Iterative Imputation in Python

A study on the performance of the package IterativeImputer

Tinke Klomp
6252923

Supervisors:
Hanne Oberman
Gerko Vink
Valentijn de Jong

Department of Methodology & Statistics
University Utrecht
Applied Data Science MSc
1 July 2022

Iterative Imputation in Python

A study on the performance of the package IterativeImputer

Tinke Klomp

Abstract

This study evaluates whether the *Python* package *IterativeImputer* can yield valid estimates through iterative imputation of missing data. The performance was analyzed by means of a simulation study and compared to the benchmark methods of iterative imputation with *mice* in R and complete case analysis. With each simulation repetition data was generated, amputed with varying conditions (e.g. missing data mechanisms and missing data proportions), handled by the three missingness techniques and multiple regression models were estimated. Estimates were evaluated on bias, coverage rate and confidence interval width were pooled and obtained. *IterativeImputer* generated results that were relatively low in bias. However, the produced coverage rates were found to be below nominal coverage. This may be explained by the confidence interval widths, as they were generally too small to contain the true value of the data. The *Python* package doesn't operate as adequately as *mice* and doesn't outperform complete case analysis. Therefore, *IterativeImputer* isn't suitable as a imputation tool for drawing inferences.

Introduction

Missingness is an inevitable problem that occurs with data collection. There are various methods available for attempting to mitigate this issue. One of these mechanisms is iterative imputation. This method, also called conditional modelling imputation, specifies the conditional distribution of each variable (Nijman et al., 2021). As a result, iterative imputation adopts a separate imputation model for each variable showing missingness (Hughes et al. 2014). As an example, for binary variables a logistic regression model can estimated, while a linear regression model can be used for a continuous variable. In *R*, the iterative imputation method can be applied with the *mice* (Multivariate Imputation by Chained Equations) package. Although Python offers packages for handling missing data, there isn't such a package that serves as the de facto standard with which the iterative technique can be performed. The purpose of this study is to analyse whether one of the available iterative imputation packages in *Python* is able to provide valid estimates.

The *Python* package that will be analyzed in thesis is *IterativeImputer*. This package is available in the popular package *scikit-learn*. If this package produces valid estimates, it would give a great amount of data scientists operating with *Python* an opportunity to accurately handle missing data.

The research question that will be studied in this thesis is: *To what extent can IterativeImputer in Python generate valid inferences?* The findings of this project have the potential to gain insight into whether there are valid options for iterative imputation in *Python*. Although *mice* is an influential standard, not all data scientists work with *R*. Many data scientists who don't use *R* operate in *Python*, which would make it beneficial to identify an imputation package in the latter programming language that is capable of generating valid inferences. Additionally, *R* offers a service where packages

are centralized in a repository (CRAN), which facilitates the maintenance and improvement of packages. *Python* doesn't provide such a repository, which may complicate preservation and advancements of the quality of functions. Discerning whether a *Python* package is efficient in producing valid estimates may therefore also be necessary.

A potential consequence of adopting missing data handling methods that produce invalid estimates is that such outcomes can lead to inaccurate conclusions. Not only are such inferences scientifically inadequate, faulty conclusions can have societal repercussions that are difficult to obtain a comprehensive view of. Therefore, the performance of *IterativeImputer* will be examined in a simulation study.

Missing Data Mechanisms

A factor that further should be taken into account for the quality of imputation models is the manner in which missing data is distributed. Missingness can occur in several arrangements. An approach to categorize these patterns is through the concepts of MCAR, MAR and MCAR, classified by Rubin (1976).

Missing completely at random (MCAR) implies that all observations have the same probability of being missing. In contrast, the missing at random concept (MAR) involves data that is missing with a probability that depends on observed data. Finally, missing not at random (MNAR) indicates that the probability of values being missing are by causes that are not captured by the observed data.

Van Buuren (2018) describes that these patterns have different repercussions for the handling of missingness. For instance, because the probability of missing data is induced randomly with a MCAR pattern, it can be relatively convenient to impute data. However this sequence is often an unrealistic scenario (Oberman and Vink, n.d.). MAR is more realistic to occur and is assumed most often in modern missing data methods. MNAR patterns require the most effort to manage, as strategies to handle MNAR involve obtaining more data for the causes of missingness and executing analyses on the sensitivity to various conditions. The missing data patterns can occur simultaneously and are not exclusive to one another. *mice* is able to handle both MCAR and MAR data (Van Buuren and Groothuis-Oudshoorn, 2010). In this study, it is assessed whether *IterativeImputer* is capable of doing so as well.

Methods

The aim of this study is to assess whether *IterativeImputer* generates valid inferences. The validity of these estimates are evaluated through a simulation study, where bias, coverage rate (CR) and confidence interval width (CIW) of inferences produced by *IterativeImputer* are compared to the complete data. The performance of the *Python* imputer is compared to *mice* and complete case analysis (CCA) as benchmark methods.

