Universiteit Utrecht

Can a `Python` package do what `mice` can?

Elviss Dvinskis 2459302

Supervisors:
Hanne Oberman
Gerko Vink
Valentijn de Jong

ADS Master's Thesis

Department of Methodology and Statistics
Utrecht University
Netherlands

July 2022

Abstract

Missing data frequently complicate data analysis. Multiple imputation is a well known and robust technique for addressing missing data. In `R`, multiple imputation is commonly implemented through the `mice` package which utilizes the multiple imputation by chained equations (MICE) algorithm. However, such a standard choice is not yet established for `Python`. This study addresses four imputation methods that are implemented in `Python` to assess if they can yield unbiased and confidence valid estimates. A model-based simulation study is carried out to evaluate the performance of `KNNImputer`, `IterativeImputer`, `miceforest` and `MIDASpy`. The obtained results demonstrate that while under certain conditions `IterativeImputer` can show comparable performance to the conventional `R` imputation method `mice`, the other methods (`KNNImputer`, `miceforest` and `MIDASpy`) underperform under most conditions specified in this simulation study. This study suggests that it would be unwise to recommend these `Python` approaches as a general imputation strategy without a detailed comprehension of each of the method's proper application settings and fine-tuning.

Contents

1 Introduction

1.1 Motivation and context

Most real world datasets contain at least some missing values. This may be due to various circumstances and situations, with varying patterns and complexities, but missing values significantly affect the outcome of data analysis [1]. Missing data occurs across different fields and studies. For example, in the medical field it is not uncommon for a person to discontinue a clinical trial [2]. Another typical example is when people occasionally do not answer all the questions of a survey in which they participate [3].

Researchers frequently disregard the significance of missing data in their studies [1]. Historically, addressing missing data in research has received little attention. Common practices have relied on quick fixes or dismissing the problem as a whole, producing and presenting findings that might be misleading [1, 3]. Underestimating the importance of missing data leads to systematic biases and uncertainty of estimates. Moreover, it can lead to completely invalid conclusions even in otherwise well-conducted research [1, 4]. Therefore, proper treatment of missing data should be an essential part to any data related study.

1.2 Literature overview

Missingness mechanisms

Conceptualized more than four decades ago by Rubin, missing data can be categorized in three types of mechanisms in accordance to assumptions of why missing data occurs [1, 5].

When missing data occurs unrelated to the observed and unobserved data, the missingness mechanism is missing completely at random (MCAR). The propensity of data being missing is completely random [2]. An example of this would be, if participants of a survey would have either fully or incompletely responded to that survey, but some of the survey results got lost. In this example one could draw a random sample from the data and the missing data would not introduce bias, but it would have an overall reduced population sample. MCAR is frequently used as a reference mechanism in simulation studies, but is rarely applicable in practice [1, 6, 7].

When data is missing because of the observed data, but missingness is

not related to the unobserved data, the missingness mechanism is missing at random (MAR) [8]. One could think that the missingness pattern is MAR if, for example, in a study a given age group is less likely to answer certain questions, meaning that completion of the survey is dependent on the respondents age (and the age is always completely observed). MAR is the most frequently used assumption in practice [2, 9].

If missingness depends both on the observed and unobserved values, the missingness mechanism is missing not at random (MNAR). Usually the missing data is MNAR if both MCAR or MAR mechanisms are not met [4, 6]. MNAR analyses are problematic because missingness is related to factors unknown. The same example in the case of a survey here would be if a respondent fails to finish it for undisclosed reasons. The only way to address MNAR is to approach missingness with modelling [1, 4].

To obtain valid statistical conclusions, the possible missingness mechanisms should be accounted for and discerned which type of missing data one has in a given analysable dataset [1].

Handling missing data

Ideally, one would want to have a dataset with complete data, but the reality is that even within a carefully planned and executed real life study missing data often occurs [4]. One solution is to use data analysis techniques that are robust towards missing data, however that cannot always be implemented, or might not even be applicable [10].

There are several ad-hoc methods developed for dealing with missing data. The most common approach is just omitting cases with missing data and analyzing the complete cases. This method is therefore referred to as complete-case analysis or also listwise deletion [1]. Under MCAR, complete-case analysis may produce unbiased results. When data is MAR or MNAR the estimates will be biased and the results will have a high standard error [3]. This might be a reasonable approach to handle missing data if indeed MCAR is met and the sample size is large enough that statistical power is not an issue [4].

Another approach is available-case analysis, also known as pairwise deletion. This method removes information only when specific data points are required for testing and are missing. The other existing values in the dataset that have missing data are included in statistical analysis [3, 8]. This method quite understandably preserves more information than complete-case analy-

sis, and available-case analysis is unbiased under MCAR, but again is biased under MAR and MNAR. Applying pairwise deletion means that individual parameters will be estimated on other data that have different standard errors and sample size, leading to complicated further analysis, which ultimately defeats the purpose of its use [8].

Mean substitution (or mean imputation) is an easy way of handling missing data. It simply takes the mean of a variable for a missing value [1]. Regardless of its simplicity and popularity, it contributes no new information, disrupts relationships between variables whilst completely misrepresenting the variance and biasing every estimate except for the mean (and even the mean if data are MAR or MNAR). It should be avoided almost at all times [1, 8].

More sophisticated methods implement imputation by replacing missing data with estimated values based on the additional information available [1]. Regression imputation utilizes the available variables to estimate coefficients and fit predictions on the missing values [11]. It maintains more information than complete-case and available-case analysis and does not considerably modify the standard deviations [8]. In theory, it can produce unbiased estimates of the mean and regression weights under MAR [1]. Seemingly adequate imputations are in fact unrealistic because standard errors are underestimated, correlations are overestimated and sample size is artificially increased [3].

Stochastic regression imputation adds a noise term to the predictions. Noise is generated randomly from non-missing cases and is included in the estimates [8]. Compared to a simple regression imputation, it can also have unbiased correlations under MAR [3]. It is considered a better method than those described previously, but also has its disadvantages. It underestimates the standard error and can produce estimates outside a logical allowed range [1, 8]. These and other single imputation methods are generally not recommended, but the underlying idea of single imputation is an important concept for multiple imputation. Multiple imputation addresses all the limitations that single imputation has, while at the same time utilizing its strengths [3, 8].

Maximum likelihood methods (expectation-maximization method) and accounting for different scenarios of uncertainty (sensitivity analysis) have their use cases for missing data [3, 6, 12]. However, they will not be discussed further due to the scope of this study.

Multiple imputation

Rather than imputing a single value for each missing data point, multiple imputation repeats the process m times. The possible values are drawn from a distribution that is specific for each missing data value [1]. Each time a different imputed value is assigned to the missing value and a set of plausible values are formed [8]. The parameters are estimated for each of the m imputations as if one would be working with complete data and then the estimates are pooled across the m imputations to get the final estimate (see Figure 1) [1, 3]. In most situations, when multiple imputation is used, it is implemented under the MAR assumption, but can also be applied for MCAR and MNAR [4].

Multiple imputation separates and addresses the missing data problem before the analysis of scientific interest is conducted [13]. It accounts for uncertainty and solves estimation problems of variability in the missing values, resulting in valid statistical inference [1, 3]. Multiple imputation is considered a conventional method across many study fields and is becoming increasingly more popular (e.g. due to its availability in different programming languages) [4].

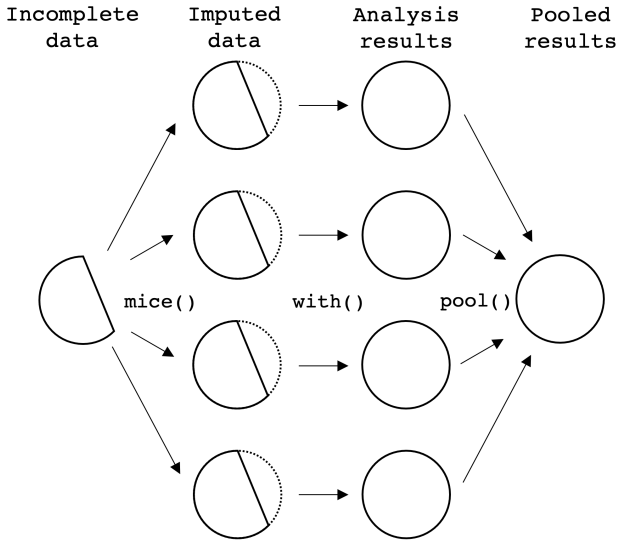


Figure 1: Multiple imputation procedure ($m = 4$). Adapted from Van Buuren (2018) [1].

Implementation of multiple imputation and approaches available in Python

The most widely recognized and practically implemented use of multiple imputation is through the `mice` package developed for use with `R`, employing the multiple imputation by chained equations (MICE) algorithm [14, 15]. The `mice` package handles the three essential stages of multiple imputation (imputing, analysing and then pooling data) with the functions `mice()`, `with()` and `pool()` (see Figure 1). It is considered an obvious and standard choice for handling missing data with `R` and can be used for a broad range of situations [14, 16].

