Monday, April 17, 2023 Working paper. Please do not distribute without author consent.
Replication materials are stored at https://github.com/DamonCharlesRoberts/mice-imputation-psci

Seeing the leaves through the forest*

A peak under the hood using random forest models for missing data problems

Damon C. Roberts †
University of Colorado Boulder
damon.roberts-1@colorado.edu

ABSTRACT Political scientists often struggle with decisions about what to do with incomplete cases for their regression analyses, and one can often make several decisions that influence one's ultimate substantive conclusions. In many areas of research outside of political science and the social sciences, scholars take advantage of an extension of multiple imputation, which offers the choice to leverage machine learning models for predicting values in missing data. This manuscript provides a summary of missing data and its consequences for our regression models along with providing an explanation of how to implement random forest models with an expanded form of the multiple imputation procedure, called multiple imputation with chained equation to handle complex causes for non-random missingness in our data. After providing a primer on standard missing data procedures in political science and random forest with multiple imputation with chained equations, I examine its performance on simulated data. I conclude by providing recommendations for dealing with missing data in practice.

KEYWORDS Missing data; Multiple imputation; Machine learning

*I would like to thank Andrew Q. Philips, Jennifer Wolak, and Madeline Mader for their advice and many conversations during the development of this project as well as Andy Baker for offering me the space to write the manuscript and for his feedback. I would also like to thank the discussants and panelists at MPSA for their useful feedback and encouragement.

†Corresponding author.

Introduction

Missing values are common in social science data. King et al. (2001) estimate that political scientists lose about one-third of their data in their complete case regression analyses due to missing data. There are many reasons why missing values arise in our data. In surveys, these are referred to as item-nonresponse and pose threats to obtaining unbiased estimates for public opinion research when researchers utilize listwise deletion (`LWD`) in their regression analyses (Weisberg 2005); mainly when the researcher can predict the cause of the missingness with an observed cause - which scholars refer to the data as "missing at random" (King et al. 2001).[1] More generally, missing values come about in other types of data due to the unit's (e.g., country, respondent, politician) attempt to obfuscate information, data collector error, or researcher error. When common and not from a stochastic data generating process (DGP), missing data pose threats to causal inference through omitted variable bias where the missing value in the variables included in a statistical model can be predicted by another variable in your model.

As this poses a serious threat to our ability to make conclusions about causal processes, several political scientists generate tools and learn from other fields to deal with these challenges due to the severity of the consequences for inference. There are four primary approaches that political scientists use in dealing with missing data: (1) listwise deletion or complete case analysis (LWD), (2) simple mean or median based imputation, (3) hot deck or regression-based imputation, or (4) multiple imputation (MI). Many of the most popular approaches to imputation carry assumptions that scholars have to satisfy. This manuscript aims to provide a primer to political scientists on the use of machine learning models in the Multiple Imputation with Chained Equations (`MICE`)

1. Which is a common occurrence, more so than missingness caused entirely by random chance - which, if one meets this condition, gives unbiased estimates with LWD.

framework - an extension of the familiar MI approach.

Machine learning with `MICE` is not a new approach. While a JSTOR search of Political Science journals reports a handful of articles that discuss this particular approach, it is certainly not widely used by political scientists nor has it received much accessible discussion of how it works under the hood.[2]

As the present article discusses, `MICE` allows the researcher to choose various models to aid the imputation process. Selecting a model with its underlying assumptions provides many benefits for researchers in choosing a procedure that is *most* appropriate for their particular circumstances. Choosing models that allow for flexibility and models that do not have a number of restrictive assumptions benefit those unsure about the exact data-generating process of the missingness. The specific emphasis in this paper on using random forest models in `MICE` (`RF-MICE`) results from the strong preference of those who use machine learning models for predictive regressions to use random forest models due to their flexibility as a result of their fully non-parametric nature and for their performance under many different, and complex, circumstances (see Montgomery and Olivella 2018; James et al. 2013). Fields like the biomedical sciences treat missing values as out-of-sample predictions that random forest models predict; they see the purpose of imputation as aligned with a task that random forest models are optimal. As other fields use this `RF-MICE` procedure, there are implementations in `STATA`, `R`, and `Python`. All of which are relatively easy in terms of knowledge to code in either of these languages (Buuren and Groothuis-Oudshoorn 2011a). To be clear, this manuscript argues that we should not only consider using the `MICE` framework for imputation but also should consider using Random Forest models.

This manuscript compares the utility of random forest models in the `MICE` procedure for impu-

2. Though, see Marbach (2022).

tation to other common imputation tools used in political science. In doing this, the manuscript encourages political scientists to consider this procedure when faced with conditions where missing data are present, and the other common tools seem unsuitable. It is not to paint these models as superior in absolute terms to other procedures for imputation. To do that is a waste of time given the variety of circumstances where some methodological tools are suitable in some circumstances and not for others. This manuscript provides a primer encouraging political scientists to consider this tool when faced with missing data and to give them enough background so they may be comfortable using it. Furthermore, the manuscript agrees with the common recommendation that practitioners should recognize the value of reducing one's dependence on a single procedure and contends that one must consider using multiple procedures to reduce the dependency of one's results on any given procedure. This article provides code snippets that readers can use to implement all of these procedures in R.

The next section reviews common approaches handling missing data that political scientists currently use. The next section describes machine learning and random forest models and links this to my claim of their utility for predicting missing data when used in the `MICE` procedure. I then move into applied examples where I examine the performance of these random forest models to other common approaches to dealing with missing data on simulated data. I then discuss recommendations for when one should use the reigning popular techniques or the random forest application in the `MICE` procedure.

Types of missing data, imputation, and MI in Political Science

MCAR, MAR, and MNAR

Missing data arise in different forms. Researchers describe missing data in three ways - often using somewhat unintuitive acronyms. The first form missing data takes is Missing Completely At Random (`MCAR`). This means that the data generating process for the missingness is random - there are no observed or unobserved causes of missingness. The second form missing data takes is Missing At Random (`MAR`). In `MAR`, these data are missing due to some observed cause. However, they are "Missing at Random" once you account for that observed cause of missingness. Some argue that `MAR` is much more common given the state of how large most contemporary social science data sets are (Schunk 2008). The third form that missing data take is Missing Not At Random (`MNAR`)[3]. `MNAR` happens when observed and unobserved causes explain the missingness. What distinguishes `MNAR` from `MAR` is that the researcher does not have a clear path forward to handle the cause of missingness. This occurs either because the variable where there is missingness is, itself, a cause of the missingness, or data explaining the cause of missingness is unobserved by the researcher. Since missing data take different forms, researchers use a few different approaches to deal with these challenges. I summarize the types of missing data problems there are and their potential ramifications in Table 1 and the common tools used to solve these problems in Table 2.

Dealing with missingness

At the time of writing, King et al. (2001) estimated that 94% of political scientists use `LWD` to deal with missing data. In short, `LWD` does not seek to impute missing values. Instead, if the dependent

3. Sometimes called Non-ignorable (`NI`).

variable or any of the covariates in a regression model for a given observation are missing, the researcher does not include that observation in the analysis. Traditionally, scholars argue that `LWD` performs best (in terms of reducing the resulting bias in the researcher's subsequent regression models) when the data are `MCAR`.

If the data are `MAR` or `MNAR`, deleting observations with missing data introduces bias in one's regression estimates through a failure to account for correlation between the independent variable(s) and the error (King et al. 2001; Weisberg 2005; Schunk 2008; Azur et al. 2011). Furthermore, it has the potential to decrease statistical power. A meta-analysis of comparative and international political economy papers that use `LWD` demonstrates that political scientists have much, upwards of 50%, more Type I error - an incorrect rejection of the null hypothesis - than we would expect as a result of how we implement `LWD` (Lall 2016). To visualize this, we can draw a directed acyclic graph; Figure 1.

