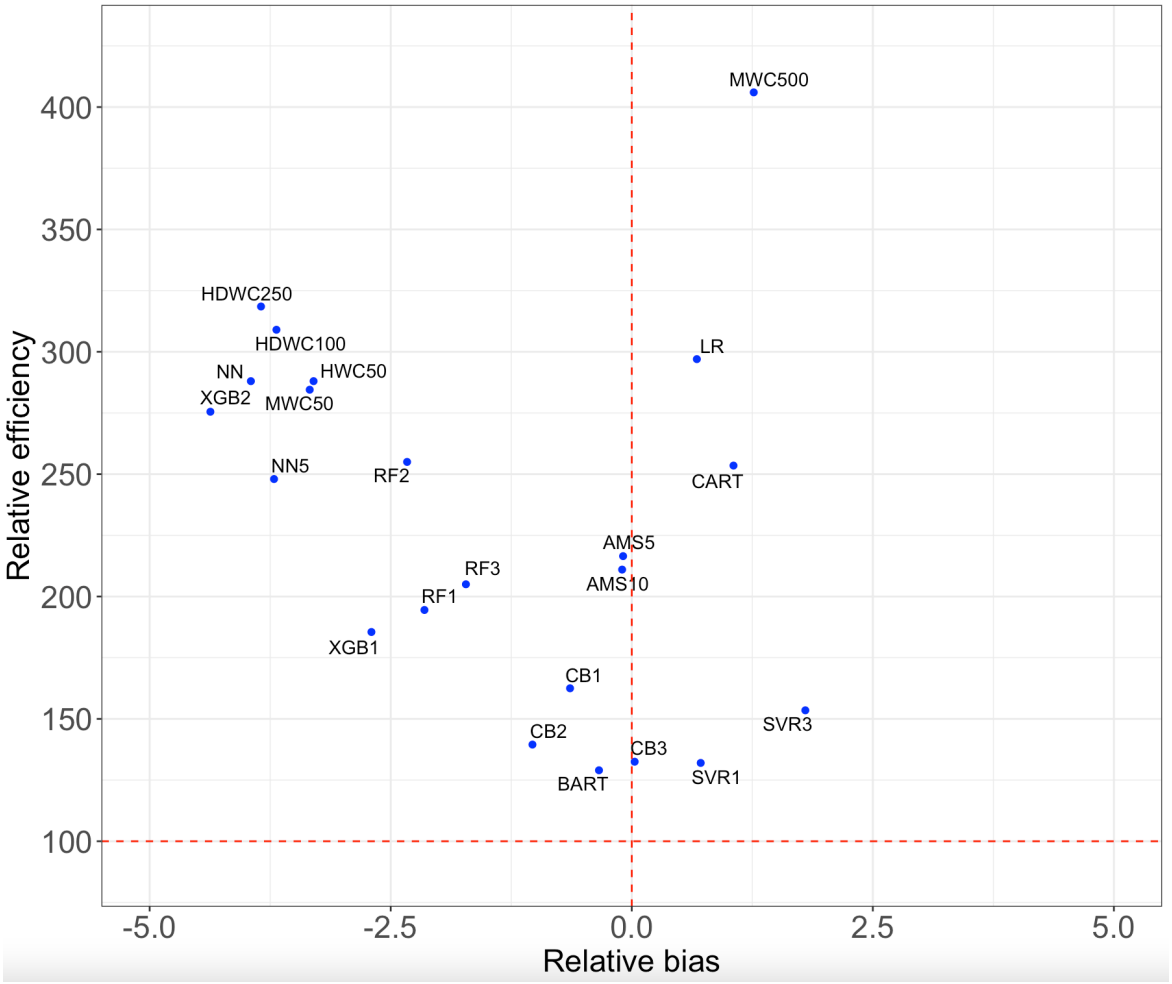

Figure 8: Median performances of the best imputed estimators for the estimation of $\mathcal{Q}_{0.75}$.

nonresponse and imputation. The reader is referred to Haziza and Vallée (2020) for a comprehensive overview of variance estimation procedures in the presence of singly imputed data sets. Estimating the variance of imputed estimators obtained through machine learning procedures is challenging and requires further research. If the sampling fraction is negligible, one can recourse to the bootstrap procedure of Shao and Sitter (1996) that consists of selecting bootstrap samples according to a complete data bootstrap procedure and reimputing the missing values within each bootstrap sample using the same imputation method that was used on the original data. If a machine learning procedure is used to impute the missing data, the Shao-Sitter procedure may be highly computer intensive. When the sampling fraction is not negligible, the problem of bootstrap variance estimation is more intricate (Chen et al., 2019). To make the variance estimation process simpler for survey practitioners, it would be desirable to derive a "universal" variance estimator based on Taylor expansion procedures

that could be applicable to a wide class of machine learning imputation procedures, at least in the case of negligible sampling fractions. This is currently under investigation.