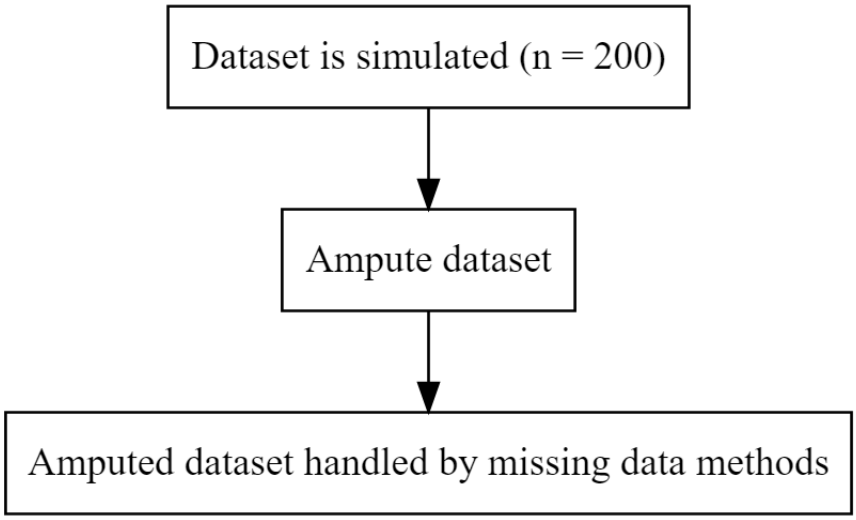
Each simulation round will consist of several stages (see figure 1). First, a multivariate normal distribution dataset is simulated with sample size $n = 200$. Five continuous variables are created, one of which is a dependent variable and the other four are used as the independent variables in the simulation study. The next step is to ampute the dataset with different conditions. The data is alternately amputed with MCAR and right-tailed MAR mechanisms per round. This type of MAR mechanism is implemented, as it generally is the most challenging kind to impute for a method (Oberman and Vink, n.d.). If an imputation technique is able to perform acceptably with right-tailed MAR, it is insightful for the competence of that approach. Additionally, the variables should be correlated sufficiently in case of a MAR mechanism, as the relationship between the variables would otherwise be spurious. In case correlations between variables are too low, spuriousness would result in a MAR mechanism that matches MCAR. The variables are all generated with $mean = 0$ and the independent variables share correlations of 0.3. The regression coefficients of the independent variables are simulated as $X1 = -0.5$, $X2 = -0.1$, $X3 = 0.1$ and $X4 = 0.5$.

Varying proportions of missingness (10%, 25% and 50%) is another condition that is interspersed in this stage. Subsequently, the amputed data is separately imputed by *IterativeImputer* and *mice* each simulation cycle. CCA deletes all rows that contain any missing data during the same stage. *IterativeImputer* maintains a default maximum amount of iterations of 10,

whereas *mice* adopts a maximum of 5 on this aspect. Per round the outcome variable is estimated with the four dependent variables through a multiple linear regression model. The results for bias, CR and CIW are collected afterwards.

The simulation cycle is repeated 1000 times, of which all outcomes relating to the performance measures will be aggregated. This results in a distribution of the three performance measures. At the end of the simulations *IterativeImputer* is compared to the other two techniques based on the bias, coverage rate and confidence interval width distributions.

Figure 1: Stages in simulation cycle



The performance measures are assessed according to the suggestions of Van Buuren (2018). Bias is measured through raw bias (RB) in this study. It is defined as the difference between the expected value of the estimate and the true value: $RB = E(\hat{Q}) - Q)$. An imputation technique should display bias as close to zero as possible to be considered a method that performs sufficiently.

Coverage rate is the proportion of confidence intervals that contain the true value. The CR should be equal or surpass the nominal rate for a method to be deemed acceptable (Van Buuren, 2018). In case that the coverage of a method falls below 90 percent with a nominal rate of 95 percent, it can be

determined to not be confidence valid. If a CR is lower than that benchmark, the method is "too optimistic". This can indicate that confidence intervals are too small, there is too much bias or there is a combination of both these issues (Demirtas et al., 2008). Confidence intervals that are too small imply that p-values are too low, increasing the risk that spurious conclusions can be made from statistical inferences. Too much bias would imply that estimates are made too far from the true value. A coverage rate can be too large as well, for which a cut-off of 99 percent is maintained in this study. When a CR exceeds this value, the approach is regarded as "too conservative", which increases the chance that false negatives are generated. According to Van Buuren (2018) a CR that is too high is less harmful than a rate that is too low. The higher the coverage rate the more uncertainty is displayed due to missingness. Because larger confidence intervals show more uncertainty, there is a higher possibility that the true value falls into the confidence interval.

Finally, confidence interval width is determined to judge the efficiency *IterativeImputer*. This performance measure is closely related to coverage rate. CIW should be as small as possible. However, it shouldn't be too short, as this could make the CR fall below the nominal level, in turn increasing the chance for false positives (Van Buuren, 2018). A width that is too large could result in a coverage rate that is too high, which indicates inefficiency in the missing data method.

As previously mentioned, *IterativeImputer* is compared to *mice* and complete case analysis. *mice* is included into the study, because it continues to operate as a 'gold standard' on the level of missing data imputation. CCA is used for comparison, as out-performing complete case analysis is a minimum requirement for any imputation method to reach. CCA therefore provides a lower limit for reasonable imputation performance. The code to the simulation setup for this study is provided in the Appendix. There are no ethical considerations involved with this study, as the data is simulated.

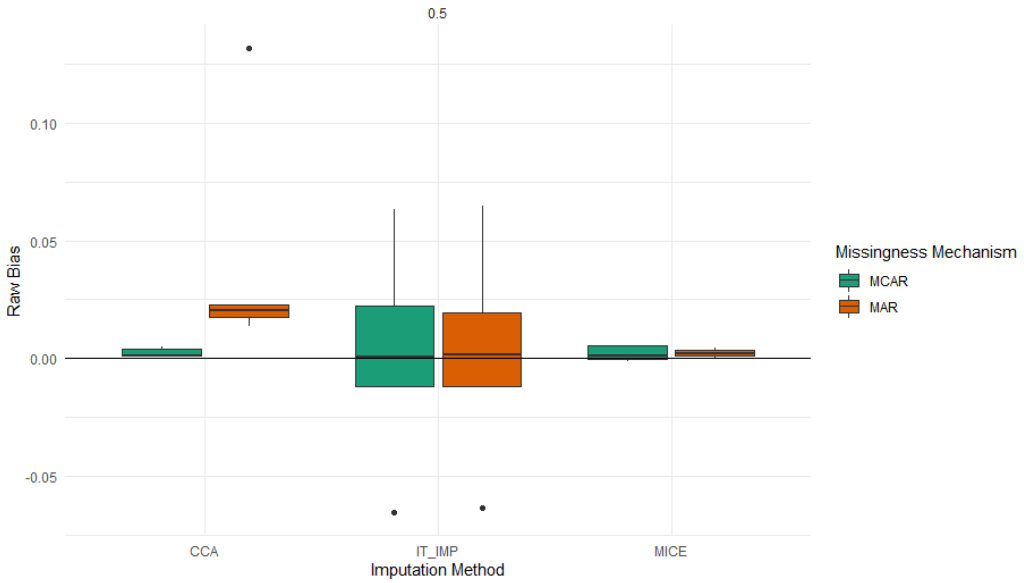
Results

The simulation results display the performance of *IterativeImputer* in terms of bias, coverage rate and confidence interval width. Results are split up by simulation condition (e.g. missingness mechanism and missingness proportion) and compared to the benchmark methods *mice* and CCA. Graphs presented in this thesis only display values that pertain to a 50% missing data percentage and right-tailed MAR, unless otherwise specified, as they show the largest difference in performance. Results achieved with smaller missingness percentages (10% and 25%) are included in the appendix.