However, such a default package has not yet been established for `Python` [17]. Since many data scientists prefer to use `Python` over `R` [18], it is of relevance to investigate whether the currently available options in `Python` can produce unbiased and confidence valid estimates, comparable to what `mice` can. There are several options available for handling missing data in `Python`, such as using `KNNImputer` from `sklearn.impute`, `IterativeImputer` from the `fancyimpute` library, `ImputationKernel` from `miceforest` and the package `MIDASpy` [19–23].

1.3 Research question

The goal of this study is to determine whether the `Python` approaches `KNNImputer`, `IterativeImputer`, `miceforest` and `MIDASpy` for handling missing data can produce valid inferences. The `R` package `mice` will serve as a benchmark method in a model-based simulation study.

2 Data and Methods

Simulation studies are an empirical way of evaluating statistical research and methodology. Since there is no single standardised way of performing and evaluating a simulation study, the approach in this model-based study follows the recommendations outlined in Morris et al. (2019) and Oberman and Vink (n.d.) [24, 25].

This simulation study consists of four sequential repeated main steps: 1) generating complete data; 2) inducing missingness (amputation procedure); 3) imputation procedures with different methods; 4) estimating regression

coefficients. The simulation is repeated a thousand times ($n_{\text{sim}} = 1000$). Performance is evaluated by bias, confidence interval width and coverage rate. The simulation setup is shown in Figure 2.

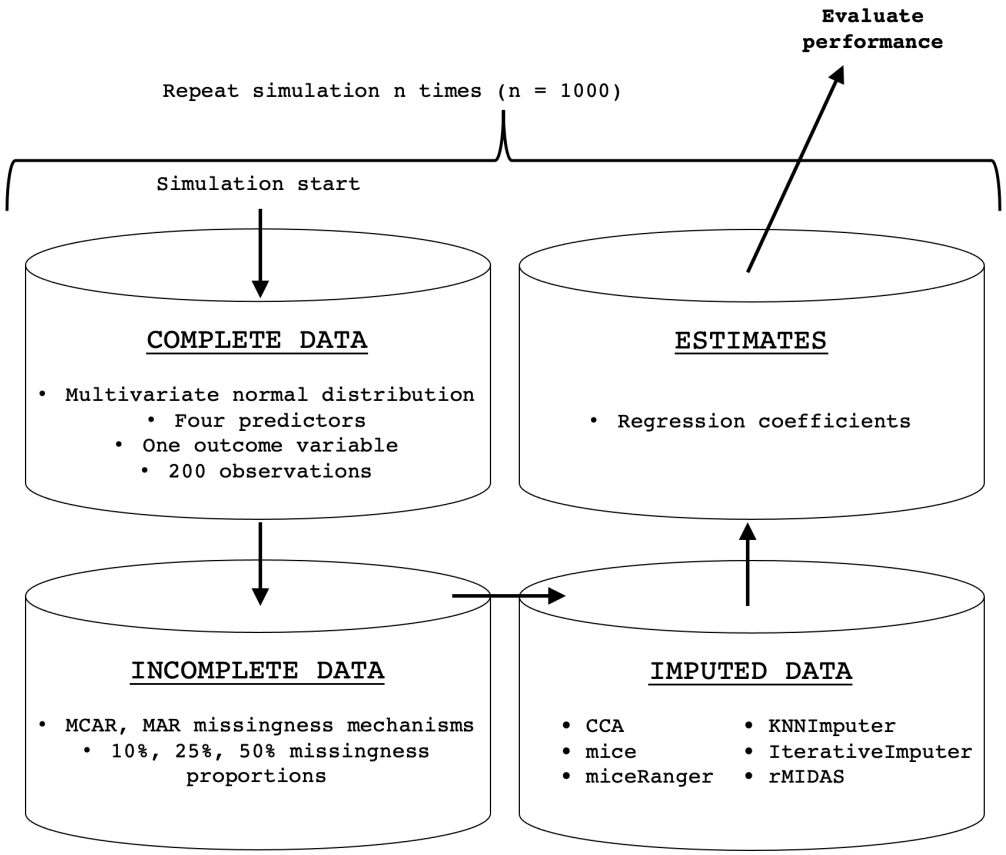


Figure 2: Simulation study design schema.

2.1 Data generation and amputation

Data were drawn from a multivariate normal distribution $\mathcal{N}(\mu, \Sigma)$ using `rmvnorm()` from package `mvtnorm` [26], with four predictor variables (X_1, X_2, X_3, X_4), one outcome variable (Y) and 200 observations per dataset ($n_{\text{obs}} = 200$). The predictor space can be notated

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} \sim \mathcal{N} \left[\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0.3 & 0.3 & 0.3 \\ 0.3 & 1 & 0.3 & 0.3 \\ 0.3 & 0.3 & 1 & 0.3 \\ 0.3 & 0.3 & 0.3 & 1 \end{pmatrix} \right]$$

and the outcome variable Y for each simulation run is

$$Y = -0.5X_1 - 0.1X_2 + 0.1X_3 + 0.5X_4 + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 1)$.

Missingness in the complete data was generated through the `ampute()` function from the `mice` package, introducing two types of missingness mechanisms (right tailed MAR and MCAR) and three missingness proportions (10%, 25% and 50%). MCAR is considered as a reference missingness mechanism because any imputation method should be able to produce valid inferences under MCAR, otherwise the method itself is not justifiable. MAR is considered as it is the most broadly and frequently assumed missingness mechanism in empirical studies [1, 25]. The three chosen missingness proportions impact the severity of the missing data problem, where more drastic differentiation between the methods should occur with increased missingness proportion. One could consider these three missingness proportions to be approximate to real life missingness percentage, ranging from realistic to moderate (but still realistic), to extreme [25].

2.2 Imputation methods

Six methods of handling missing data are implemented in this simulation study — complete-case analysis (CCA), `mice`, `miceRanger`, `KNNImputer`, `IterativeImputer` and `rMIDAS`. CCA and `mice` are used as benchmarks for this study.

Any other method should outperform CCA under the MAR assumption and have at least equal performance under the MCAR assumption [1, 8]. Furthermore, CCA is chosen to see if the data were indeed generated and

amputed correctly, knowing that the method will be unbiased under MCAR and is not expected to produce sound results under MAR. If any of the other methods produce worse results than CCA, it either can be an indication that something was wrong within the simulation study setup, or that a given imputation method has a fundamental problem. Imputations from `mice` are expected to yield valid inferences, and it is an ubiquitous method for multiple imputation in `R` [1]. If any of the `Python` methods would show an equal (or better) performance than `mice`, that would suggest it being a feasible `Python` alternative to `mice`.

Imputation with `mice`

For `mice` imputation, default settings were used - the number of imputations to perform (m) and the number of iterations ($maxit$) were set to five. For other imputation methods to be comparable to `mice`, where applicable, these settings were adjusted accordingly.

Imputation with `miceRanger`

The `miceforest` package in `Python` utilizes the MICE algorithm using random forests. With speed and memory efficiency in mind, `miceforest` was developed as an alternative to the conventional `mice` package. Originally, `miceforest` was developed for `Python`, but it has also been translated to an `R` package called `miceRanger` [22]. Essentially, both packages are the same, and in this simulation study `miceRanger` was used due to ease of syntax. Imputations were performed using the `miceRanger()` function (in `Python` this would be equivalent to using `ImputationKernel`) with m and $maxit$ set to five.

Imputation with `KNNImputer` and with `IterativeImputer`

Both `KNNImputer` and `IterativeImputer` were implemented through the package `reticulate()` [27], which allows `Python` syntax usage in `R`.

`KNNImputer` fills in the missing values using the k-nearest neighbours approach. It maps the non-missing values in an n-dimensional coordinate space and groups them, then computes the closest points to a missing value. The mean of those closest points is the imputed value [28]. Within this simulation study the default settings for `KNNImputer` were used ($n\_neighbors$

= 5), and the data were imputed with the `KNNImputer.fit_transform()` function.

`IterativeImputer` uses each missing value as a feature in a function to regress on the other features in a dataset, then it replaces the missing values with the obtained predictions. It was also inspired by the MICE algorithm, but generally is used to return a single imputation. Multiple imputations can be achieved when `sample_posterior` is set to `True` [19, 20]. For this study imputations from `IterativeImputer` were obtained with `IterativeImputer.fit_transform()`, where `max_iter` was changed from the default 10 to five, and `sample_posterior` was set to `True`. The imputation process was repeated five times ($m = 5$).