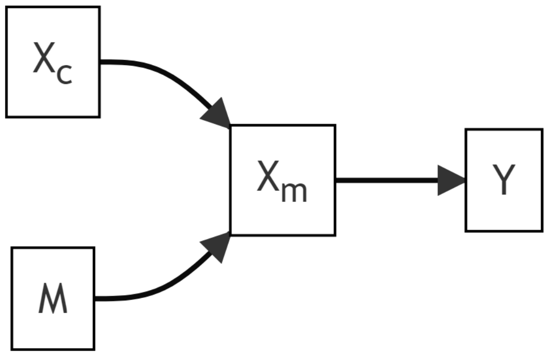

Figure 1: MAR data as confound

Others push against this claim and instead argue that `LWD` does not inherently generate bias for non-`MCAR` data but that researchers neglect to control for the cause of `MAR` or `MNAR` (Arel-

Bundock and Pelc 2018). This is still dependent on the researcher's grasp of theory and ability to identify the DGP leading to the missingness. Though the onus is on the researcher to do this, it is often a relatively high standard given the complexity to which social phenomena relate and the tendency for our datasets to be highly-dimensional. Furthermore, this also increases the number of parameters one must include in their statistical models - which runs the risk of increasing collinearity (Schrodt 2014).

Like `LWD`, simple imputation techniques like mean-and-median-based imputation do not reduce the chances of biased regression estimates. These approaches, called interpolation and extrapolation, are common for panel data and cross-sectional time series data. If you have missing data for an observation in one panel, you can take the same observations' responses in a previous panel and a latter panel. You then take the mean or the median of that particular observation for that variable. Other approaches seek to reduce this `MAR`-based bias through conditioning on other variables.

Hot Deck approaches to imputing missingness are regression-based in that they define the dependent variable as the one the researcher is attempting to impute and use variables thought to predict the cause of missingness in `MAR` contexts (Schunk 2008). In this approach, you often use a few variables to condition on. In many cases, the precise mechanism generating missingness is often tricky to triangulate. As a result, if you fail to provide the correct model specification when the data are `MAR`, you often end up with biased regression estimates.

`MI` seeks to solve this issue by using the entire dataset for imputing missing values (Rubin 1996). This approach uses the other variables in the dataset to generate a joint posterior distribution of all possible missing values for that particular observation. Many assume that most social science data sets are sufficiently large enough to condition on the mechanism generating `MAR`

(Schunk 2008). Unlike the other approaches, MI also generates uncertainty around the imputed values - via its construction of the joint posterior distribution (Rubin 1996) - which enables the researcher to be more transparent about the validity of those imputed values and to include that uncertainty in the researcher's subsequent statistical analyses (King et al. 2001; Hippel 2015). A prevalent implementation of MI in political science is the AMELIA II software (Honaker, King, and Blackwell 2011b; Honaker, King, and Blackwell 2011a; Lall 2016). This useful tool provides a computationally fast and simple process for imputation. Compared to the other approaches to missing data, AMELIA II performs quite well (Honaker, King, and Blackwell 2011a; Kropko et al. 2014). MI, however, often requires a set of distributional assumptions for the joint distribution - often the multivariate normal (Honaker, King, and Blackwell 2011a). Another challenge with this tool is that it runs up with the curse of dimensionality – if you are asking for more information by using more variables than you have observations, many non-regularized models will provide inaccurate estimates.

There is a variant to MI that seeks more computational efficiency and loosens some of the distributional assumptions required. This variant is called Multiple Imputation through Chained Equations (MICE). MICE performs quite well for large imputation tasks. MI struggles to impute values when there is missingness in the other variables of the dataset (Kropko et al. 2014) - which is quite common. Though not reliant on a multivariate normal distribution, Conditional MI still relies on general linear models (GLM) in calculating the values. MICE tries to get around this limitation in a few steps, as described by Azur et al. (2011). Figure 2 provides a visual representation of the procedure for a form of MICE used in this manuscript. I include more details about random forest models in the following subsection.

First, MICE performs a simple imputation, or interpolation, for every missing value in the entire

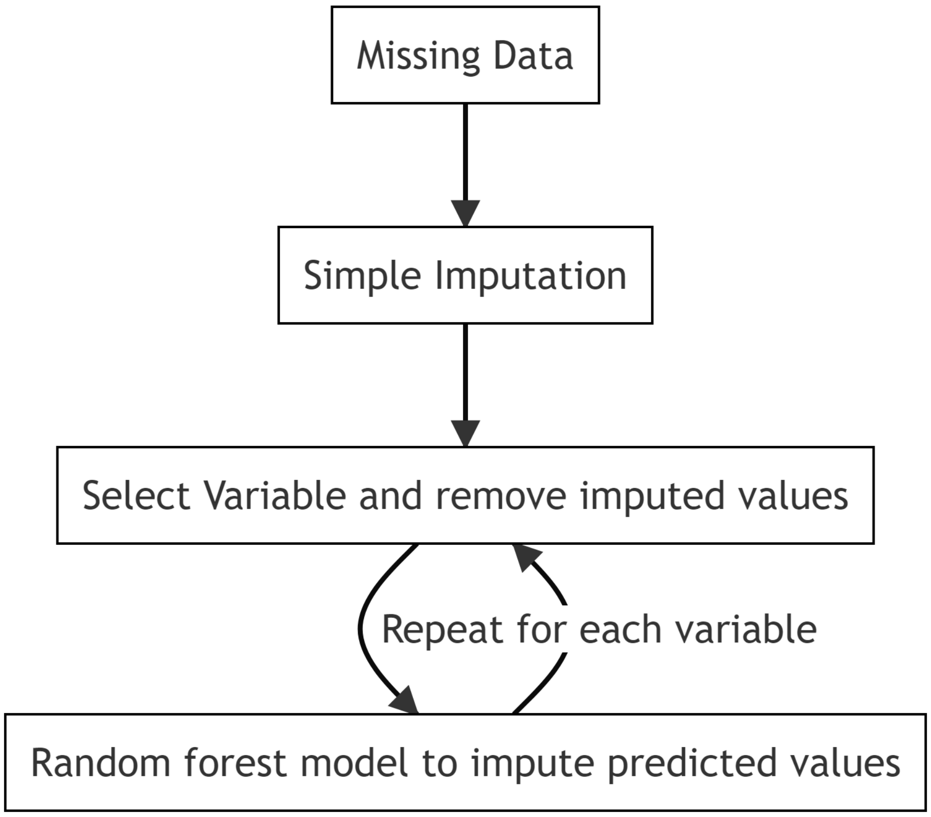
Missing Data
Simple Imputation
Select Variable and remove imputed values
Repeat for each variable
Random forest model to impute predicted values

Figure 2: Steps of RF-MICE procedure

dataset. These are the placeholder values. The second step in the general MICE paradigm involves identifying one variable to impute. Once complete, it then removes those placeholder values. The third step then involves regressing the observed values of the variable on the other variables in the model and replacing the predicted values generated from the regression model for the missing values. The fourth step is to repeat steps two and three for each variable in the data set with missing values - this constitutes a single iteration. As a fifth step, you perform between five and ten iterations[4].

The advantage of this chained equation procedure is to estimate each variable as an outcome with its own regression model that is most appropriate for it. The regression models that one may use in MICE are as numerous as those a researcher may choose from when engaged in statistical analysis. This means that the assumptions and the performance of the model one uses for the imputation are the same as in standard statistical analyses. Though it decreases some of the requirements for modeling the MAR process, it is not entirely atheoretical. We can, however, reduce the dependence that the imputed data have on a researcher's ability to theorize about the MAR process by selecting models that are accustomed to dealing with a large number of parameters without increasing inefficiency.