Although *IterativeImputer* doesn't display a lot of bias on average across simulation repetitions, there seems to be more variability with this method for both the MCAR and the MAR mechanisms than the other approaches (see figure 2). *mice* demonstrates a small amount of bias with both missingness mechanisms. Complete case analysis exhibits barely any bias with MCAR structures. However, as CCA doesn't accommodate relationships between missingness in one variable and observations of other variables, the method shows the highest average bias under MAR. On average *IterativeImputer* displays comparable performance to *mice*, but the *Python* imputation function appears to have more variability in bias than *mice* and complete case analysis.

Bias was separately calculated for each independent variable that was imputed by *IterativeImputer* and *mice*, or handled by CCA (see appendix, figure 5) for the regression analyses. *IterativeImputer* produces quite varying values in bias between the predictors. Even though X2 and X3 are estimated with small amounts of bias, X1 and X4 deviate quite far from the ideal bias value of 0. *mice*, just as with the average raw bias, presents hardly any bias, when the predictors and intercept are analyzed. CCA demonstrates variability across simulation repetitions for each predictor that is larger compared to the other approaches. *IterativeImputer* therefore seems to accomplish a sim-

Figure 2: Raw Bias based on Imputation Method

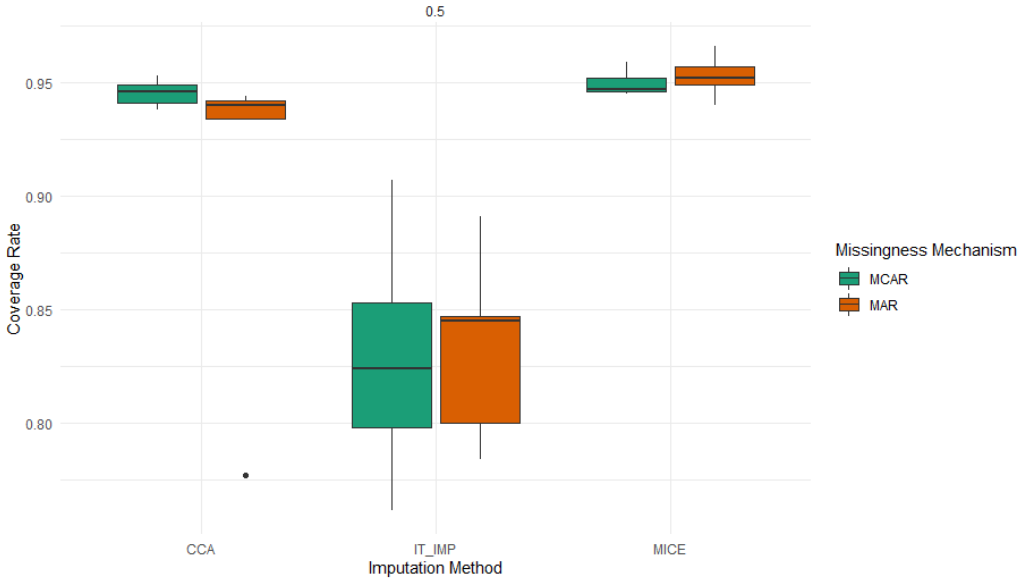


Note: Error bars represent variability in RB across simulation repetitions
Line on y-axis indicates ideal RB

ilar level of performance as complete case analysis, although *mice* produces better results than the other two methods.

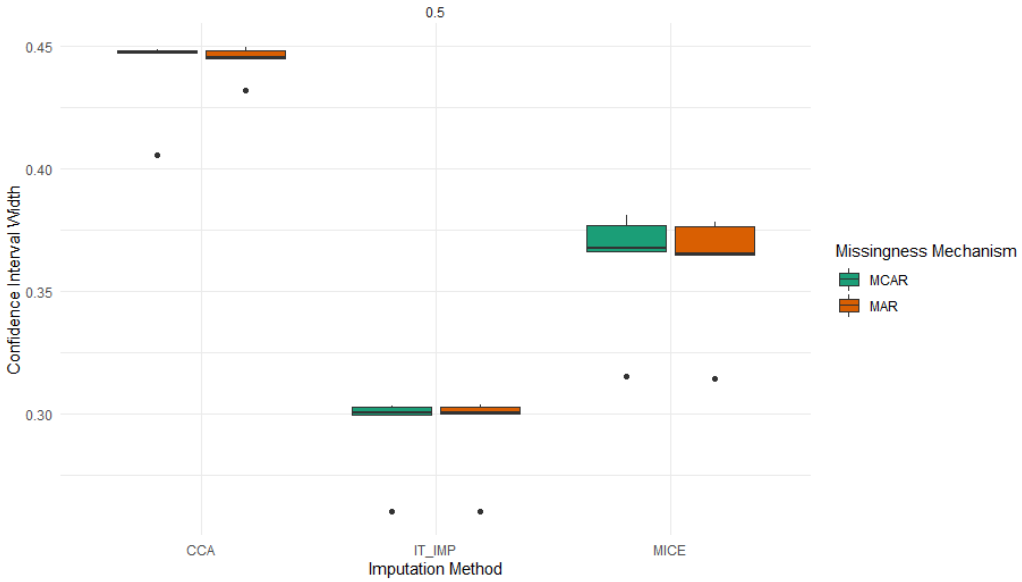
Performance in terms of coverage rates is presented in Figure 3. *IterativeImputer* exhibits a coverage rate that is too low, as it doesn't reach the required minimum 90 percent CR. This could be explained by the fact that the method shows the smallest confidence interval width (CIW) compared to the other approaches (see graph 4). The confidence intervals that are produced don't cover the population estimand adequately. *mice* displays a CR that is large enough. Complete case analysis shows a CR that is also located above the benchmark of 90 percent. This can be explained by a larger CIW, especially compared to the confidence interval widths of the two imputation methods. Normally, coverage rate doesn't show variability, as the measure is a proportion value. However, because the coverage is calculated across 5 estimates for each method and missingness mechanism, the differing values of these variables cause the coverage rate to show variability.

