Non-convergence in iterative imputation

H. I. Oberman

Iterative imputation has become the de facto standard to accommodate for the ubiquitous problem of missing data. While it is widely accepted that this technique can yield valid inferences, these inferences all rely on algorithmic convergence. Our study provides insight into identifying non-convergence in iterative imputation algorithms. We show that these algorithms can yield correct outcomes even when a converged state has not yet formally been reached. In the cases considered, inferential validity is achieved after five to ten iterations, much earlier than indicated by diagnostic methods. We conclude that it never hurts to iterate longer, but such calculations hardly bring added value.

Introduction

Multiple imputation has become the de facto standard to accommodate missing data in social scientific research. The technique enables researchers to draw valid inferences from incomplete data (Rubin 1976). Multiple imputation separates the missing data problem from the scientific problem. First, the missing values are imputed (i.e., filled in) using an imputation algorithm. Once the data are completed, the analysis of scientific interest can be performed on the completed data. This process is repeated several times to obtain a distribution of plausible values, which reflects the uncertainty in the data due to missingness. The analysis of scientific interest thus yields one estimate for each imputed dataset. Afterwards, the estimates from the different imputations are pooled into a single estimate. This pooled estimate is unbiased and confidence-valid if–and only if–both the missing data problem and the scientific problem are appropriately considered.

To obtain valid scientific estimates from incomplete data, both the missing data problem and the scientific problem should be appropriately considered. The popular imputation algorithm 'Multivariate Imputation by Chained Equetions' (MICE) has proven to be a powerful tool to draw valid inferences from incomplete data under many circumstances (Buuren and Groothuis-Oudshoorn 2011; Van Buuren 2018). The algorithm is named after its iterative nature: missing data are imputed on a variable-by-variable basis, using a sequence of univariate imputation models. The validity of this iterative process naturally depends on the convergence of the

imputation algorithm. Yet, there has not been a systematic study on how to evaluate the convergence of iterative imputation algorithms.

[TODO: move/integrate this section?] There is no scientific consensus on how to evaluate the convergence of imputation algorithms (Zhu and Raghunathan 2015; Takahashi 2017). Moreover, the behavior of such algorithms under certain default imputation models (e.g., 'predictive mean matching') is an entirely open question (Murray 2018). Therefore, algorithmic convergence should be monitored carefully—although this is not straightforward.

Determining whether an algorithm has converged is not trivial, especially in the context of iterative imputation. Since the aim of iterative imputation is to converge to a distribution and not to a single point, the algorithm produces some fluctuations even after it has converged. Because of this property, it may be more desirable to focus on *non*-convergence. A widely accepted practice is visual inspection of the algorithm (Raghunathan and Bondarenko 2007; Van Buuren 2018). Diagnosing non-convergence through visual inspection may, however, be undesirable: 1) it may be challenging to the untrained eye, 2) only severely pathological cases of non-convergence may be diagnosed, and 3) there is not an objective measure that quantifies convergence (Van Buuren 2018). [TODO: add argument: "Inspection of such plots is a notoriously unreliable method of assessing convergence and in addition is unwieldy when monitoring a large number of quantities of interest, such as can arise in complicated hierarchical models" (Gelman et al., 2013, p. 285).] Therefore, a quantitative diagnostic method to assess convergence would be preferred. Fortunately, there are non-convergence identifiers for *other* iterative algorithms, but the validity of these identifiers has not been evaluated on imputation algorithms.

In this study, we explore different methods for identifying non-convergence in the iterative imputation algorithm MICE. We evaluate whether these methods are able to cover the extent of the non-convergence, and we also investigate the relation between non-convergence and the validity of the inferences. We translate the results of our simulation study into guidelines for practice, which we demonstrate by means of a case study.

[TODO: merge with prev section] In this paper we study different methods for assessing non-convergence in iterative imputation algorithms. We define several diagnostics and evaluate how these diagnostics could be appropriate for iterative imputation applications. We then address the impact of inducing non-convergence in iterative imputation algorithms through model-based simulation in `R` (R Core Team 2024). For reasons of brevity, we only focus on the iterative imputation algorithm implemented in the popular `mice` package in `R` (Van Buuren and Groothuis-Oudshoorn 2011). The aim of the simulation study is to determine whether unbiased, confidence-valid inferences may be obtained if the algorithm has not (yet) converged. And additionally, to evaluate the behavior and performance of several diagnostic methods to identify non-convergence. With that, we formulate an informed advice on when it is safe to conclude that the algorithm is converged *enough* for valid inferences. We translate the results of the study into guidelines for applied researchers, which may facilitate drawing valid inferences from incomplete data.