One valuable model for allowing one to include a large number of parameters without losses to efficiency, is a form of ensemble machine learning model called Random Forests. As the following discussion highlights, random forest models are optimal for engaging in predictive tasks (see Montgomery and Olivella 2018), which appears appropriate for the task of predicting missing values. These models have the additional benefit of not requiring a multivariate normal distribution, not

4. Though, the exact number of recommended iterations used in MICE are still up for debate (see Buuren and Groothuis-Oudshoorn 2011a; Azur et al. 2011). The recommendation is that you elect to go with more iterations if not constrained by computational limitations.

requiring one to specify a potentially incorrect model of which variables are included in the `MAR` process, nor is it in the form of a `GLM`. That is, these models reduce some of the dependence of an imputation task on the researcher's beliefs about the source of the missingness. As researchers apply these random forest models within the `MICE` framework, they also benefit from the advantages that `MI` provide over the hot deck framework. Furthermore, machine learning models like ensemble procedures apply regularization to deal with highly dimensional datasets. These three features suggest that this procedure offers much more flexibility to the researcher.

Table 1: Types of missing data processes, problems, and solutions

Type	Cause	Problem
MCAR	Missing data patterns are stochastic	Does not cause bias in estimates
MAR	Missing data patterns are not stochastic; however, once accounting for observed causes of missingness, any remaining missing data are as-if random.	Generates a type of omitted variable bias without accounting for it.
MNAR	Missing data patterns are not stochastic; caused by unobserved sources.	Generates a type of omitted variable bias; hard to correct for as the source is unobserved.

Table 2: Common solutions for missing data

Procedure	Assumptions	What it does	Fixes
LWD	Sources of missingness are random.	Removes rows that have a missing value for *any* variable in the statistical model	MCAR
Simple	Sources of missingness may be explained by some type of autocorrelation (spatial, temporal, non-independence of observations).	Takes the average or median value of observations that occur either before (if temporal data), proximate to (if spatial data), or similar to (if in the same sample), and fills that value into all rows with missingness for that variable.	Data with autocorrelation or are not i.i.d.
Hotdeck	Regression-based. So assumptions made are dependent on the particular regression you choose for this procedure.	Fits a regression model (specified by the researcher) where the researcher regresses the variable with missing values on other columns in the dataset. Fills empty rows with their predicted values from the regression model. Assumes that the regression model is properly specified.	MAR

Procedure	Assumptions	What it does	Fixes
MI	Assumes a normally distributed joint posterior distribution. Assumes that a GLM is appropriate for describing the non-stochastic sources of missingness.	Fits a Bayesian Linear Regression. Iteratively regresses each variable containing missing values onto all other variables in the dataset. Completes this process multiple times to account for uncertainty in the particular construction of the posterior distribution.	MAR

Procedure	Assumptions	What it does	Fixes
RFMICE	MICE, in general, is constrained by the assumptions of the particular estimator one chooses. RFMICE, however, is designed to loosen a number of assumptions. Still in the Frequentist framework so it does not produce posterior distributions to reflect uncertainty. Uncertainty is reflected by variation within and between iterations.	First performs simple imputation on each variable that has missingness. Then fits a random forest model where it attempts to predict each variable containing missingness using all other variables present in the dataset. Within each iteration, Random Forest models perform their own iterative procedures to maximize predictive capacity using cross-validation. It performs this for each variable and produces some user specified number of datasets which are the result of those maximally predicted values for each missing value.	MAR

The utility of random forest models for imputing political science data

Random forest models are concerned with calculating a fixed out-of-sample prediction under the supervised machine learning framework. They are popular among data scientists and researchers primarily interested in providing predictions instead of engaging in causal inference. In non-imputation applications of supervised machine learning models - a broader category of machine learning models that includes random forests, these models take partial, observed information about an outcome and estimate the relationship between the units with observed outcomes and several other observed features for those units. Then for the units without an observed outcome, we generalize that relationship to predict an outcome for them.

To get an intuition of the fundamental goal of a supervised machine learning model, I will provide an analogy. If we are looking to sort beans into a good or bad pile before we toss them into a pot, we often want to collect information about them. Features like color, size, and plumpness can all be good indicators of whether a bean will taste good or bad. Say we have over 5,000 beans, and we are a chef at a restaurant approaching the dinner hour, and we do not have time to sort all these beans. To save time, we look at these features like color, size, and plumpness and make two piles - good or bad for a subset of our 5,000 beans; say, in this instance, about 25%. We then get one of our employees to sort the rest of the beans for us. This employee may know less than us about what features matter more and how to identify a bean that will taste good or bad. Nevertheless, they can look at the two piles we have already made and try to pick up on patterns that make good and bad beans different from one another. With this information, the employee can "learn" these patterns so that even without the same knowledge as the chef, they can still make predictions about whether a bean will taste good or bad.

We have some units with recorded observations for a particular variable for imputation. Leveraging this, we can treat these documented observations as information so that we can "train" our computer to find a relationship between the observed information in that variable and information from other variables for that unit. We can then generalize this relationship to predict what that value would be for a missing unit. This seems like a reasonable approach to thinking about imputing missing values. We are not making naive imputations by taking the mean value. As we are using an expanded form of `MI`, we can also use all variables in the dataset to clarify that pattern. As a `MICE` procedure, we can specify a machine learning model, and we do this iteratively so that if we have more than one variable that we want to impute, we are not limited to accurate imputed values only for the units that are complete except for that particular variable to be imputed. I will elaborate more on how this works in a few paragraphs. Before we go there, however, I want to take the time to provide a few more details about how random forest models work, as they do not represent the whole of supervised machine learning.

Practitioners refer to random forests as tree-based ensemble models. As a supervised machine learning model, they start with the basic intuition described above; but in the process of "learning" or "training," these models follow a few distinct steps. Decision tree models split regions of the predictive space. For example, say we want to predict vote choice by one's partisanship. We would split the predictive space into regions. These regions could be different degrees of partisanship, like strong Republicans and weak Republicans. When we split this predictive space into regions, we essentially are subsetting our dataset of partisans into these different areas of the predictive space and trying to optimize a model to provide the best predictions in each region in our predictive space. That is, we try to find a model that maximizes the predictive performance of vote choice for strong Republicans, weak Republicans, independents, and so on. We examine performance by comparing how well we are predicting the unobserved outcomes relative to the observed outcomes within those predictive regions. We can "bag" our trees. When we "bag" the decision trees, we bootstrap the training sample and build a tree for each bootstrapped sample and average across them. Doing this allows for a decrease in the variance of the predictions coming from the model, which helps with reducing the chances of generating a trained model that will fail to adequately generalize to our data set not included in the training step.

As this is a primer for the applied researcher, I note that this discussion simplifies decision trees and random forest models. For those interested in more details about these concepts, James et al. (2013) provide a helpful discussion of these concepts. The main point is that random forest models specialize in generating predictions by optimizing at the *value* and *not* the *variable* level. This characteristic suggests that these models have the potential to be powerful tools for many different applications.

We can discuss their applicability to imputation in the `MICE` framework with a foundation in how random forest models work. As random forest models specialize in generating fixed out-of-sample predictions, these models have a joint goal with multiple imputation in that it should not be evaluated on the model's correctness but on the model's ability to predict a fixed (true) value (Rubin 1996). Here, one might think of missing data as the out-of-sample predictions intended to be estimated. With random forest models, you train the model on a training data set (often randomly generated through cross-validation), a randomly selected portion of your data that you train the model on, and then fit the model on the testing set, the remaining data not used for the training stage of your model (Hastie, Tibshirani, and Friedman 2009).