Imputation with `rMIDAS`

Lastly, multiple imputation with denoising autoencoders (MIDAS) was employed, which is a deep learning approach that uses unsupervised neural networks for multiple imputation. The idea behind this approach is to use the denoising autoencoders to corrupt and reconstruct data. Missing values are treated as a corrupt data subset and imputations are generated from a trained model used for reconstruction [23]. It can be implemented through the `MIDASpy` class for `Python` and is also partially translated to `R (rMIDAS)`. In `R` the `rMIDAS` package still uses the underlying `Python` functions through `reticulate()`. For syntax simplicity `rMIDAS` was used in this study. The `rMIDAS train()` function was left at its default tuning as described in the package documentation and Lall et al. (2022) [23]. As in the other simulation methods, m was set to five.

2.3 Estimation and performance evaluation

For each method of handling missing data, a linear regression model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \epsilon$$

was used to obtain regression estimates. For CCA the regression model was fit on the list-wise deleted data and estimates were obtained for performance evaluation. Estimates for `KNNImputer` were acquired from a single imputation, and then used for performance evaluation. For `mice`, `miceRanger`, `IterativeImputer` and `rMIDAS` the linear regression model was fit on each

imputation using the `with()` function and the estimates were pooled by applying Rubin's rules [29] with the `pool()` function (for `rMIDAS` the `with()` and `pool()` functions are performed with its own package function `combine()`). The pooled estimates were then used for performance evaluation.

Performance of the different methods was evaluated by bias, coverage rate and confidence interval width, as supported by recommendations from literature [1, 24, 25]. Since the true parameters are known, the raw bias is calculated as the estimates divergence from the actual truth, and it should be as close to zero as possible. The confidence interval width is assessed as the difference between the upper and lower 95% confidence interval bounds. Coverage for an estimate is quantified as the proportion of confidence intervals that contain the true estimand. Instead of the typical cut-off at 0.95, in this simulation study a coverage rate from 0.94 to 0.96 is approximate for an estimate being confidence-valid, in accordance with the Markov chain Monte Carlo standard error of the metric [24].

2.4 Ethical and legal considerations

Since the data used in this study is simulated and only open-source software was used, no ethical or legal considerations arise. The complete code for the simulation study is available at `github.com/edvinskis/python_mice` .

3 Results

The results section covers selected simulation results. Full simulation results can be found in the repository `github.com/edvinskis/python_mice` . The average raw bias, coverage rate and confidence interval width for each predictor variable under each missingness mechanism and missingness proportion are summarized in Appendix A. Plots for bias, coverage rate and confidence interval width for each predictor variable under each missingness mechanism and missingness proportion can be found in appendices B through D. Note that method abbreviations for figures are used throughout for readability - `IterativeImputer`, `KNNImputer`, `miceRanger` and `rMIDAS` are referred to as ITIMP, KNN, MICER and MIDAS, respectively.

3.1 Selected analysis results

Performance of the studied methods for handling missing data varies across predictor variables, missingness mechanisms and missingness proportions. To highlight the differences in performance between the different methods, the predictor variable X_1 under both MAR and MCAR missingness mechanisms with a 50% missingness proportion is examined in detail.

Figure 3 shows the bias in the estimated regression coefficients across simulation repetitions of the studied methods for handling missing data. One could consider that `IterativeImputer`, `mice` and `rMIDAS` are unbiased under both MAR and MCAR, but `miceRanger` and `KNNImputer` display bias. As expected, CCA is biased under the MAR assumption, but is unbiased under the MCAR missingness mechanism.

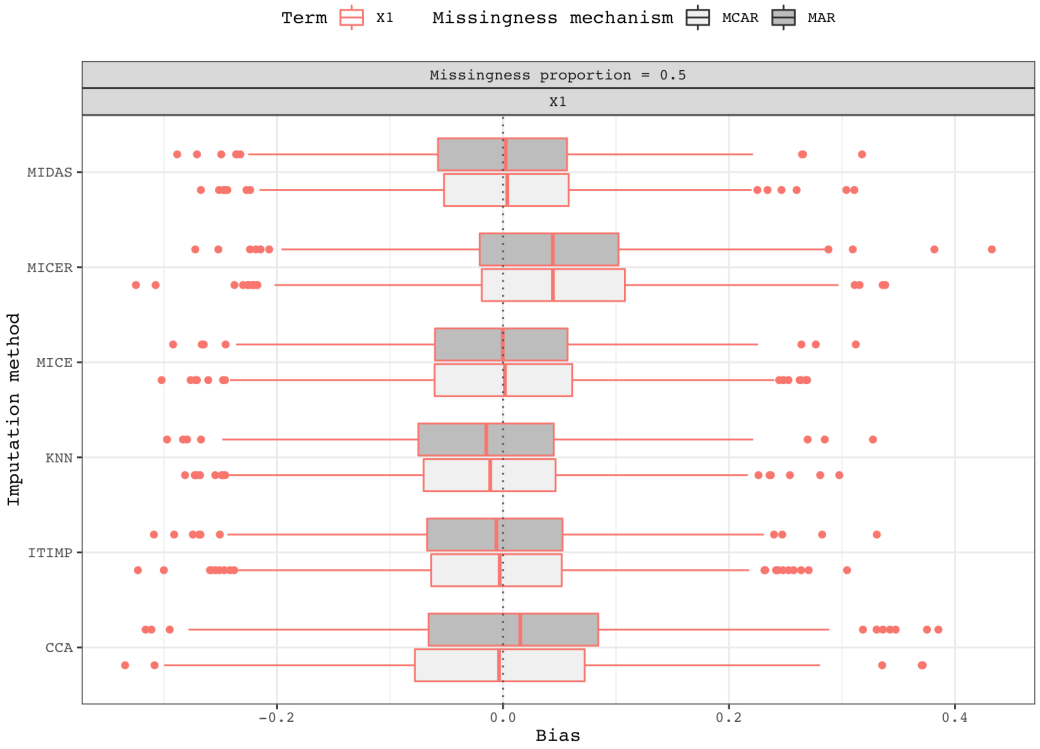


Figure 3: Bias in the estimated effect of X_1 for different imputation methods under MAR and MCAR missingness mechanisms with a missingness proportion of 50%.

The coverage rate is shown in Figure 4. One can observe that `rMIDAS` presents under-coverage, and more so, `KNNImputer` and `miceRanger` exhibit strong under-coverage for both missingness mechanisms. Slight under-coverage can also be observed for `IterativeImputer` under the MCAR missingness mechanism. As anticipated, CCA for MCAR, and `mice` under both MAR and MCAR are confidence valid.

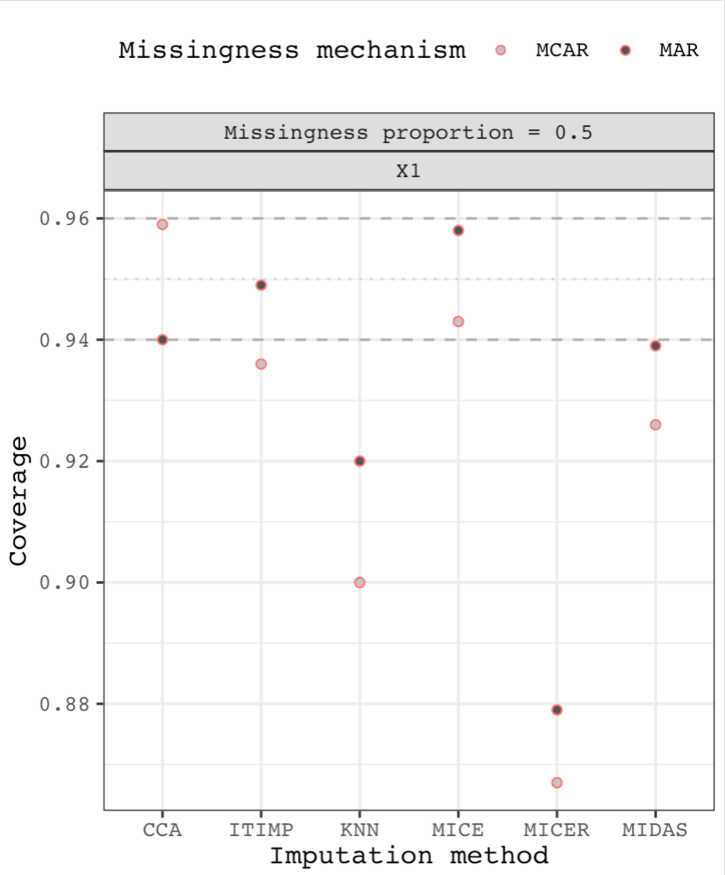


Figure 4: Coverage rate in the estimated effect of X_1 for different imputation methods under MAR and MCAR missingness mechanisms with a missingness proportion of 50%.

The confidence interval width for the different methods can be seen in Figure 5. `CCA`, `IterativeImputer` and `mice` have relatively wide confidence interval widths, but they are narrow for `KNNImputer`, `miceRanger` and `rMIDAS`.