Background

Notation

Let $\mathbf{Y}$ denote an $n \times p$ matrix containing the data values on p variables for all n units in a sample. The data value of unit i ($i = 1, 2, ..., n$) on variable Y_j ($j = 1, 2, ..., p$) may be either observed or missing. The collection of observed data values in $\mathbf{Y}$ is denoted by $\mathbf{Y}_{\text{obs}}$; the missing part of $\mathbf{Y}$ is referred to as $\mathbf{Y}_{\text{mis}}$. We define the proportion of incomplete cases, π_{inc}, as the number of units with at least one missing value divided by the total number of units n in dataset $\mathbf{Y}$.

Figure Figure 1 provides an overview of the steps involved with MI. The missing part $\mathbf{Y}_{\text{mis}}$ of an incomplete dataset is imputed m times. This creates m sets of imputed data $\dot{\mathbf{Y}}_{\text{imp},\ell}$, where $\ell = 1, 2, ..., m$. The imputed data is then combined with the observed data $\mathbf{Y}_{\text{obs}}$ to create m completed datasets. On each of these datasets, the analysis of scientific interest is performed to estimate Q: the quantity of scientific interest (e.g., a regression coefficient). Since Q is estimated on each completed dataset, m separate $\hat{Q}_\ell$-values are obtained. Finally, the $\hat{Q}_\ell$-values are combined into a single pooled estimate $\bar{Q}$. The premise of multiple imputation is that $\bar{Q}$ is an unbiased and confidence-valid estimate of the true—but unobserved—scientific estimand Q (Rubin 1996).

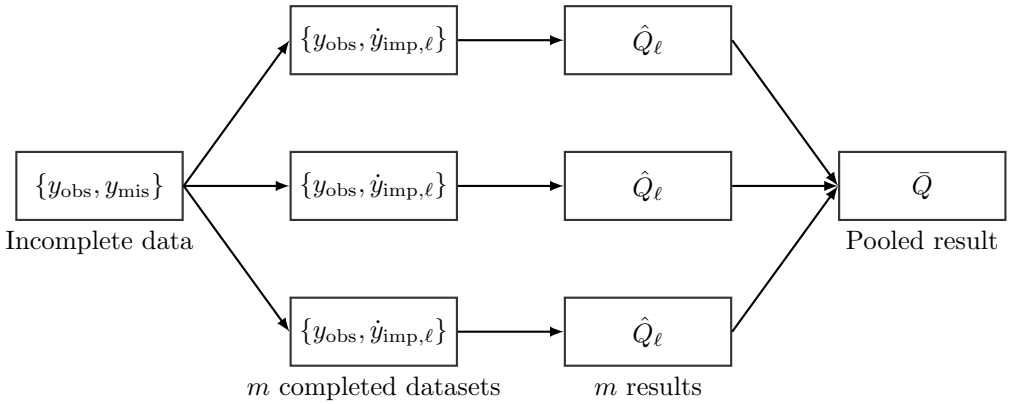


Figure 1: Scheme of the main steps in multiple imputation—from an incomplete dataset, to $m = 3$ multiply imputed datasets, to $m = 3$ estimated quantities of scientific interest $\hat{Q}$, to a single pooled estimate $\bar{Q}$.

A popular method to obtain imputations is to use the 'Multiple Imputation by Chained Equations' algorithm, shorthand 'MICE' (Buuren and Groothuis-Oudshoorn 2011). With MICE, imputed values are drawn from the posterior predictive distribution of the missing values on a variable-by-variable basis. The algorithm is initialized by filling in m sets of starting values, randomly drawn from the observed data. The algorithm then iterates over the p variables, generating m sets of imputed values for each variable j. After one full run through the data,

the imputed values are 'updated' in the next iteration. This process should be repeated until convergence is reached.

With iterative imputation, plausible values are sampled at every iteration t of the imputation algorithm. But only the imputed values drawn at the final iteration τ (where $t = 1, 2, \ldots, \tau$) are used for further analysis. The state-space of the algorithm at a certain iteration t may be summarized by a scalar summary θ. A typical θ to track over iterations is the 'chain mean': the average of the imputed values for a variable Y_j at iteration t. The collection of θ-values between $t = 1$ (the state-space of the algorithm after initialization) and $t = \tau$ (the state-space for the imputed values) will be referred to as an 'imputation chain'.

Note 1: Algorithm 1: MICE algorithm

Initialization phase $(t = 0)$

1. Specify an imputation model for each variable Y_j, where $j = 1, ..., p$.
2. Repeat over variables $(j = 1, ..., p)$.
    2. Repeat over imputations $(\ell = 1, ..., m)$.
        a. Fill in starting imputations $(\dot{Y}_j^0)$ by random draws from the observed part of each variable (Y_j^{obs}).
3. End repeat j.