OLS models often perform relatively poorly on making out-of-sample predictions as they are

BLUE, assuming the data on hand are relatively representative of the population. As we have MAR data, this assumption likely fails, and any out-of-sample predictions are likely to be biased due to overfitting. Further, OLS may also generate out-of-bounds predictions for non-continuous data (Long 1997; Gelman, Hill, and Vehtari 2021) which is particularly troublesome in settings of prediction.

Other models provide within-bounds predictions, such as logistic models; however, as generalized linear models, they still assume a linear functional form and often produce biased interpretations of the likelihood function when presented with unobserved systematic processes (Mood 2010). Though OLS and Logistic regression underlie a lot of machine learning as tools, many consider them to have limited applicability to complicated settings requiring prediction. Random forest models provide within-bounds predictions, and they are fully non-parametric (Hastie, Tibshirani, and Friedman 2009), meaning they do not assume a functional form and consequently a joint distribution. This means that we have more flexibility in terms of what variables we have in our datasets that we need to impute. As social scientists, we rarely have datasets that contain DGPs that fall neatly within the optimal realm for GLMs. This added flexibility by using MICE and random forests makes the researcher's job easier.

On other performance metrics, random forest models, as an ensemble method, provide much more accurate predictions than single-tree alternatives in machine learning, such as CART (Montgomery and Olivella 2018). As discussed above, rather than generating a single estimate from a single model, ensemble models, like random forests, calculate multiple models and learn from their performance; this is the purpose of using the bootstrapped samples.

In the context of using MICE, I argue that political scientists should *consider* using random forest models to make accurate predictions with fewer assumptions and be more lenient in terms of conditioning on the cause of MAR in the data set. Recall that within each MICE iteration, one performs a model predicting (imputing) a missing value based on the other variables in the dataset. Explicitly, this means you are running a model per variable with missing data.

Given that random forests are non-parametric, within each MICE iteration, the relationship between the variable to be imputed and those used to make the predictions can be non-linear and can take many different multivariate distributional forms. This is a significant advancement on traditional MI, which assumes a multivariate normal distribution. Additionally, this is an advancement on other MICE models that political scientists use, which may not inherently conceptualize missing data as these unobserved values to predict from a generalized relationship of the observed variable with the other variables in the dataset. These relationships are also not assumed to be linear. Furthermore, using RF-MICE has the advantage over hot-deck procedures for those unsure about a variable's precise DGP for MAR.

In the next section, I illustrate the use of random forest models for political scientists by demonstrating a simulated application of a random forest implementation of MICE. The following section also compares this implementation's performance to other common approaches to handling missing data in political science in terms of our ability to reduce unobserved bias in our data and in computational costs.

Application of MICE with random forests

Using the numpy library (Harris et al. 2020), I simulate a population where $N = 1000000$. The population has 5 variables with the DGPs presented in Equation 1.

$$\begin{aligned} a_i &= Gamma(2,2) \\ b_i &= Binomial(1,0.6) \\ x_i &= 0.2 \times a_i + 0.5 \times b_i + Normal(0,1) \\ z_i &= 0.9 \times a_i \times b_i + Normal(0,1) \\ y_i &= 0.6 \times x_i + 0.9 \times z_i + Normal(0,1) \end{aligned} \tag{1}$$

I then use the `polars` library (team 2023) to generate 1000 random samples from the population with $n = 100$ for each sample. To introduce missingness into my data, I use the `miceRanger` library (Wilson 2021) to "ampute" 40% of the data for each of these samples with a `MAR` process. As one of the advantages of the `RF-MICE` procedure is to impute data generated from more complicated `MAR` processes, I ampute the data by constructing a logistic regression for each variable. Where the predicted value equals one, the corresponding observation is counted as missing (Wilson 2021). For the reader, I use the Plotly library (Inc. 2015) to produce Figure 9; which presents the distributions of data for the `X`, `Z`, and `Y` variables for my samples *before* amputation as well as the distribution of data for the `X`, `Z`, and `Y` variables for my samples *after* amputation. I would like to note, there are other ways I can perform the amputation of the data. The particular DGP of the missing data here is relatively simple relative to what we may have in real-world situations. However, given that `RF-MICE` is expected to perform better in situations with more complicated missing data patterns, the results of the simulated analyses may be an understatement of the differences in the reduction of bias between these different tools.

With these amputed datasets, I then apply some of the procedures I have discussed to impute these values. Interpolation is a quite simple procedure where I can fill in missing values by using the mean value of that particular variable for the non-missing observations. I perform