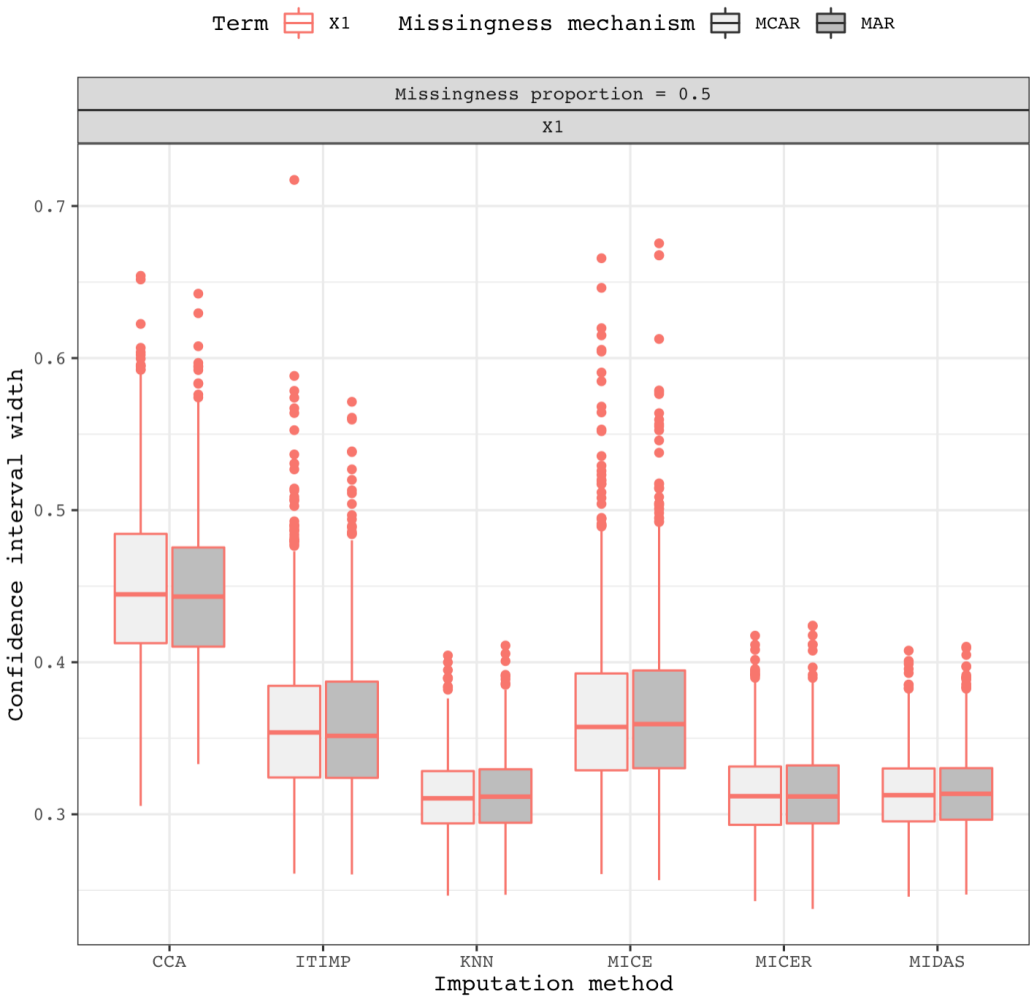


Figure 5: Confidence interval width in the estimated effect of X_1 for different imputation methods under MAR and MCAR missingness mechanisms with a missingness proportion of 50%.

Together, Figures 3-5 reveal that for the predictor variable X_1, both `miceRanger` and `KNNImputer` are biased, they present with considerable under-coverage and their confidence interval widths are narrow. Although `rMIDAS` is unbiased on average, it is not confidence valid. Consequently, Figure 5 shows that it has a very narrow confidence interval width. Lastly, `mice` is unbiased, has proper coverage rate and a relatively large confidence interval width.

When examining the other predictor variables in the same manner (under 50% missingness), similar findings occur with subtle differences and will not be discussed further (see Appendix B-D). However, it should be noted that for `IterativeImputer`, predictor variables X_2 and X_3 are confidence valid under both MAR and MCAR (Appendix C).

Generally, the possibility for handling missing data should improve with decreasing missingness proportion for all methods (Appendix B-D), however `KNNImputer`, `miceRanger` and `rMIDAS` show an overall poor performance across different missingness proportions and missingness mechanisms for almost every predictor variable (Appendix B-D). These methods are not confidence valid for any predictor variable under both missingness mechanisms with a missingness proportion of 25% (Appendix C). Furthermore, even when the missingness proportion is 10%, confidence validity varies considerably between the predictor variables and missingness mechanisms (Appendix C).

4 Conclusion and Discussion

The most significant findings of this study can be summarized as follows:

1. `mice` generally performs well under different missingness mechanisms and proportions, and CCA produces valid results under the MCAR missingness mechanism.

2. `IterativeImputer` varies in performance across missingness mechanisms, missingness proportions and for different predictor variables, but can yield valid inferences under certain conditions.

3. `KNNImputer`, `miceRanger` and `rMIDAS` reveal poor performance for almost every defined condition in this simulation study.

Biased estimates will cause a method to under-perform and not produce sound results. Having unsuitable confidence intervals is attributed to ei-

ther under-coverage or over-coverage, thus resulting in invalid inferences [24]. Therefore, a reason why `KNNImputer`, `miceRanger` and `rMIDAS` were not confidence valid can be explained by them having too narrow confidence interval widths. On the other hand, confidence interval widths should not be too large as it leads to more uncertainty. Narrower confidence interval widths that still have a proper coverage result in better inferences [25]. In that regard, for those conditions where `IterativeImputer` was unbiased and confidence valid, the confidence interval widths were similar to `mice`.

4.1 Limitations

The poor performance of the imputation methods in `Python` may be explained by how they were implemented in this simulation study, and each of these methods may have been affected by different limitations.

Although `KNNImputer` is a multivariate single imputation method, its performance could be improved by selecting a more appropriate number of neighbours as it is the most important hyperparameter for `KNNImputer`. In this simulation study the default of five was used. By testing and cross-validating a different number of neighbours, the imputation method could provide more accurate estimates [30]. Moreover, the choice of the distance metric for `KNNImputer` can notably impact the imputation results, especially with increasing missingness proportion. In this simulation study Euclidean distance was used, and this choice could be evaluated as well [31].

The performance of `IterativeImputer` differed across missingness mechanisms, missingness proportions and predictor variables, but was unbiased and confidence valid under most scenarios in this simulation study. Possibly, better performance could be achieved by setting its number of iterations per imputation to the default ten. It was changed to five in order to be more directly comparable to `mice`. It is possible that due to this change, `IterativeImputer` could not fully converge and therefore provided such performance [32].

Non-convergence should not have been a problem for `miceRanger` as it essentially uses the same MICE algorithm, and in comparison `mice` did provide valid results with the same number of iterations per imputation. The dire performance of `miceRanger` could be explained by not controlling trees and their growth in the random forest part of the method [33].

The overall poor performance of `rMIDAS` can be attributed to the fact that the model was trained in a way that it could not generalize on unseen data.

Adjusting the layer structure and nodes of the network, together with the dropout rate, could make it less prone to overfitting. Another possibility for improvement would be changing the default activation function (by default it is an exponential linear unit). Additionally, the authors of this package acknowledge and warn about the possibility of poor performance for smaller datasets, due to the learning nature of neural networks [23].

Nevertheless, fine-tuning every method to each specific dataset is a task of its own. Obviously, better results can be obtained when a method is fine-tuned to each specific scenario, but in this simulation study, comparisons were made of the method's ability being as close to the default settings that of `mice`.

Furthermore, additional limitations of this simulation study exist. First, the performance of a certain method on real world data may reveal a completely different performance. Empirical data rarely hold a theoretical distribution and the simulation results therefore might not be attributable, therefore distributional characteristics should also be studied [24, 25]. Second, only MAR and MCAR missingness mechanisms were considered and MNAR was not included. Lastly, the performance was evaluated in terms of bias, coverage rate and confidence interval width. Additionally, convergence for multiple imputation methods and the fit of the imputation models should be taken into consideration [25].

4.2 Research question answer

The goal of this study was to determine if the investigated `Python` approaches could produce valid inferences. Neither `KNNImputer`, `miceforest` (`miceRanger`) or `MIDASpy` (`rMIDAS`) yielded valid inferences. The performance of `IterativeImputer` varied, but it could provide valid results under most of the studied conditions.

4.3 Implications

These findings entail certain considerations for real world use. If the missingness mechanism is indeed MCAR, it is justifiable to apply a simple method such as CCA for handling missing data. Under other missingness mechanisms the method will fail [1]. However, if the data are not MCAR then `mice` is a fine choice for handling missing data. Considering the perfomance of `KNNImputer`, `miceRanger` and `rMIDAS` in this simulation study, it

is hard to recommend them for general use without a detailed understanding about their fine-tuning, and further evaluation should be considered. More importantly, further research is required for `IterativeImputer` as it produced sound results under most conditions in this simulation study and therefore, with certain limitations, it might be a viable alternative to `mice`.