Figure 3: Coverage Rate based on Imputation Method



Note: Error bars represent variability in CR across simulation repetitions

Figure 4: Confidence Interval Width based on Imputation Method



Note: Error bars represent variability in CIW across simulation repetitions

Discussion

This simulation study showed that the Python package IterativeImputer may yield unbiased estimates of regression coefficients, but with too narrow confidence intervals to obtain nominal coverage rates. The answer to the research question "*To what extent can IterativeImputer in Python generate valid inferences?*" is that *IterativeImputer* generates coverage rates that lie below the required criterion of 90 percent, which indicates that the package performs poorly. This result is corroborated by the observed confidence interval widths, which suggests that *IterativeImputer* produces confidence intervals that are too narrow. A CIW that is too low increases the chance that the true value of the regression estimates will fall outside of the confidence interval, which indicates that the imputation method doesn't capture the uncertainty due to missingness.

IterativeImputer was compared to *mice*, which is regarded as a 'gold standard' for imputation in *R*, and complete case analysis, functioning as a minimal benchmark the *Python* package should outperform. The *Python* method was not able to perform as well as *mice* and couldn't provide more sufficient estimates than CCA. Because *IterativeImputer* displays a poor performance on both coverage rate and confidence interval width compared to the other two methods, it can't be concluded that the method produces valid statistical inferences. This package should not be considered as an imputation tool if the goal is inference.

The generalizability of this study is restricted due to the source and the type of data that was used. The data was simulated to control the conditions of the analysis (e.g. missingness mechanisms and proportions of missing data). *IterativeImputer* weren't tested on empirical data.

The simulated dataset that was used exclusively contained continuous, normally distributed variables. However, iterative imputation is a technique that separately estimates an imputation model for each variable based on

the distribution of that attribute (Hughes et al. 2014; Nijman et al., 2021). The performance of *IterativeImputer* wasn't examined on data with other and varying types of distributions, such as binary variables.

Future studies should apply *IterativeImputer* in observed data to examine the performance of the package in other circumstances than a simulated dataset. Further research could consider the competence of *IterativeImputer* to produce valid inferences with data where varying distributions are incorporated in the analysis.

The applicability of *IterativeImputer* in prediction modeling is an aspect that could be studied as well. Low levels of bias are required to make valid predictions. The *Python* package produces imputations that are relatively unbiased, meaning that the difference between true values and estimates is quite low. It would be valuable to assess whether *IterativeImputer* would be more competent in providing imputations for data that is used to predict.

Lastly, *IterativeImputer* isn't the only imputation package that *Python* has to offer. Other packages and functions in the programming language could be studied to possibly find a method that does generate proper estimates through missing data imputation in *Python*.

References

Demirtas, H., Freels, S.A. Yucel, R.M. (2008). Plausibility of Multivariate Normality Assumption When Multiply Imputing Non-Gaussian Continuous Outcomes: A Simulation Assessment. *ournal of Statistical Computation and Simulation 78*(1), 69-84.

Hughes, R.A., White, I.R., Seaman, S.R., Carpenter, J.R., Tilling, K. Sterne, J.A.C. (2014). Joint modelling rationale for chained equations. *BMC Medical Research Methodology, 14*(28).

Nijman, S.W.J., Groenhof, T.K.J., Hoogland, J., Bots, M.L., Brandjes, M., Jacobs, J.J.L., Asselbergs, F.W., Moons, K.G.M. Debray, T.P.A. (2021). Real-time imputation of missing predictor values improved the application of prediction models in daily practice. *Journal of Clinical Epidemiology 134*, 22-34.

Oberman, H.I. Vink, G. (n.d.). *Towards a Standardized Evaluation of Imputation Routines*. Retrieved from https://doi.org/https://www.gerkovink.com/evaluation.

Rubin, D. (1976). Inference and Missing Data. *Biometrika, 63*(3), 581-590.

Van Buuren, S. (2018). *Flexible Imputation of Missing Data* (2nd ed). London: Chapman and Hall/CRC.

Van Buuren, S., Groothuis-Oudshoorn, K. (2010). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software, 45*(3), 1-67.

Appendix

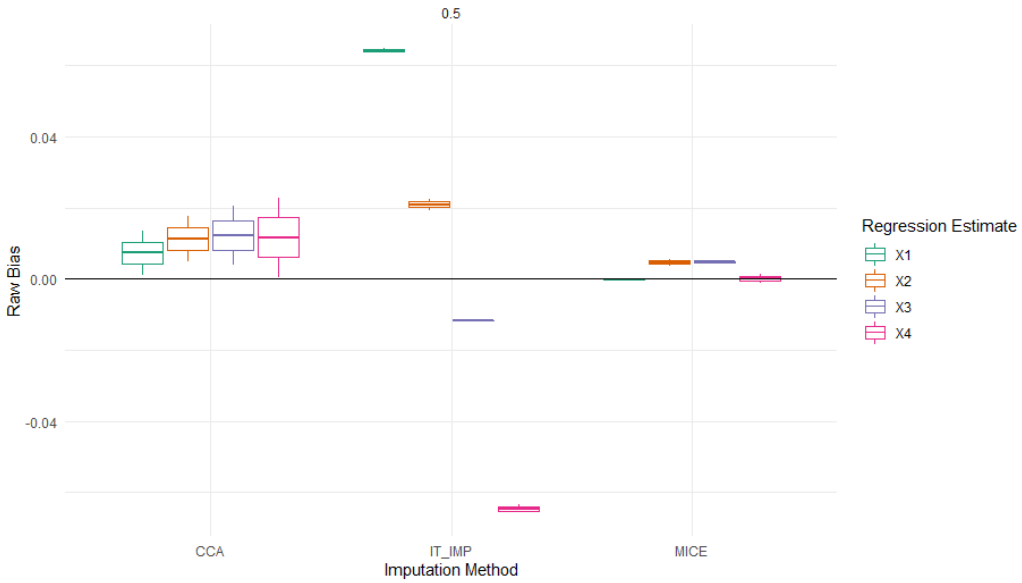
Simulation Setup

The code to the simulation setup for this study is provided with the following link: https://github.com/tinkeklomp/Iterative-Imputation-in-Python