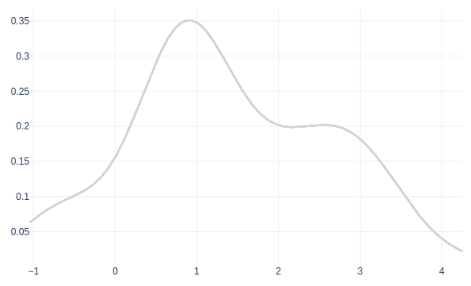

Figure 3: X

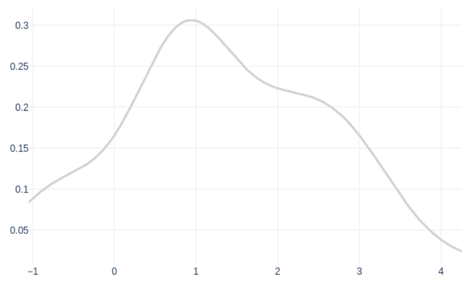

Figure 4: X - amputed

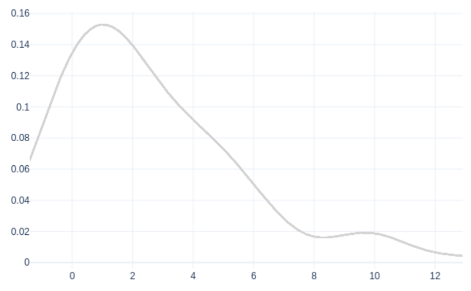

Figure 5: Z

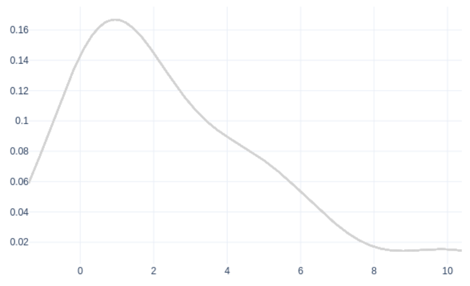

Figure 6: Z - amputed

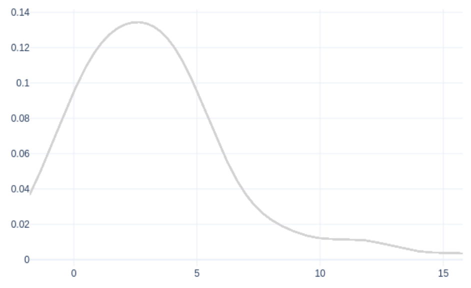

Figure 7: Y

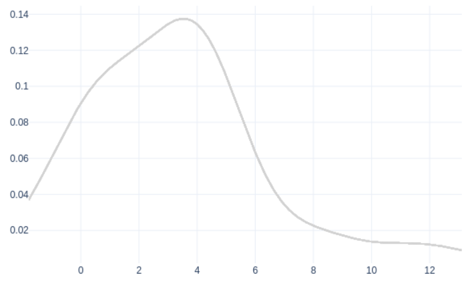

Figure 8: Y - amputed

Figure 9: Distributions of simulated data

this interpolation with the `mice` package (van Buuren and Groothuis-Oudshoorn 2011b) and iterate over it to provide 10 datasets. I also use the `AMELIA II` package (Honaker, King, and Blackwell 2011a) to perform standard `MI` and also store 10 datasets from the iterations. I use a standard Bayesian linear model in the `MICE` framework with the `mice` package (van Buuren and Groothuis-Oudshoorn 2011b). Bayesian linear models with uniform distributions or a weak prior distribution are similar to the familiar Ordinary Least Squares (Gelman, Hill, and Vehtari 2021). As discussed before, the final procedure I use is a random forest in the `MICE` framework. I perform the `RF-MICE` procedure using the `mice` package (van Buuren and Groothuis-Oudshoorn 2011b) and an alternative package `miceRanger` (Wilson 2021). I can use the codeblock below to perform the imputation with each of the tools I use on any given dataset I would use in a realistic research setting.

```
#| label: imputation-code-block-example

#| echo: true

#| eval: false



# Install libraries

install.packages(

    c(

        "AMELIA"

        , "mice"

        , "miceRanger"

    )

)
```

```r
# Load libraries

library(Amelia) # for MI

library(mice) # for many MICE and interpolation procedures

library(miceRanger) # for RF-Mice procedure

# Dataset



df



# Listwise Deletion

dfImputed <- df[complete.cases(df), ] # exclude rows that have missing values in any column



# Interpolation

dfImputed <- mice(

    df # dataframe

    , m = 10 # number of imputations

    , method = "mean" # mean interpolation

)



# Amelia

dfImputed <- amelia(

    df # dataframe

    , m = 10 # number of imputations

)
```

```
# Linear Bayesian MICE
dfImputed <- mice(
    df # dataframe
    , m = 10 # number of imputations
    , method = "linear" # Bayesian linear MICE
)

# RF-MICE with mice package
dfImputed <- mice(
    df # dataframe
    , m = 10 # number of imputations
    , method = "rf" # RF-MICE
)

# RF-MICE with miceRanger package
dfImputed <- miceRanger(
    df # dataframe
    , m = 10 # number of imputations
)
```

When producing the imputed datasets, I use the `tictoc` package to record the amount of time each procedure takes to complete the task on the 1000 samples as a measure of computational cost.[5] As a number of factors may affect the absolute computational costs for these procedures (e.g., hardware, whether other applications or software are running, whether one uses parallelization, etcetera), I am primarily going to focus on the relative computational costs of each procedure.

Each imputation procedure produced $m = 10$ datasets per simulated dataset, $s = 1000$. I have a total of $s \times m$ datasets. For each s dataset, I took the difference the values for each m dataset from the values in the complete dataset and took the average of these differences to give me a mean score of the discrepancy for each s dataset. Using the `GGPlot2` package (Wickham 2016) I produce Figure 10, Figure 12, and Figure 11, which represents these mean discrepancy scores for the three variables that were originally imputed.

Overall, we see that the procedures yield somewhat small differences between the actual and estimated values. For Y, `AMELIA II` has a mean discrepancy of -0.309, the Linear `MICE` procedure has a mean discrepancy of -0.198, the interpolation procedure has a mean discrepancy of 0.039, and `RF-MICE` with the `mice` package has a mean discrepancy of -0.188 and a mean discrepancy of -0.233. For X, we see that the `RF-MICE` with the `mice` package has a mean discrepancy of -0.030 and -0.008 with the `miceRanger` package, interpolation has a mean discrepancy of -0.074, Linear `MICE` has a mean discrepancy of -0.038, and `AMELIA II` has a mean discrepancy of -0.060. For Z, we also see that `RF-MICE` with the `mice` package has a mean discrepancy of -0.103 and -0.017 with the `miceRanger` package, interpolation has a mean discrepancy of -0.110, Linear `MICE` has a mean discrepancy of -0.106 and `AMELIA II` has a mean discrepancy of -0.199.

Though it is not a novel claim, I argue that in situations where we have missingness due to a `MAR` pattern, our regression models suffer from bias due to the systematic process generating

5. It is important to note that these benchmarks are based on a computer 32 GB of RAM, on a Intel i7-9700K processor at 3.60GHz with 8 cores using a NVIDIA GEForce RTX 2070 graphics card.

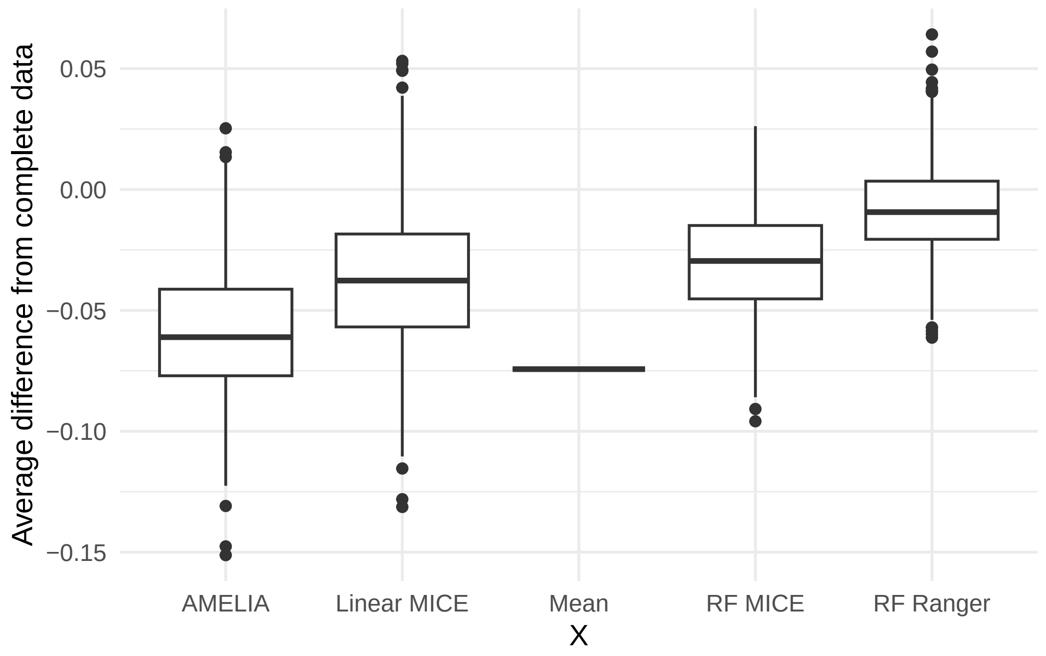

Figure 10: Discrepancies - X

that missingness. What I have argued so far is that we should consider using `RF-MICE` as it is, relative to other `MI` tools, a flexible tool that may be able to model a number of `MAR` processes which would help with reducing bias in our regression models.

To examine this claim, I take the amputed and imputed datasets (10) and use Rubin's rule (Rubin 1996) to pool across the regression models performed on each amputed and imputed sample. In total, I use the `rstan` package (Stan Development Team 2023) to fit $m ×s $ – 10 × 1000 regressions where I pool across my average posterior draws from `m` to produce a single set of statistics for each `s`.[6] I calculate the discrepancy of the pooled average posterior draws for `X` and `Z` – the variables directly contributing to the DGP of `Y` – from their parameters. I

6. As the model is not complicated and I am familiar with `Y`'s `DGP`, I fit the model using one chain with 1000 iterations.