Investigating the performance of deep learning methods in the context of imputation for missing survey data would constitute a promising direction for future research. There exist a wide class of deep learning procedures based on relatively sophisticated algorithms that proved to be extremely efficient in the context of unstructured data such as signal processing or text analysis. However, for deep learning procedures to "shine" in terms of efficiency typically requires a huge volume of unstructured data, which is seldom the case in surveys. In practice, most data sets in surveys consist of structured data and contains, at most, a few millions observations and a few hundred survey variables. As noted by Choley (2018):
"(...) *gradient boosting (such as XGBoost) is used for problems where structure data is available, whereas deep learning is used for perceptual problems such as image classification*".
We believe that the class of imputation procedures considered in this article, that includes bagging and boosting among others, offers a number of very good options that may be applicable to virtually all the surveys conducted by NSOs.

References

Beaumont, J.-F. and Bocci, C. (2009). Variance estimation when donor imputation is used to fill in missing values. *Canad. J. Statist.*, 37:400–416.

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

Binder, D. A. (1983). On the variances of asymptotically normal estimators from complex surveys. *International Statistical Review*, 51:279–292.

Breiman, L. (1984). *Classification and regression trees*. Routledge.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Chapman & Hall/CRC, Boca Raton.

Cardot, H., Goga, C., and Shehzad, M.-A. (2017). Calibration and partial calibration on principal components when the number of auxiliary variables is large. *Statistica Sinica*, 27(243-260).

Chen, J. and Shao, J. (2000). Nearest neighbor imputation for survey data. *Journal of official statistics*, 16(2):113.

Chen, S. and Haziza, D. (2019). Recent developments in dealing with item non-response in surveys: A critical review. *International Statistical Review*, 87:S192–S218.

Chen, S., Haziza, D., Léger, C., and Mashreghi, Z. (2019). Pseudo-population bootstrap methods for imputed survey data. *Biometrika*, 106(2):369–384.

Chen, T. and Guestrin, C. (2016). XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16*. ACM Press.

Chipman, H., George, E., and McCulloch, R. (2010). BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298.

Choley, F. (2018). *Deep learning with Python*. Manning.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Creel, D. and Krotki, K. (2006). Creating imputation classes using classification tree methodology. In *In Proc. Surv. Res. Methods Sect., Am. Stat. Assoc.*, pages pp. 2884–2887.

Dagdoug, M., Goga, C., and Haziza, D. (2020a). Model-assisted estimation through random forests in finite population sampling. *in revision*. arXiv preprint arXiv:2002.09736.

Dagdoug, M., Goga, C., and Haziza, D. (2020b). Random forest imputation in surveys and application to data integration. *in work*.

De Moliner, A. and Goga, C. (2018). Sample-based estimation of mean electricity consumption curves for small domains. *Survey Methodology*, 44(2):193–214.

Díaz-Uriarte, R. and de Andrés, S. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3.

Dierckx, P. (1993). *Curves and Surface Fitting with Splines*. Oxford: Clarendon.

Fraiwan, L., Lweesy, K., Khasawneh, N., Wenz, H., and Dickhaus, H. (2012). Automated sleep stage identification system based on time–frequency analysis of a single EEG channel

and random forest classifier. *Computer Methods and Programs in Biomedicine*, 108(1):10–19.

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232.

Gelein, B. (2017). *Handling missing data with superpopulation model, design-based approach and machine learning*. PhD thesis, Université Bretagne Loire.

Goga, C., Haziza, D., and Dagdoug, M. (2019). B-spline based imputation procedures for the treatment of item nonresponse in surveys. In work.