Graphs

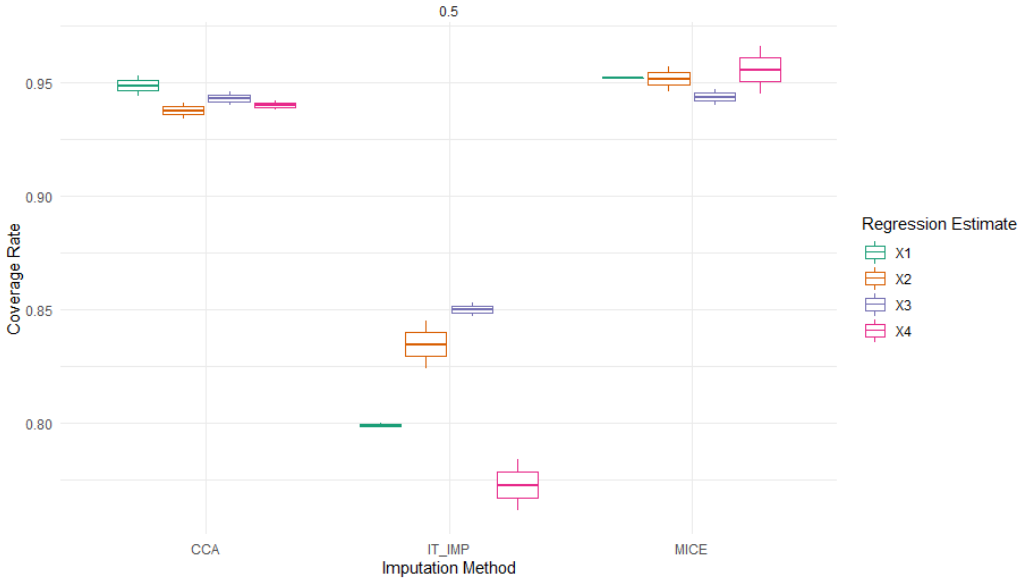
An analysis of coverage rate and confidence interval width was separately applied to the predictors of which the missing data was handled. The results of this comparison are largely similar to the evaluation of the missing data related to missingness mechanisms. Again, *IterativeImputer* can't be deemed confidence valid based on coverage rate that is far too low. This again could be explained by the CIWs that are quite small. The *Python* method is overly confident. CCA is the least efficient method, as the method's confidence intervals are wide. *mice* performs more efficiently, while simultaneously being confidence valid.

Figure 5: Raw Bias of Regression Estimates



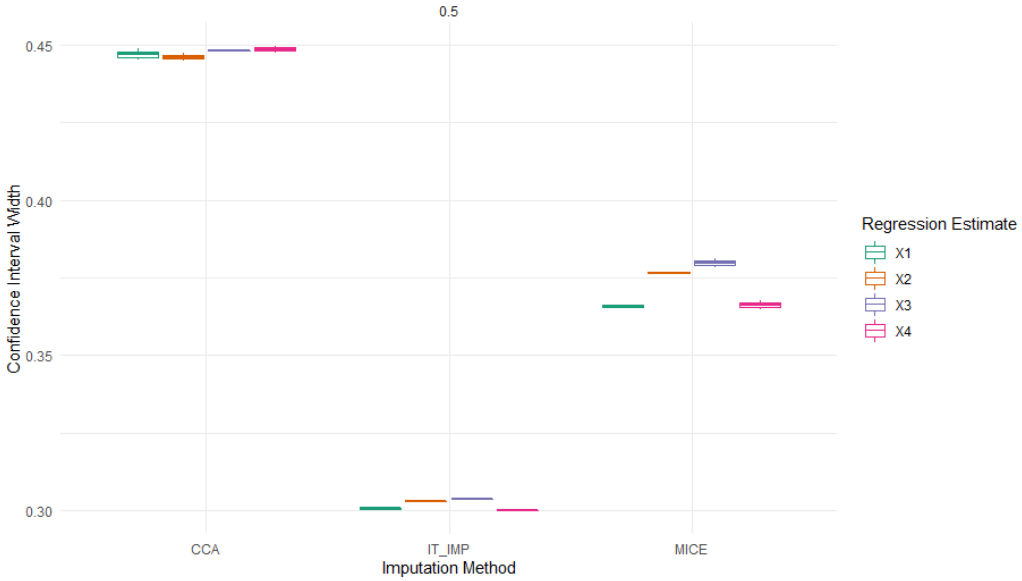
Note: Error bars represent variability in RB across simulation repetitions
Line on y-axis indicates ideal RB; Missingness proportion is 50%