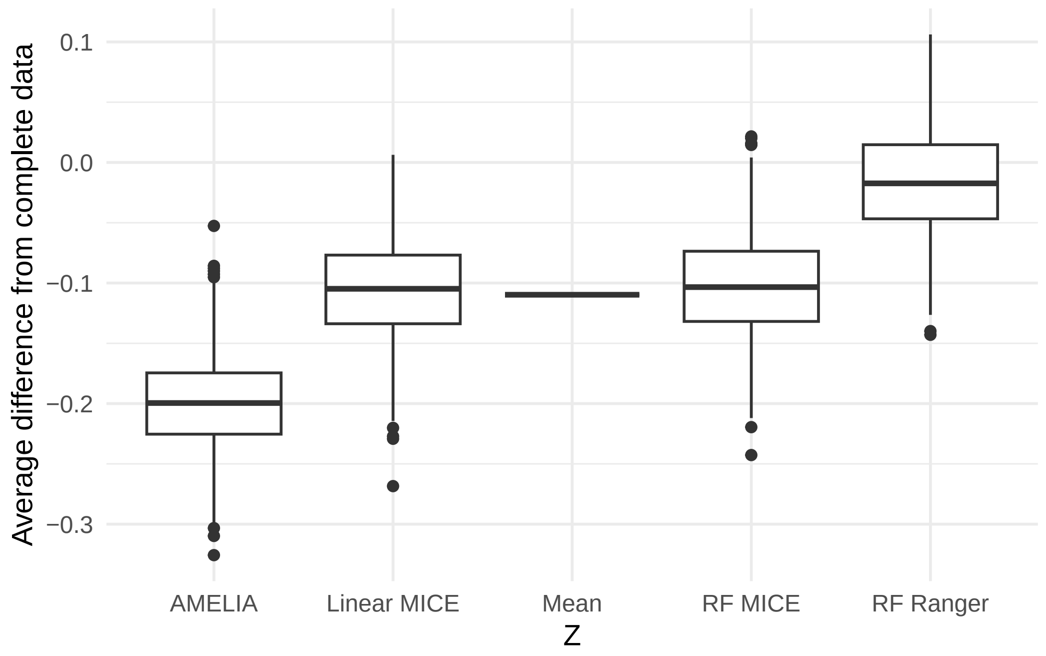

Figure 11: Discrepancies - Z

then present boxplots, produced by `GGPlot2` (Wickham 2016), that demonstrates the differences in levels of bias between `LWD`, `AMELIA II`, Linear `MICE`, and `RF-MICE` across my samples. The specification for the regression is depicted in Equation 2.

$$\begin{aligned} y_i \; Normal(\mu_i, \sigma) \\ \mu_i = \alpha + \beta_1(x_i - \bar{x}) + \beta_2(z_i - \bar{z}) \\ \alpha = Uniform(-\infty, \infty) \\ \beta_1 = Uniform(-\infty, \infty) \\ \beta_2 = Uniform(-\infty, \infty) \\ \sigma = Uniform(0, \infty) \end{aligned} \tag{2}$$

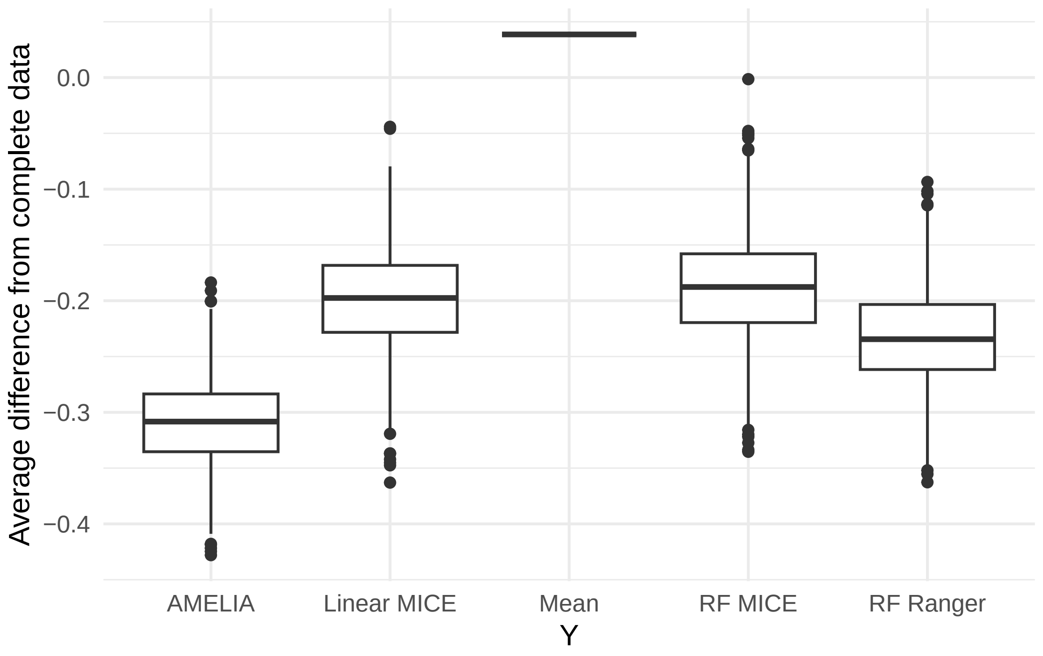

Figure 12: Discrepancies - Z

For readers not familiar with this notation, Equation 2 essentially describes the familiar OLS regression. I fit a model where y_i is explained by my likelihood function, μ_i and I assume that my parameters in my model can take any real number value. That is, I am not imposing a constraint with my defined priors about how large or small my estimates of β_1, β_2, and α can be. σ depicts the variance, which I constrain to be positive.

In the codeblock below, I provide an example of how to take the output `dfImputation` from the example codeblock above and pool the regression results across the imputed data generated from each tool. I demonstrate how to do this with the `lm` function which fits a Ordinary Least Squares regression and with the `brms_multiple` function which fits a Bayesian equivalent to the Ordinary Least Squares regression when using the default uniform priors.

```
#| label: codeblock-example-of-regression

#| echo: true

#| eval: false



# Install packages

install.packages("brms") # for bayesian regression models

install.packages("miceadds") # for converting Amelia outputs



# Load libraries

library(brms)

library(miceadds)



# Listwise deletion

    #* With the lm command

regression <- lm(

    formula = Y ~ X + Z # Y is the dependent variable with X and Z as predictors

    , data = dfImputed # use the imputed data

)

    #* With brms

regression <- brms_multiple(

    formula = Y ~ X + Z # Y is the dependent variable with X and Z as predictors

    , data = dfImputed # use the imputed data

)
```

```
# Interpolation

    #* with the lm command

regression <- pool(

    with(

        data = dfImputed # use the imputed data

        , exp = lm(Y ~ X + Z) # Y is the dependent variable with X and Z as predictors

    )

) # pool across each of the imputed datasets

    #* With brms

regression <- brms_multiple(

    formula = Y ~ X + Z # Y is the dependent variable with X and Z as predictors

    , data = dfImputed # use the imputed data

)


# Amelia

    #* with the lm command

regression <- pool(

    with(

        data = datlist2mids(dfImputed$imputations) # use the imputed data

        , exp = lm(Y ~ X + Z) # Y is the dependent variable with X and Z as predictors

    )

) # pool across each of the imputed datasets
```

```
    #* With brms
regression <- brms_multiple(
    formula = Y ~ X + Z # Y is the dependent variable with X and Z as predictors
    , data = datlist2mids(dfImputed$imputations) # use the imputed data
)


# Linear bayesian mice
    #* with the lm command
regression <- pool(
    with(
        data = dfImputed # use the imputed data
        , exp = lm(Y ~ X + Z) # Y is the dependent variable with X and Z as predictors
    )
) # pool across each of the imputed datasets
    #* With brms
regression <- brms_multiple(
    formula = Y ~ X + Z # Y is the dependent variable with X and Z as predictors
    , data = dfImputed # use the imputed data
)


# RF-MICE with mice package
    #* with the lm command
regression <- pool(
    with(
        data = dfImputed # use the imputed data
        , exp = lm(Y ~ X + Z) # Y is the dependent variable with X and Z as predictors
    )
) # pool across each of the imputed datasets
    #* With brms
regression <- brms_multiple(
    formula = Y ~ X + Z # Y is the dependent variable with X and Z as predictors
    , data = dfImputed # use the imputed data
)

# RF-MICE with the miceRanger package
    #* with the lm command
regression <- pool(
    with(
        data = completeData(dfImputed) # use the imputed data
        , exp = lm(Y ~ X + Z) # Y is the dependent variable with X and Z as predictors
    )
) # pool across each of the imputed datasets
    #* With brms
regression <- brms_multiple(
    formula = Y ~ X + Z # Y is the dependent variable with X and Z as predictors
    , data = completeData(dfImputed) # use the imputed data
```

```
)
```