References

[1] Stef Van Buuren. *Flexible imputation of missing data*. CRC press, 2018.

[2] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[3] Daniel A Newman. Missing data: Five practical guidelines. *Organizational Research Methods*, 17(4):372–411, 2014.

[4] Alma B Pedersen, Ellen M Mikkelsen, Deirdre Cronin-Fenton, Nickolaj R Kristensen, Tra My Pham, Lars Pedersen, and Irene Petersen. Missing data and multiple imputation in clinical epidemiological research. *Clinical epidemiology*, 9:157, 2017.

[5] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[6] John W Graham et al. Missing data analysis: Making it work in the real world. *Annual review of psychology*, 60(1):549–576, 2009.

[7] Paul Madley-Dowd, Rachael Hughes, Kate Tilling, and Jon Heron. The proportion of missing data should not be used to guide decisions on multiple imputation. *Journal of Clinical Epidemiology*, 110:63–73, 2019.

[8] John W Graham. *Missing data: Analysis and design*. Springer Science & Business Media, 2012.

[9] Yiran Dong and Chao-Ying Joanne Peng. Principled missing data methods for researchers. *SpringerPlus*, 2(1):1–17, 2013.

[10] Adam Kapelner and Justin Bleich. Prediction with missing data via bayesian additive regression trees. *Canadian Journal of Statistics*, 43 (2):224–239, 2015.

[11] Zhongheng Zhang. Missing data imputation: focusing on single imputation. *Annals of translational medicine*, 4(1), 2016.

[12] James R Carpenter and Melanie Smuk. Missing data: A statistical framework for practice. *Biometrical Journal*, 63(5):915–947, 2021.

[13] Gerko Vink and Stef van Buuren. Pooling multiple imputations when the sample happens to be the population. *arXiv preprint arXiv:1409.8542*, 2014.

[14] Stef Van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45:1–67, 2011.

[15] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL `https://www.R-project.org/`.

[16] Melissa J Azur, Elizabeth A Stuart, Constantine Frangakis, and Philip J Leaf. Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research*, 20 (1):40–49, 2011.

[17] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[18] Joel Grus. *Data science from scratch: first principles with python*. O'Reilly Media, 2019.

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12: 2825–2830, 2011.

[20] Alex Rubinsteyn and Sergey Feldman. *fancyimpute: An Imputation Library for Python*, 2016. URL `https://github.com/iskandr/fancyimpute`.

[21] Samuel Wilson. *miceforest: Fast, Memory Efficient Imputation with lightgbm*, 2020. URL `https://github.com/AnotherSamWilson/miceforest`.

[22] Samuel Wilson. *miceRanger: Multiple Imputation by Chained Equations with Random Forests*, 2020. URL `https://github.com/FarrellDay/miceRanger`.

[23] Ranjit Lall and Thomas Robinson. The midas touch: Accurate and scalable missing-data imputation with deep learning. *Political Analysis*, 30(2):179–196, 2022.

[24] Tim P Morris, Ian R White, and Michael J Crowther. Using simulation studies to evaluate statistical methods. *Statistics in medicine*, 38(11): 2074–2102, 2019.

[25] Hanne Oberman and Gerko Vink. Towards a standardized evaluation of imputation methodology. URL `https://www.gerkovink.com/evaluation/`. Date accessed 11-05-2022.

[26] Alan Genz, Frank Bretz, Tetsuhisa Miwa, Xuefei Mi, Friedrich Leisch, Fabian Scheipl, and Torsten Hothorn. *mvtnorm: Multivariate Normal and t Distributions*, 2021. URL `https://CRAN.R-project.org/package=mvtnorm`.

[27] JJ Allaire, Kevin Ushey, Yuan Tang, and Dirk Eddelbuettel. *reticulate: R Interface to Python*, 2017. URL `https://github.com/rstudio/reticulate`.

[28] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6): 520–525, 2001.

[29] Donald B Rubin. *Multiple imputation for nonresponse in surveys*. John Wiley & Sons, 1987.

[30] Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1): 112–118, 2012.

[31] Miriam Seoane Santos, Pedro Henriques Abreu, Szymon Wilk, and João Santos. How distance metrics influence missing data imputation with k-nearest neighbours. *Pattern Recognition Letters*, 136:111–119, 2020.

[32] Xiaofeng Zhu, Shichao Zhang, Zhi Jin, Zili Zhang, and Zhuoming Xu. Missing value estimation for mixed-attribute data sets. *IEEE Transactions on Knowledge and Data Engineering*, 23(1):110–121, 2010.

[33] Anne-Laure Boulesteix, Silke Janitza, Jochen Kruppa, and Inke R König. Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6): 493–507, 2012.

Appendices

Appendix A

Table 1: Simulation study results (summarized across simulation runs).

Method	Missingness Mechanism	Missingness Proportion	Term	Bias	Coverage Rate	Confidence Interval Width
CCA	MAR	0.10	(Intercept)	2.89e-02	0.938	0.297
CCA	MAR	0.10	X1	1.97e-03	0.955	0.326
CCA	MAR	0.10	X2	2.38e-03	0.939	0.327
CCA	MAR	0.10	X3	9.43e-03	0.945	0.327
CCA	MAR	0.10	X4	1.24e-02	0.940	0.327
CCA	MAR	0.25	(Intercept)	6.83e-02	0.880	0.331
CCA	MAR	0.25	X1	4.44e-03	0.954	0.359
CCA	MAR	0.25	X2	8.25e-03	0.949	0.361
CCA	MAR	0.25	X3	1.55e-02	0.943	0.360
CCA	MAR	0.25	X4	2.24e-02	0.944	0.362
CCA	MAR	0.50	(Intercept)	1.35e-01	0.767	0.432
CCA	MAR	0.50	X1	1.22e-02	0.940	0.446
CCA	MAR	0.50	X2	9.31e-03	0.963	0.448
CCA	MAR	0.50	X3	2.28e-02	0.946	0.449
CCA	MAR	0.50	X4	2.87e-02	0.949	0.450
CCA	MCAR	0.10	(Intercept)	2.29e-03	0.954	0.297
CCA	MCAR	0.10	X1	-1.35e-03	0.949	0.327
CCA	MCAR	0.10	X2	-4.45e-03	0.943	0.327
CCA	MCAR	0.10	X3	1.52e-03	0.948	0.327
CCA	MCAR	0.10	X4	5.60e-03	0.944	0.326
CCA	MCAR	0.25	(Intercept)	5.94e-04	0.941	0.327
CCA	MCAR	0.25	X1	-6.60e-04	0.954	0.360
CCA	MCAR	0.25	X2	-3.56e-03	0.937	0.361
CCA	MCAR	0.25	X3	1.20e-03	0.949	0.360
CCA	MCAR	0.25	X4	5.20e-03	0.945	0.361
CCA	MCAR	0.50	(Intercept)	1.63e-03	0.953	0.406
CCA	MCAR	0.50	X1	-1.85e-03	0.959	0.450
CCA	MCAR	0.50	X2	-3.60e-03	0.943	0.450
CCA	MCAR	0.50	X3	-3.22e-03	0.959	0.448
CCA	MCAR	0.50	X4	9.06e-03	0.940	0.447