Iteration phase $(t = 1, ..., \tau)$

4. Repeat over iterations $(t = 1, ..., \tau)$.
    a. Repeat over variables $(j = 1, ..., p)$.
    I. Define $\dot{\mathbf{Y}}_{-j}^t$ as the currently complete data *except* variable Y_j at iteration t, where $\dot{\mathbf{Y}}_{-j}^t = (\dot{Y}_1^t, \ldots, \dot{Y}_{j-1}^t, \dot{Y}_{j+1}^{t-1}, \ldots, \dot{Y}_p^{t-1})$.
    II. Draw imputation model parameters $\dot{\phi}_j^t$ from the posterior predictive distribution of the observed data, where $\dot{\phi}_j^t \sim P(\phi_j^t | Y_j^{\text{obs}}, \dot{Y}_{-j}^t, R)$.
    III. Draw imputations $\dot{Y}_j^t$, where $\dot{Y}_j^t \sim P(Y_j^{\text{mis}} | Y_j^{\text{obs}}, \dot{Y}_{-j}^t, R, \dot{\phi}_j^t)$.
    b. End repeat j.
5. End repeat t.

Convergence of MICE

Iterative imputation algorithms such as MICE often are special cases of Markov chain Monte Carlo (MCMC) methods. In MCMC methods, convergence is not from a scalar to a point, but from one distribution to another. The values generated by the algorithm will vary even after

convergence (Gelman et al. 2013). The most converged state that such an algorithm may reach is *approximate* convergence. Since MCMC algorithms do not reach a unique point at which convergence is established, diagnostic methods may only identify signs of *non*-convergence (Hoff 2009).

Currently, the recommended practice for evaluating the convergence of the MICE algorithm is through visual inspection (Raghunathan and Bondarenko 2007; Van Buuren 2018). After running the imputation algorithm for a certain number of iterations, researchers are encouraged to produce traceplots. In a traceplot, a scalar summary of the state-space of the algorithm θ is plotted against the iteration number. Non-convergence is diagnosed if the chains show definite trends, or if the imputation chains are not freely intermingled with one another (Van Buuren 2018, § 6.5.2). These guidelines are in line with the two requirements for convergence in MCMC algorithms: stationarity and mixing (Gelman et al. 2013). Stationarity is characterized by the absence of trending across iterations. Mixing implies that the chains intermingle nicely. If one of the two requirements is not met, we speak of non-convergence. Without mixing, imputation chains may be 'stuck' at a local optimum instead of sampling imputed values from the entire predictive posterior distribution of the missing values. The distribution of imputed values then differs across imputations. This may cause under-estimation of the variance between chains, which results in spurious, invalid inferences. Without stationarity, there is trending within imputation chains. Trending implies that further iterations would yield a systematically lower or higher set of imputations. Iterative imputation algorithms that have not (yet) reached stationarity, may thus yield biased estimates.

Non-convergence diagnostics

There are many diagnostic tools to identify non-convergence in iterative (MCMC) algorithms (Brooks and Gelman 1998; El Adlouni, Favre, and Bobée 2006). Non-convergence is diagnosed using an identifier and a parameter. The parameter can be any statistic that we track across iterations, for example the average imputed value per imputation (i.e., chain means). The identifier is a calculation of some sort that quantifies non-convergence in the scalar. We consider only two of them that may be appropriate for imputation algorithms—one to monitor signs of non-mixing, and one for non-stationarity. As recommended by e.g. Cowles and Carlin (1996) we will use the potential scale reduction factor $\widehat{R}$ to evaluate mixing ['Gelman-Rubin statistic'; Gelman and Rubin (1992)], and autocorrelation to diagnose trending [AC; Schafer (1997); Gelman et al. (2013)]. With a recently proposed adaptation, $\widehat{R}$ may also serve to diagnose non-stationarity (Vehtari et al. 2021). We will evaluate the appropriateness of $\widehat{R}$ and AC. Other methods are outside the scope of this study, e.g. because they assume that values within chains represent independent samples, whereas the MICE algorithm only uses the final iteration to produce imputations.