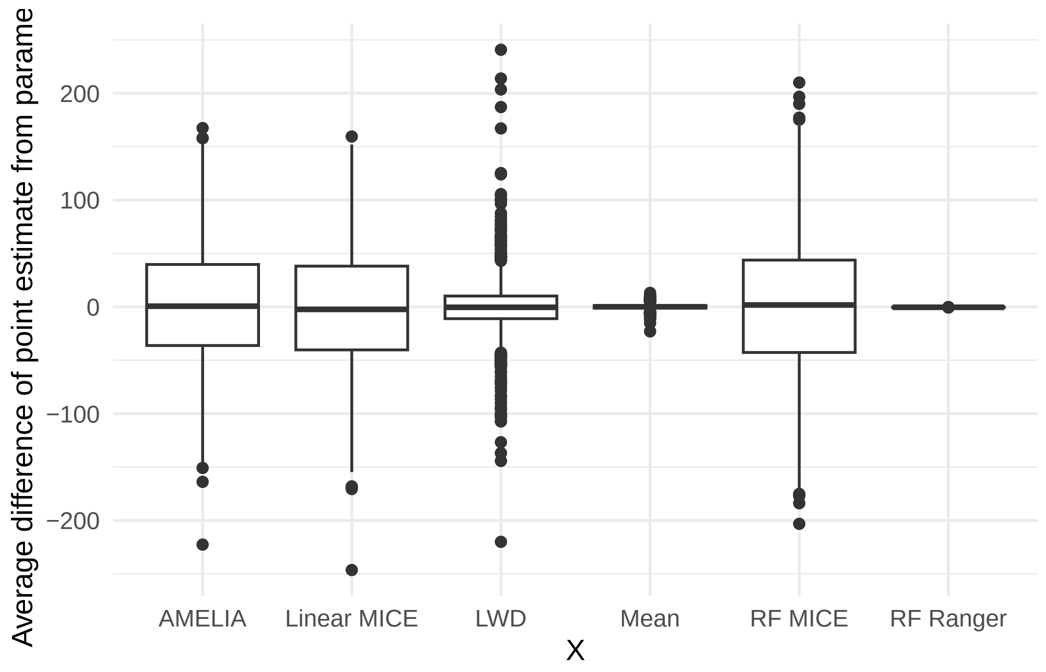

Figure 13: Average discrepancy scores - X

Figure 13 and Figure 14 present the distribution of differences between my average posterior draw and of my parameter value for the β coefficient for `X` and for `Z` respectively. First looking at bias in the β estimates for `X`, I see that the median discrepancy caused by `LWD` is quite small (close to zero). However, in looking at the spread, there are a number of models that produced biased estimates. It appears that the only two procedures that had a median of no bias was Linear `MICE` and `RF-MICE` with the `miceRanger` package (Wilson 2021). We see, that the spread of discrepancies is incredibly small when performing `RF-MICE` with the `miceRanger` package (Wilson 2021). `AMELIA II` (Honaker, King, and Blackwell 2011b) performs quite well in terms of median levels of discrepancies, but still produces biased estimates on average, with some examples of

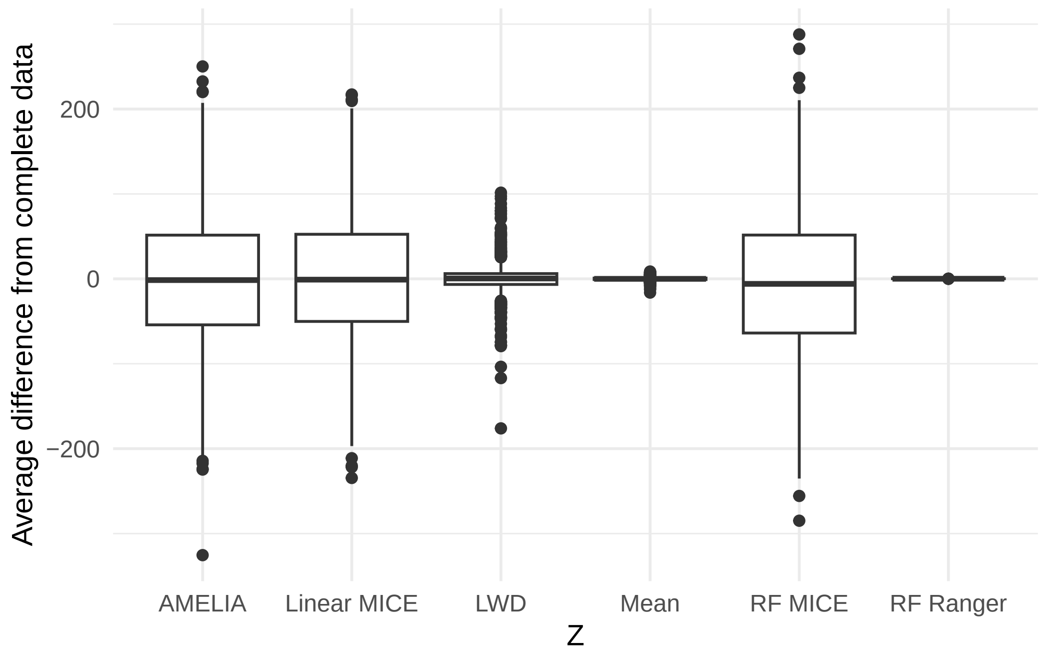

Figure 14: Average discrepancy scores - Z

extremely poor performance (in terms of bias). Turning to discrepancies between the β statistics and parameters for `Z`, we see a very similar story. `LWD` produces a distribution of discrepancies that is quite concerning while `RF-MICE` as implemented through the `miceRanger` package (Wilson 2021) does quite well at reducing bias caused by the `MAR` process. `AMELIA II` (Honaker, King, and Blackwell 2011b) also does quite well in reducing bias, on average, however can vary quite significantly in its ability to do so.

Turning to the computational performance of each procedure, interpolation with the `mice` package (van Buuren and Groothuis-Oudshoorn 2011b) took an average of 0.25 seconds; AMELIA II (Honaker, King, and Blackwell 2011b) took an average of 0.32 seconds; linear `mice` took an average of 0.30 seconds; `RF-MICE` with the `mice` package (van Buuren and Groothuis-Oudshoorn 2011b) took an average of 1.00 seconds; and `RF-MICE` with the `miceRanger` package (Wilson 2021) took an average of 2.68.

Conclusions

In theory, RF-MICE is a quite flexible tool that can operate in a number of circumstances to not only discover systematic processes leading to missingness in one's data, but to also use such information to recover the values. These expectations rely on the claim that Random Forest models are optimized for discovering patterns and to use that information to make out-of-sample predictions. With RF-MICE Random forests are coupled with MICE to produce significant improvements at retrieving the true values when a MAR process is present.

Using simulated data, I demonstrate two implementations of RF-MICE in R and compare it to implementations of more common procedures for dealing with MAR processes in political science. Overall, my simulated data and a number of measures of performance favor those theoretical expectations.

When comparing the distribution of imputed samples to the true samples, as if the MAR process was not present, both RF-MICE implementations do quite well in recovering the true values that were missing. When examining the discrepancy between what the imputed value is and the true value, the two RF-MICE implementations provide quite small discrepancies on average.