IterativeImputer	MAR	0.10	(Intercept)	2.08e-03	0.947	0.286
IterativeImputer	MAR	0.10	X1	-2.32e-03	0.953	0.318
IterativeImputer	MAR	0.10	X2	-4.49e-03	0.947	0.320
IterativeImputer	MAR	0.10	X3	2.31e-03	0.947	0.320
IterativeImputer	MAR	0.10	X4	5.88e-03	0.939	0.318
IterativeImputer	MAR	0.25	(Intercept)	2.45e-03	0.950	0.295
IterativeImputer	MAR	0.25	X1	-4.88e-03	0.944	0.332
IterativeImputer	MAR	0.25	X2	-3.42e-03	0.946	0.338
IterativeImputer	MAR	0.25	X3	1.76e-03	0.948	0.338
IterativeImputer	MAR	0.25	X4	7.49e-03	0.944	0.333
IterativeImputer	MAR	0.50	(Intercept)	1.24e-03	0.948	0.311
IterativeImputer	MAR	0.50	X1	-7.73e-03	0.949	0.359
IterativeImputer	MAR	0.50	X2	-7.50e-03	0.951	0.370
IterativeImputer	MAR	0.50	X3	4.79e-03	0.961	0.372
IterativeImputer	MAR	0.50	X4	1.09e-02	0.946	0.360
IterativeImputer	MCAR	0.10	(Intercept)	2.04e-03	0.950	0.287
IterativeImputer	MCAR	0.10	X1	-2.53e-03	0.949	0.318
IterativeImputer	MCAR	0.10	X2	-4.50e-03	0.946	0.320
IterativeImputer	MCAR	0.10	X3	1.93e-03	0.952	0.319
IterativeImputer	MCAR	0.10	X4	6.88e-03	0.944	0.318
IterativeImputer	MCAR	0.25	(Intercept)	2.77e-03	0.944	0.295
IterativeImputer	MCAR	0.25	X1	-3.86e-03	0.945	0.332
IterativeImputer	MCAR	0.25	X2	-4.91e-03	0.951	0.337
IterativeImputer	MCAR	0.25	X3	3.07e-03	0.945	0.336
IterativeImputer	MCAR	0.25	X4	7.48e-03	0.929	0.331
IterativeImputer	MCAR	0.50	(Intercept)	1.78e-03	0.938	0.309
IterativeImputer	MCAR	0.50	X1	-6.04e-03	0.936	0.361
IterativeImputer	MCAR	0.50	X2	-6.44e-03	0.943	0.370
IterativeImputer	MCAR	0.50	X3	1.65e-03	0.964	0.371
IterativeImputer	MCAR	0.50	X4	1.33e-02	0.938	0.360
KNNImputer	MAR	0.10	(Intercept)	4.60e-04	0.932	0.279
KNNImputer	MAR	0.10	X1	-1.36e-03	0.942	0.310
KNNImputer	MAR	0.10	X2	-4.39e-03	0.934	0.310
KNNImputer	MAR	0.10	X3	6.46e-04	0.935	0.310
KNNImputer	MAR	0.10	X4	3.83e-03	0.926	0.309
KNNImputer	MAR	0.25	(Intercept)	-3.19e-03	0.932	0.277
KNNImputer	MAR	0.25	X1	-5.31e-03	0.933	0.311
KNNImputer	MAR	0.25	X2	-3.93e-03	0.920	0.311
KNNImputer	MAR	0.25	X3	-8.89e-05	0.933	0.311
KNNImputer	MAR	0.25	X4	5.23e-03	0.918	0.310
KNNImputer	MAR	0.50	(Intercept)	-8.76e-03	0.913	0.273
KNNImputer	MAR	0.50	X1	-1.66e-02	0.920	0.313
KNNImputer	MAR	0.50	X2	-9.82e-03	0.912	0.314
KNNImputer	MAR	0.50	X3	5.55e-03	0.917	0.314
KNNImputer	MAR	0.50	X4	1.64e-02	0.895	0.313

KNNImputer	MCAR	0.10	(Intercept)	2.02e-03	0.947	0.279
KNNImputer	MCAR	0.10	X1	3.35e-04	0.942	0.310
KNNImputer	MCAR	0.10	X2	-3.48e-03	0.935	0.310
KNNImputer	MCAR	0.10	X3	7.87e-04	0.939	0.310
KNNImputer	MCAR	0.10	X4	4.23e-03	0.939	0.309
KNNImputer	MCAR	0.25	(Intercept)	3.72e-03	0.929	0.276
KNNImputer	MCAR	0.25	X1	-1.68e-03	0.925	0.310
KNNImputer	MCAR	0.25	X2	-4.82e-03	0.918	0.311
KNNImputer	MCAR	0.25	X3	2.50e-03	0.927	0.311
KNNImputer	MCAR	0.25	X4	4.94e-03	0.916	0.310
KNNImputer	MCAR	0.50	(Intercept)	1.28e-03	0.899	0.272
KNNImputer	MCAR	0.50	X1	-1.35e-02	0.900	0.312
KNNImputer	MCAR	0.50	X2	-5.91e-03	0.889	0.313
KNNImputer	MCAR	0.50	X3	4.27e-03	0.914	0.313
KNNImputer	MCAR	0.50	X4	1.73e-02	0.894	0.312
mice	MAR	0.10	(Intercept)	2.39e-03	0.948	0.287
mice	MAR	0.10	X1	-7.43e-04	0.954	0.319
mice	MAR	0.10	X2	-3.54e-03	0.937	0.321
mice	MAR	0.10	X3	1.46e-03	0.947	0.321
mice	MAR	0.10	X4	4.13e-03	0.936	0.318
mice	MAR	0.25	(Intercept)	2.50e-03	0.949	0.295
mice	MAR	0.25	X1	-1.50e-03	0.945	0.333
mice	MAR	0.25	X2	-2.68e-03	0.939	0.340
mice	MAR	0.25	X3	3.16e-04	0.948	0.338
mice	MAR	0.25	X4	4.54e-03	0.939	0.333
mice	MAR	0.50	(Intercept)	1.06e-03	0.947	0.315
mice	MAR	0.50	X1	-1.43e-03	0.958	0.368
mice	MAR	0.50	X2	-5.77e-03	0.954	0.381
mice	MAR	0.50	X3	2.66e-03	0.962	0.381
mice	MAR	0.50	X4	3.87e-03	0.945	0.366
mice	MCAR	0.10	(Intercept)	2.55e-03	0.949	0.287
mice	MCAR	0.10	X1	-1.70e-03	0.948	0.318
mice	MCAR	0.10	X2	-3.79e-03	0.946	0.320
mice	MCAR	0.10	X3	1.17e-03	0.950	0.320
mice	MCAR	0.10	X4	5.59e-03	0.942	0.318
mice	MCAR	0.25	(Intercept)	3.27e-03	0.950	0.295
mice	MCAR	0.25	X1	-7.52e-04	0.944	0.335
mice	MCAR	0.25	X2	-3.66e-03	0.937	0.339
mice	MCAR	0.25	X3	1.20e-03	0.949	0.338
mice	MCAR	0.25	X4	4.12e-03	0.935	0.334
mice	MCAR	0.50	(Intercept)	9.25e-04	0.947	0.313
mice	MCAR	0.50	X1	-2.18e-05	0.943	0.366
mice	MCAR	0.50	X2	-2.35e-03	0.941	0.378
mice	MCAR	0.50	X3	-2.29e-03	0.953	0.376
mice	MCAR	0.50	X4	6.92e-03	0.939	0.362

miceRanger	MAR	0.10	(Intercept)	2.48e-03	0.942	0.280
miceRanger	MAR	0.10	X1	6.03e-03	0.940	0.310
miceRanger	MAR	0.10	X2	-3.03e-03	0.928	0.311
miceRanger	MAR	0.10	X3	2.05e-04	0.937	0.311
miceRanger	MAR	0.10	X4	-2.62e-03	0.935	0.310
miceRanger	MAR	0.25	(Intercept)	2.27e-03	0.933	0.278
miceRanger	MAR	0.25	X1	1.68e-02	0.927	0.312
miceRanger	MAR	0.25	X2	1.07e-03	0.912	0.314
miceRanger	MAR	0.25	X3	-3.68e-03	0.922	0.313
miceRanger	MAR	0.25	X4	-1.40e-02	0.917	0.311
miceRanger	MAR	0.50	(Intercept)	2.86e-03	0.908	0.274
miceRanger	MAR	0.50	X1	4.02e-02	0.879	0.314
miceRanger	MAR	0.50	X2	2.46e-03	0.876	0.320
miceRanger	MAR	0.50	X3	-4.90e-03	0.879	0.319
miceRanger	MAR	0.50	X4	-3.62e-02	0.856	0.314
miceRanger	MCAR	0.10	(Intercept)	2.09e-03	0.944	0.280
miceRanger	MCAR	0.10	X1	5.92e-03	0.939	0.310
miceRanger	MCAR	0.10	X2	-2.35e-03	0.936	0.311
miceRanger	MCAR	0.10	X3	-1.17e-04	0.937	0.311
miceRanger	MCAR	0.10	X4	-1.87e-03	0.931	0.310
miceRanger	MCAR	0.25	(Intercept)	3.22e-03	0.928	0.277
miceRanger	MCAR	0.25	X1	1.93e-02	0.916	0.311
miceRanger	MCAR	0.25	X2	-1.83e-03	0.911	0.313
miceRanger	MCAR	0.25	X3	-1.49e-03	0.928	0.313
miceRanger	MCAR	0.25	X4	-1.52e-02	0.901	0.311
miceRanger	MCAR	0.50	(Intercept)	9.83e-04	0.906	0.273
miceRanger	MCAR	0.50	X1	4.23e-02	0.867	0.314
miceRanger	MCAR	0.50	X2	3.08e-03	0.863	0.319
miceRanger	MCAR	0.50	X3	-6.74e-03	0.887	0.318
miceRanger	MCAR	0.50	X4	-3.48e-02	0.875	0.312
rMIDAS	MAR	0.10	(Intercept)	4.76e-04	0.942	0.278
rMIDAS	MAR	0.10	X1	-1.63e-03	0.946	0.309
rMIDAS	MAR	0.10	X2	-5.14e-03	0.933	0.309
rMIDAS	MAR	0.10	X3	-1.98e-03	0.936	0.309
rMIDAS	MAR	0.10	X4	9.50e-03	0.923	0.308
rMIDAS	MAR	0.25	(Intercept)	-4.14e-04	0.946	0.275
rMIDAS	MAR	0.25	X1	-2.27e-03	0.941	0.311
rMIDAS	MAR	0.25	X2	-4.55e-03	0.928	0.311
rMIDAS	MAR	0.25	X3	-7.16e-03	0.928	0.311
rMIDAS	MAR	0.25	X4	1.56e-02	0.922	0.310
rMIDAS	MAR	0.50	(Intercept)	-3.21e-03	0.949	0.270
rMIDAS	MAR	0.50	X1	-5.59e-04	0.939	0.314
rMIDAS	MAR	0.50	X2	-1.00e-02	0.916	0.314
rMIDAS	MAR	0.50	X3	-1.51e-02	0.924	0.315
rMIDAS	MAR	0.50	X4	2.46e-02	0.907	0.313