Grimm, R., Behrens, T., Märker, M., and Elsenbeer, H. (2008). Soil organic carbon concentrations and stocks on barro colorado island — digital soil mapping using random forests analysis. *Geoderma*, 146(1-2):102–113.

Hamza, M. and Larocque, D. (2005). An empirical comparison of ensemble methods based on classification trees. *Journal of Statistical Computation and Simulation*, 75(8):629–643.

Han, X. and Clemmensen, L. (2014). On weighted support vector regression. *Quality and Reliability Engineering International*, pages 891–903.

Hastie, T. and Tibshirani, R. (1986). Generalized additive models. *Statistical Science*, 1(3):297–310.

Hastie, T., Tibshirani, R., and Friedman, J. (2011). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, New York.

Haziza, D. (2009). Imputation and inference in the presence of missing data. In Pfeffermann, D. and Rao, C., editors, *Handbook of statistics*, volume 29A, pages 215–246. Elsevier.

Haziza, D. and Beaumont, J.-F. (2007). On the construction of imputation classes in surveys. *International Statistical Review*, 75(1):25–43.

Haziza, D. and Vallée, A.-A. (2020). Variance estimation in the presence of singly imputed data: A critical review. *To appear in the Japanese Journal of Statistics and Data Science.*

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2015). *An Introduction to Statistical Learning with Applications in R*. Springer Texts in Statistics.

Kane, M., Price, N., Scotch, M., and Rabinowitz, P. (2014). Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks. *BMC Bioinformatics*, 15(1).

Kern, C., Klausch, T., and Kreuter, F. (2019). Tree-based machine learning methods for survey research. *Survey Research Methods*, (13):73–93.

Kuhn, M. and Johnson, K. (2013). *Applied predictive modelling*. Springer.

Lee, D., Song, J.-H., Song, S.-O., and Yoon, E. S. (2005). Weighted support vector machine for quality estimation in the polymerization process. *Ind. Eng. Chem. res.*, pages 2101–2105.

Little, R. J. (1986). Survey nonresponse adjustments for estimates of means. *International Statistical Review/Revue Internationale de Statistique*, pages 139–157.

Lohr, S., Hsu, V., and Montaquila, J. (2015). Using classification and regression trees to model survey nonresponse. In *JSM Proceedings, Survey Research Methods Section, Alexandria, VA: American Statistical Association*, pages 2071–2085.

McConville, K. and Toth, D. (2019). Automated selection of post-strata using a model-assisted regression tree estimator. *Scandinavian Journal of Statistics*, 46(2):389–413.

Quinlan, J. (1993). Combining instance-based and model-based learning. In *Proceedings of the tenth international conference on machine learning*, pages 236–243.

Quinlan, J. et al. (1992). Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, pages 343–348. World Scientific.

Rubin, D. (1976). Inference and missing data. *Biometrika*, 63(3):581–592.

Ruppert, D., Wand, M. P., and Carroll, R. J. (2003). *Semiparametric regression*, volume 12 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge.

Schumaker, L. L. (1981). *Spline Functions: Basic Theory*. New York: Wiley.

Scornet, E. (2017). Tuning parameters in random forests. *ESAIM: Proceedings and Surveys*, 60:144–162.

Scornet, E., Biau, G., and Vert, J.-P. (2015). Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741.

Shao, J. and Sitter, R. R. (1996). Bootstrap for imputed survey data. *Journal of the American Statistical Association*, 91(435):1278–1288.

Smola, A. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

Tipton, J., Opsomer, J., and Moisen, G. (2013). Properties of endogenous post-stratified estimation using remote sensing data. *Remote sensing of environment*, 139:130–137.

Vapnik, V. (1998). *Statistical Learning Theory*. WILEY.

Vapnik, V. (2000). *The Nature of Statistical Learning Theory*. Springer New York.

Wang, J. C. and Opsomer, J. D. (2011). On asymptotic normality and variance estimation for nondifferentiable survey estimators. *Biometrika*, 98(1):91–106.

Yang, S. and Kim, J. K. (2019). Nearest neighbor imputation for general parameter estimation in survey sampling. In *The Econometrics of Complex Survey Data: Theory and Applications*, pages 209–234. Emerald Publishing Limited.

Zhou, S., Shen, X., and Wolfe, D. (1998). Local asymptotics for regression splines and confidence regions. *The Annals of Statistics*, 26(5):1760–1782.