Measuring the performance of RF-MICE in terms of reducing bias in statistical estimation, I find that RF-MICE, when implemented with the miceRanger package (Wilson 2021), stands out as a valuable tool for researchers to use to reduce bias generated with MAR processes.

While in theory, it is nice for the applied researcher to hear about a new and powerful tool or procedure, they face many constraints. When examining how much time each implementation of a procedure took, I observed that both implementations of RF-MICE were not significantly worse in time it took to complete the procedure relative to other multiple imputation procedures.

Altogether, the simulated data suggest that `RF-MICE` is a valuable procedure for the applied researcher's toolbox. When dealing with missing data that one suspects may not be the result of a `MCAR` process, one should consider the flexibility that `RF-MICE` provides. It does not require the researcher to specify the variables involved in the `MAR` process, but also does not introduce significant computational costs nor introduce so much complexity as a procedure that it is impossible to anticipate or diagnose problems the procedure may introduce.

It is important to remind the reader that all tools have their limitations and that their value are quite dependent on the context for which they are to be applied. It is useful, however, to include tools that vary in the assumptions they make (Neumayer and Plümper 2017). The capabilities of `RF-MICE` are no different. While `RF-MICE` is quite flexible for addressing a number of `MAR` processes, it is important to note that it is not and should not be seen as a default or the sole tool for one to use when dealing with missing data. For example, simulations demonstrate that `RF-MICE` performs poorly when the missing data pattern arises from a moderating relationship (Marbach 2022). With the number of tools and their implementations being so available to the applied researcher, one should not shy away from using multiple implementations of these tools to ensure that one's substantive conclusions are not dependent on one tool or implementation.

References

Arel-Bundock, Vincent, and Krzysztof J. Pelc. 2018. "When Can Multiple Imputation Improve Regression Estimates?" *Political Analysis* 26, no. 2 (March): 240–245. https://doi.org/10.1017/pan.2017.43.

Azur, Melissa J., Elizabeth A. Stuart, Constantine Frangakis, and Philip J. Leaf. 2011. "Multiple imputation by chained equations: what is it and how does it work?" *International Journal of Methods in Psychiatric Research* 20, no. 1 (February): 40–49. https://doi.org/10.1002/mpr.329.

Buuren, Stef van, and Karin Groothuis-Oudshoorn. 2011a. “mice: Multivariate Imputation by Chained Equations in R.” *Journal of Statistical Software* 45 (9).

Foundation, The Python Software. 2023a. *itertools: Functions creating iterators for efficient looping.* https://docs.python.org/3/library/itertools.html?highlight=itertools#module-itertools.

———. 2023b. *sys: System-specific parameters and functions.* https://docs.python.org/3/library/sys.html?highlight=sys#module-sys.

Gelman, Andrew, Jennifer Hill, and Aki Vehtari. 2021. *Regression and Other Stories.* New York: Cambridge University Press.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. “Array programming with NumPy.” *Nature* 585 (7825): 357–362. https://doi.org/10.1038/s41586-020-2649-2. https://doi.org/10.1038/s41586-020-2649-2.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* 2nd. Springer Series in Statistics. New York: Springer.

Hippel, Paul T. von. 2015. “New Confidence Intervals and Bias Comparisons Show That Maximum Likelihood Can Beat Multiple Imputation in Small Samples.” *Structural Equation Modeling: A Multidisciplinary Journal* 23, no. 3 (October): 422–437. https://doi.org/10.1080/10705511.2015.1047931.

Honaker, J, G King, and M Blackwell. 2011a. “Amelia II: A program for missing data, R package version 1.5., 2012.” *Journal of Statistical Software* 45 (7): 1–3.

Honaker, James, Gary King, and Matthew Blackwell. 2011b. “Amelia II: A Program for Missing Data.” *Journal of Statistical Software* 45 (7): 1–47. https://doi.org/10.18637/jss.v045.i07.

Inc., Plotly Technologies. 2015. *Plotly.* Montreal, QC. https://plot.ly.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning: with Applications in R.* New York: Springer.

King, Gary, James Honaker, Anne Joseph, and Kenneth Scheve. 2001. “Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation.” *American Political Science Review* 95 (1).

Kropko, Jonathan, Ben Goodrich, Andrew Gelman, and Jennifer Hill. 2014. “Multiple imputation for continuous and categorical data: Comparing joint multivariate normal and conditional approaches.” *Political Analysis* 22 (4): 497–519. https://doi.org/10.1093/pan/mpu007.

Lall, Ranjit. 2016. "How Multiple Imputation Makes a Difference." *Political Analysis* 24 (4): 414–433. https://doi.org/10.1093/pan/mpw020.

Long, Scott J. 1997. *Regression Models for Categorical and Limited Dependent Variables.* Advanced Quantitative Techniques in the Social Sciences Series. Thousand Oaks, CA: Sage Publications.

Marbach, Moritz. 2022. "Choosing Imputation Models." *Political Analysis* 30 (4): 597–605.

Montgomery, Jacob M, and Santiago Olivella. 2018. "Tree-Based Models for Political Science Data." *American Journal of Political Science* 62 (3): 729–744. https://doi.org/10.1111/ajps.12361.

Mood, Carina. 2010. "Logistic regression: Why we cannot do what We think we can do, and what we can do about it." *European Sociological Review* 26 (1): 67–82. https://doi.org/10.1093/esr/jcp006.

Müller, Kirill, and Hadley Wickham. 2022. *tibble: Simple Data Frames.* R package version 3.1.8. https://CRAN.R-project.org/package=tibble.

Neumayer, Eric, and Thomas Plümper. 2017. *Robustness Tests for Quantitative Research.* New York: Cambridge University Press.

R Core Team. 2022. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

R Special Interest Group on Databases (R-SIG-DB), Hadley Wickham, and Kirill Müller. 2022. *DBI: R Database Interface.* R package version 1.1.3. https://CRAN.R-project.org/package=DBI.

Raasveldt, Mark, and Hannes Muehleisen. 2023. *DuckDB.* https://github.com/duckdb/duckdb.

Rubin, Donald B. 1996. "Multiple Imputation After 18+ Years." *Journal of the American Statistical Association* 91 (434): 473–489.

Schrodt, Philip A. 2014. "Seven deadly sins of contemporary quantitative political analysis." *Journal of Peace Research* 51 (2): 287–300. https://doi.org/10.1177/0022343313499597.

Schunk, Daniel. 2008. "A Markov chain Monte Carlo algorithm for multiple imputation in large surveys." *AStA* 92, no. 1 (January): 101–114. https://doi.org/10.1007/s10182-008-0053-6.

Stan Development Team. 2023. *RStan: the R interface to Stan.* R package version 2.21.8. https://mc-stan.org/.

team, polars core. 2023. *Polars: Blazingly fast DataFrames in Rust, Python & Node.js.* https://github.com/pola-rs/polars.

van Buuren, Stef, and Karin Groothuis-Oudshoorn. 2011b. "mice: Multivariate Imputation by Chained Equations in R." *Journal of Statistical Software* 45 (3): 1–67. https://doi.org/10.18637/jss.v045.i03.

Weisberg, Herbert F. 2005. *The Total Survey Error Approach: A Guide to the New Science of Survey Research.* Chicago, IL: The University of Chicago Press.

Wickham, Hadley. 2016. *ggplot2: Elegant Graphics for Data Analysis.* Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2022. *dplyr: A Grammar of Data Manipulation.* R package version 1.0.10. https://CRAN.R-project.org/package=dplyr.

Wilson, Sam. 2021. *miceRanger: Multiple Imputation by Chained Equations with Random Forests.* R package version 1.5.0. https://CRAN.R-project.org/package=miceRanger.