rMIDAS	MCAR	0.10	(Intercept)	2.21e-03	0.948	0.278
rMIDAS	MCAR	0.10	X1	-9.96e-04	0.945	0.309
rMIDAS	MCAR	0.10	X2	-4.46e-03	0.939	0.309
rMIDAS	MCAR	0.10	X3	-1.72e-03	0.946	0.309
rMIDAS	MCAR	0.10	X4	8.84e-03	0.934	0.308
rMIDAS	MCAR	0.25	(Intercept)	3.32e-03	0.941	0.275
rMIDAS	MCAR	0.25	X1	6.11e-04	0.942	0.311
rMIDAS	MCAR	0.25	X2	-6.45e-03	0.932	0.310
rMIDAS	MCAR	0.25	X3	-5.85e-03	0.935	0.311
rMIDAS	MCAR	0.25	X4	1.30e-02	0.922	0.310
rMIDAS	MCAR	0.50	(Intercept)	1.36e-03	0.932	0.269
rMIDAS	MCAR	0.50	X1	1.47e-03	0.926	0.314
rMIDAS	MCAR	0.50	X2	-9.64e-03	0.900	0.314
rMIDAS	MCAR	0.50	X3	-1.66e-02	0.924	0.314
rMIDAS	MCAR	0.50	X4	2.49e-02	0.911	0.312

Appendix B

Figure 6: Bias in the estimated effects of X_1, X_2, X_3, X_4 for different imputation methods under MAR and MCAR missingness mechanisms with a missingness proportion of 10%.

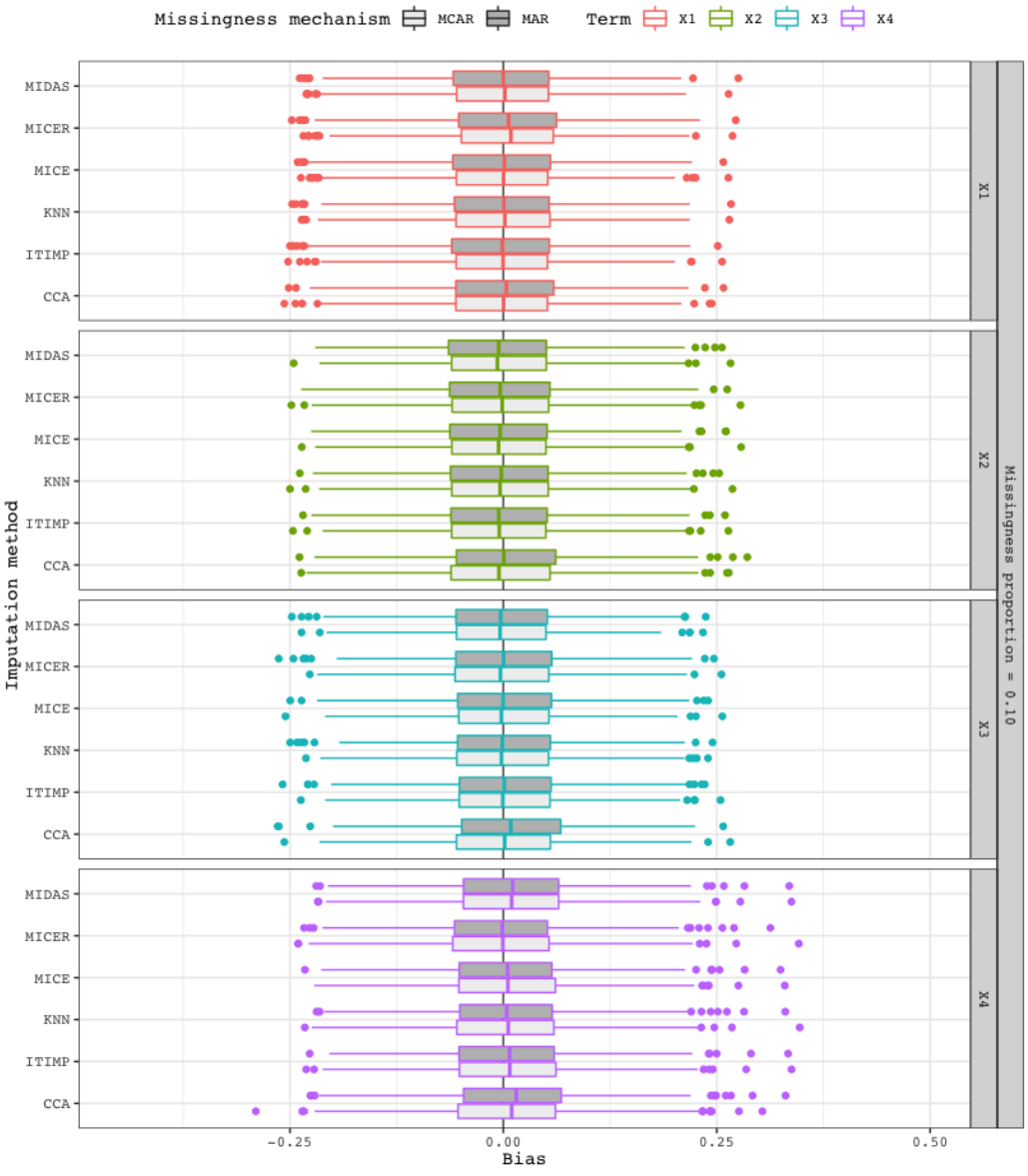


Figure 7: Bias in the estimated effects of X_1, X_2, X_3, X_4 for different imputation methods under MAR and MCAR missingness mechanisms with a missingness proportion of 25%.