Figure 6: Coverage Rate of Regression Estimates



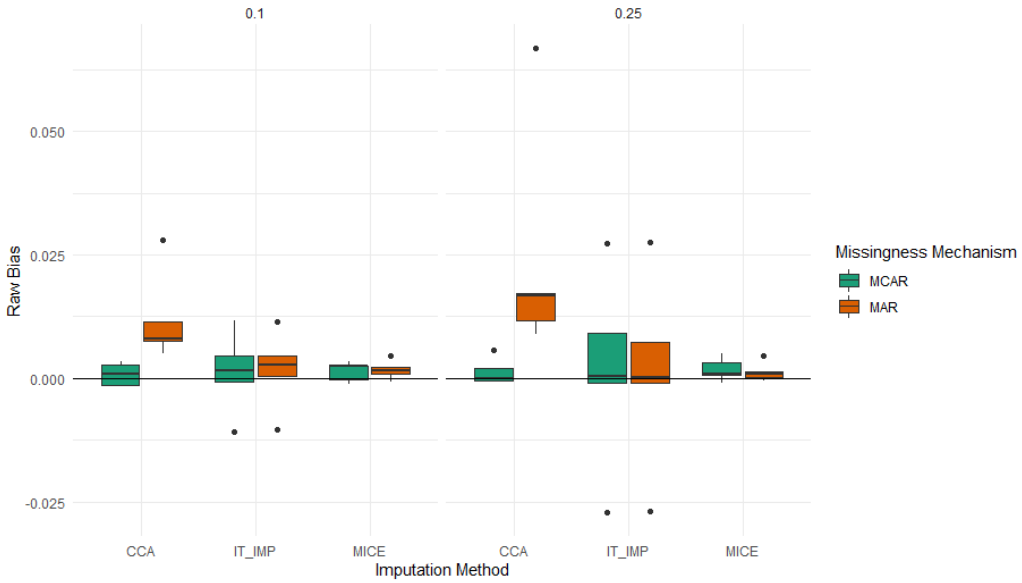
Note: Error bars represent variability in CR across simulation repetitions
Missingness proportion is 50%

Figure 7: Confidence Interval Width of Regression Estimates



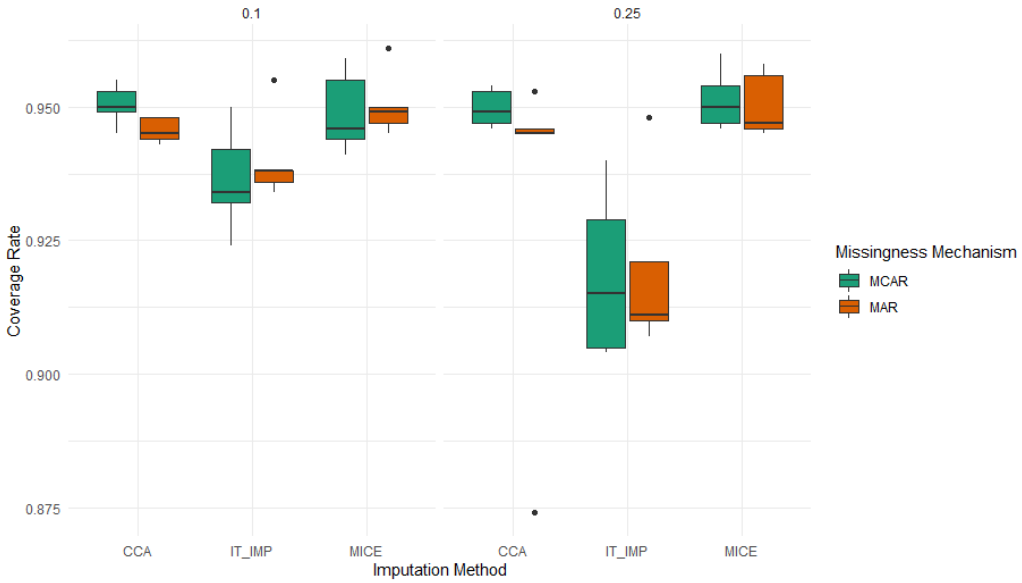
Note: Error bars represent variability in CIW across simulation repetitions
Missingness proportion is 50%

Figure 8: Raw Bias based on Imputation Method



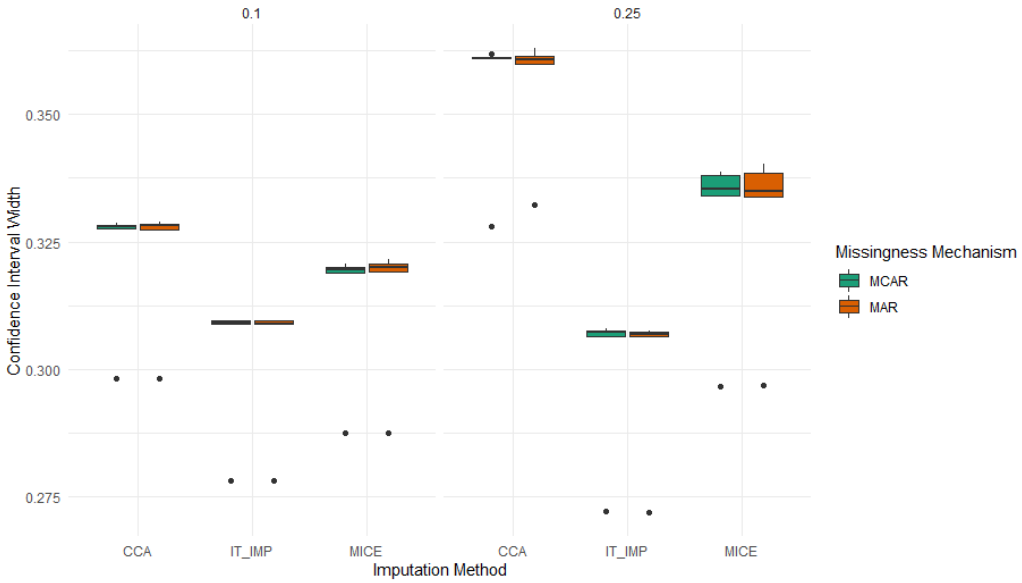
Note: Error bars represent variability in RB across simulation repetitions
Line on y-axis indicates ideal RB; Missingness proportions are 10% and 25%

Figure 9: Coverage Rate based on Imputation Method



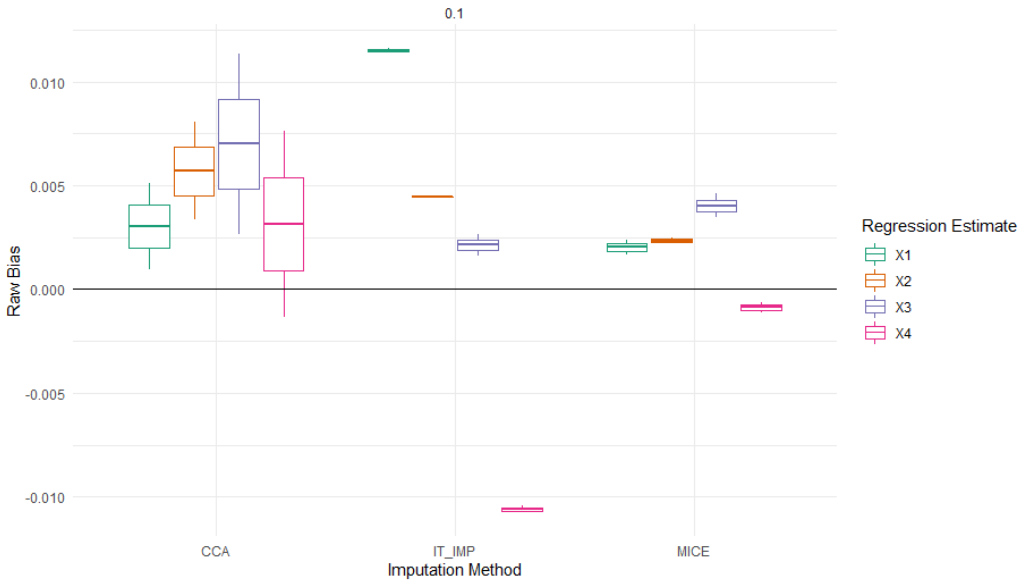
Note: Error bars represent variability in CR across simulation repetitions
Missingness proportions are 10% and 25%

Figure 10: Confidence Interval Width based on Imputation Method



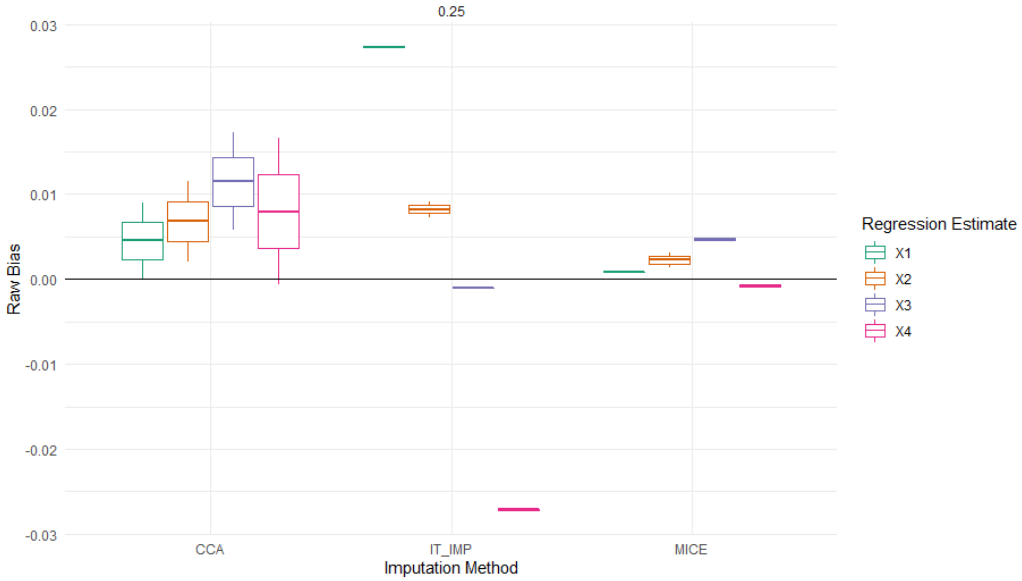
Note: Error bars represent variability in CIW across simulation repetitions
Missingness proportions are 10% and 25%

Figure 11: Raw Bias of Regression Estimates



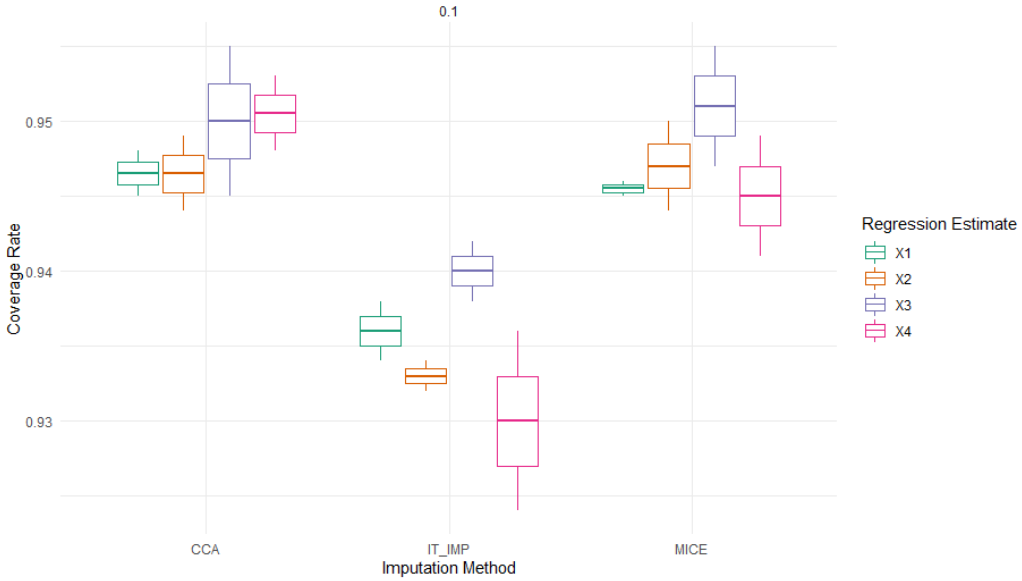
Note: Error bars represent variability in RB across simulation repetitions
Line on y-axis indicates ideal RB; Missingness proportion is 10%