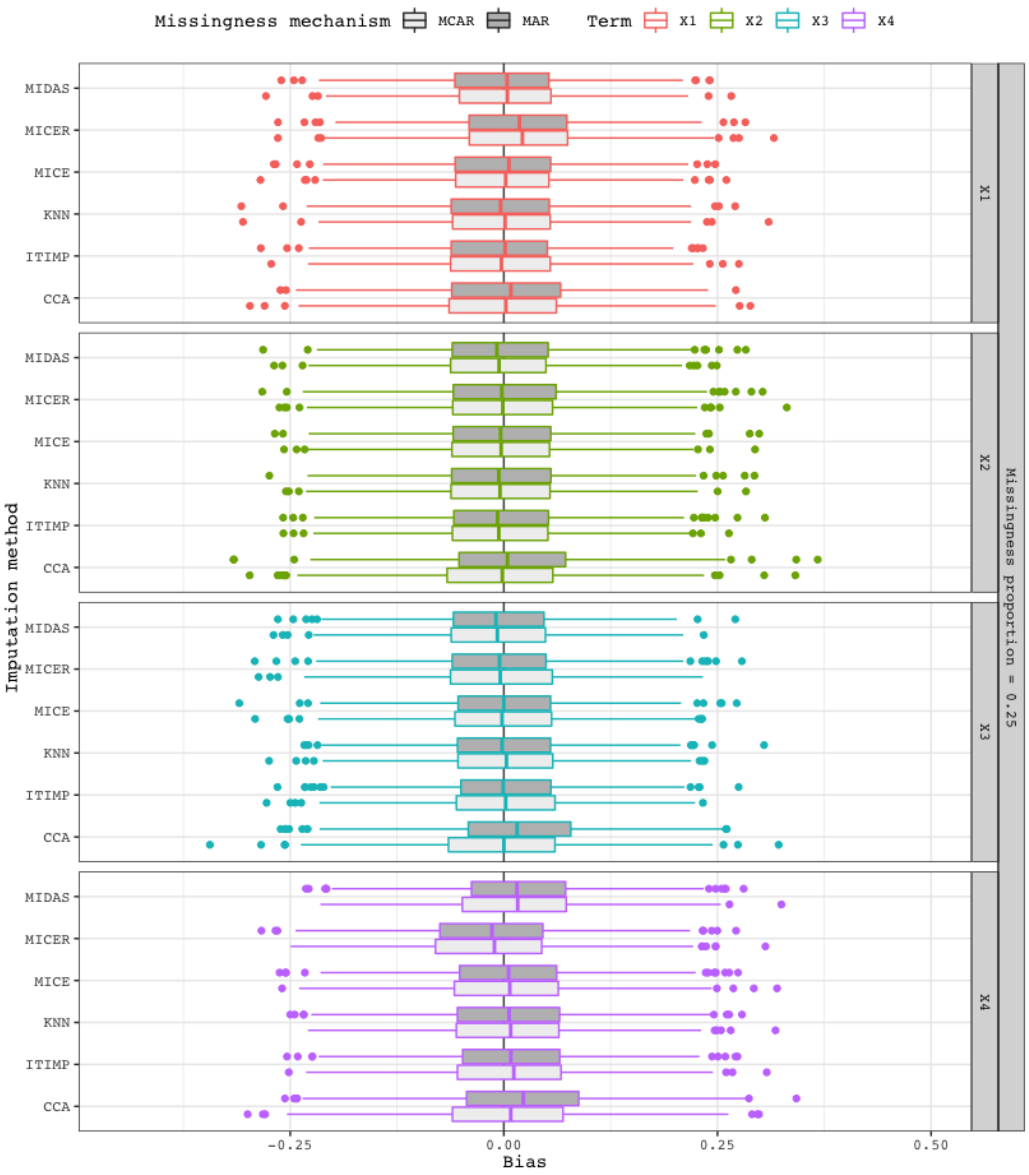
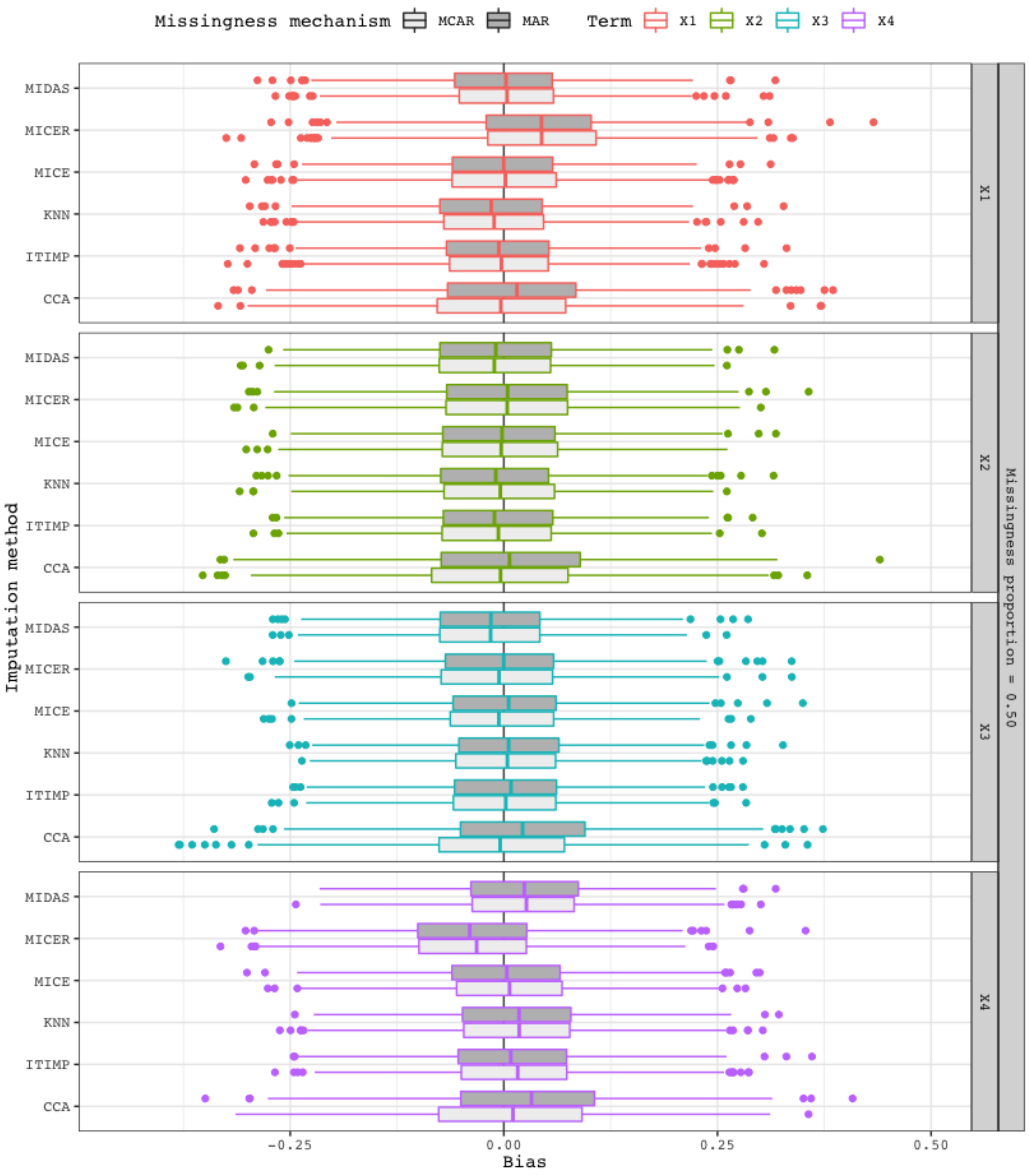
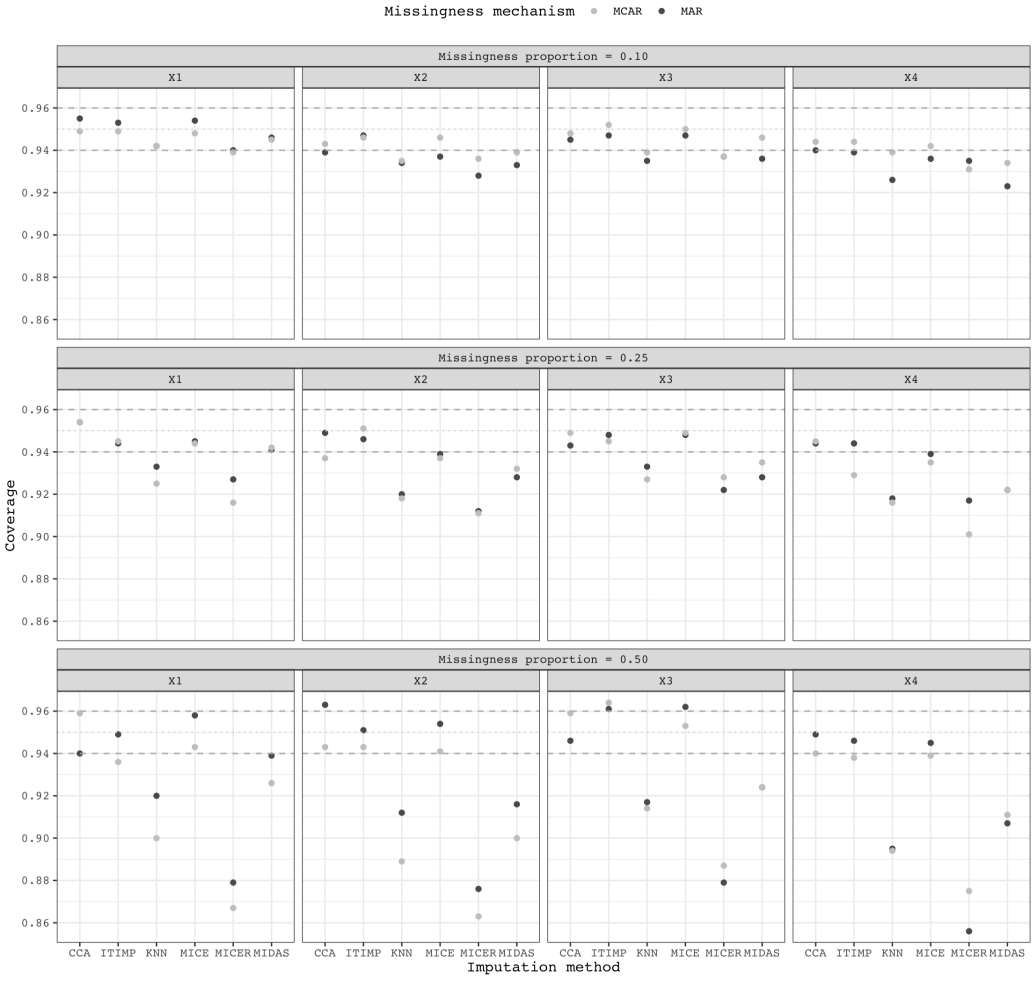


Figure 8: Bias in the estimated effects of X_1, X_2, X_3, X_4 for different imputation methods under MAR and MCAR missingness mechanisms with a missingness proportion of 50%.



Appendix C

Figure 9: Coverage rate in the estimated effects of X_1, X_2, X_3, X_4 for different imputation methods - under MAR and MCAR missingness mechanisms with three missingness proportions (10 %, 25% and 50%).



Appendix D

Figure 10: Confidence interval width in the estimated effects of X_1, X_2, X_3, X_4 for different imputation methods under MAR and MCAR missingness mechanisms with three missingness proportions.