Figure 12: Raw Bias of Regression Estimates



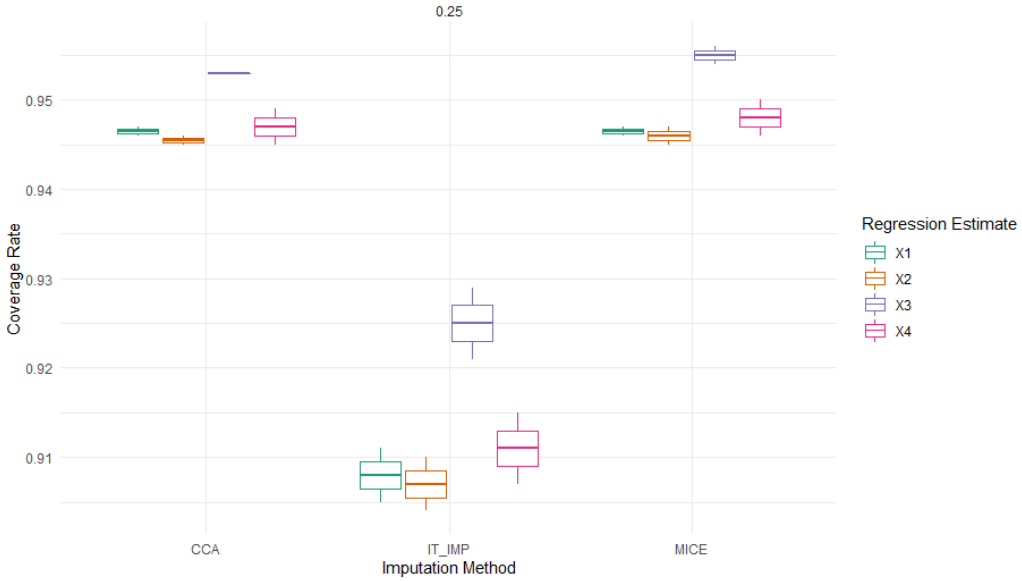
Note: Error bars represent variability in RB across simulation repetitions
Line on y-axis indicates ideal RB; Missingness proportion is 25%

Figure 13: Coverage Rate of Regression Estimates



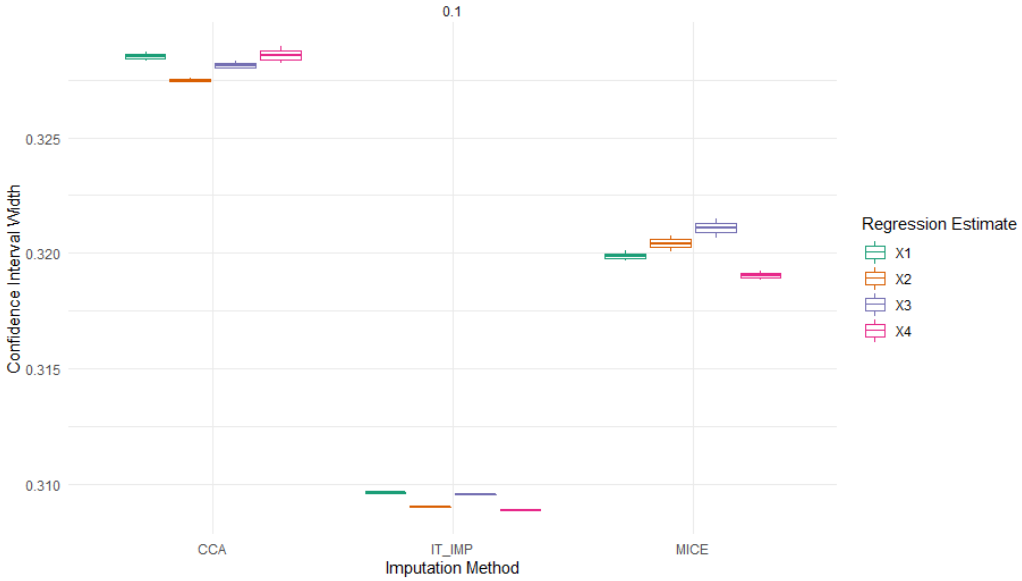
Note: Error bars represent variability in CR across simulation repetitions
Missingness proportion is 10%

Figure 14: Coverage Rate of Regression Estimates



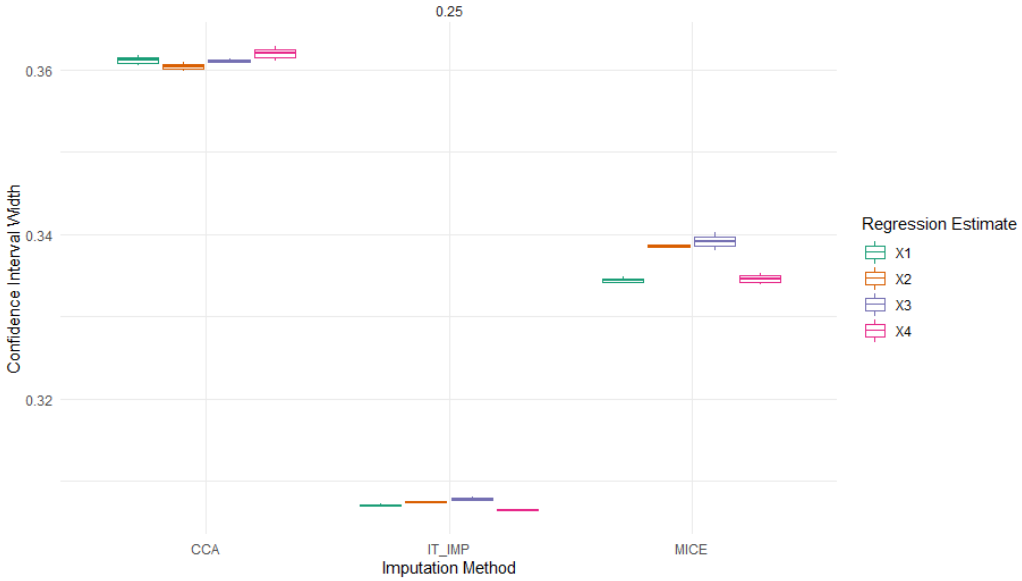
Note: Error bars represent variability in CR across simulation repetitions
Missingness proportion is 25%

Figure 15: Confidence Interval Width of Regression Estimates



Note: Error bars represent variability in CIW across simulation repetitions
Missingness proportion is 10%

Figure 16: Confidence Interval Width of Regression Estimates



Note: Error bars represent variability in CIW across simulation repetitions
Missingness proportion is 25%