Potential scale reduction factor

In 2021, Vehtari et al. published an updated version of the potential scale reduction factor $\widehat{R}$, originally coined by Gelman and Rubin in 1992. This adapted version aims to detect non-mixing even in the tails of distributions, and non-stationarity over iterationsuses. To achieve this, the 2021 version uses three transformations on the scalar summary θ before computing $\widehat{R}$-values: rank-normalization, folding, and localization. To define $\widehat{R}$, we largely follow Vehtari et al.'s formulation (2019, p. 5), with a few exceptions. Let m be the total number of chains, τ the number of iterations per chain (where $\tau \geq 2$), and θ the scalar summary of interest. For each chain ($\ell = 1, 2, \ldots, m$), we estimate the variance of θ, and average these to obtain within-chain variance W.

$$W = \frac{1}{m} \sum_{\ell=1}^{m} s_\ell^2, \text{ where } s_\ell^2 = \frac{1}{\tau - 1} \sum_{t=1}^{\tau} \left(\theta^{(t\ell)} - \bar{\theta}^{(\cdot\ell)}\right)^2.$$

We then estimate between-chain variance B (note that we diverge from the typical notation in MI, where B denotes the variance between the estimated quantities of scientific interest $\hat{Q}_\ell$ TODO: add not between imp var cf Rubin). B is defined as the variance of the collection of average θs per chain:

$$B = \frac{\tau}{m - 1} \sum_{\ell=1}^{m} \left(\bar{\theta}^{(\cdot\ell)} - \bar{\theta}^{(\cdot\cdot)}\right)^2, \text{ where } \bar{\theta}^{(\cdot\ell)} = \frac{1}{\tau} \sum_{t=1}^{\tau} \theta^{(t\ell)}, \; \bar{\theta}^{(\cdot\cdot)} = \frac{1}{m} \sum_{\ell=1}^{m} \bar{\theta}^{(\cdot\ell)}.$$

From the between- and within-chain variances we compute a weighted average, $\widehat{\text{var}}^+$, which over-estimates the total variance of θ. $\widehat{R}$ is then obtained as a ratio between this total variance and the within-chain variance:

$$\widehat{R} = \sqrt{\frac{\widehat{\text{var}}^+(\theta|y)}{W}}, \text{ where } \widehat{\text{var}}^+(\theta|y) = \frac{\tau - 1}{\tau} W + \frac{1}{\tau} B.$$

We can interpret $\widehat{R}$ as potential scale reduction factor since it indicates by how much the variance of θ could be shrunken down if an infinite number of iterations per chain would be run (Gelman and Rubin 1992). The assumption underlying this interpretation is that chains are 'over-dispersed' at $t = 1$, and reach convergence as $\tau \to \infty$. Over-dispersion implies that the initial values of the chains are 'far away' from the target distribution and each other. When the sampled values in each chain are independent of the chain's initial value, the mixing component of convergence is satisfied. The variance between chains, B, is then equivalent to the variance within chains, W, and $\widehat{R}$-values will be close to one. High $\widehat{R}$-values thus indicate non-convergence.

Autocorrelation

Autocorrelation is defined as the correlation between two subsequent θ-values within the same chain (Lynch 2007, 147). In this study, we only consider AC at lag 1, i.e., the correlation between the t^{th} and $(t+1)^{th}$ iteration of the same chain. Following the same notation as for $\widehat{R}$,

$$AC = \left(\frac{\tau}{\tau - 1}\right) \frac{\sum_{t=1}^{\tau-1}(\theta_t - \bar{\theta}^{(\cdot m)})(\theta_{t+1} - \bar{\theta}^{(\cdot m)})}{\sum_{t=1}^{\tau}(\theta_t - \bar{\theta}^{(\cdot m)})^2}.$$

We can interpret AC-values as a measure of non-stationarity. If there is dependence between subsequent θ-values in imputation chains, AC-values are non-zero. Positive AC-values occur when θ-values are recurring (i.e., high θ-values are followed by high θ-values, and low θ-values are followed by low θ-values). Recurrence within imputation chains may lead to trending. Negative AC-values occur when θ-values of subsequent iterations are less similar, or diverge from one another. Divergence within imputation poses no threat to the convergence of the algorithm—it may even speed up convergence. Complete stationarity is reached when $AC = 0$. As non-convergence diagnostic, our interest is in positive AC-values.

Thresholds

Upon approximate convergence, the imputation chains intermingle such that the only difference between the chains is caused by the randomness induced by the algorithm ($\widehat{R} \approx 1$), and there is little dependency between subsequent iterations of imputation chains ($AC \approx 0$). In practice, we diagnose non-convergence when approximate convergence is violated, i.e., when $\widehat{R}$ and AC exceed a certain threshold. The conventional thresholds to diagnose non-mixing are $\widehat{R} > 1.2$ (Gelman and Rubin 1992) or $\widehat{R} > 1.1$ (Gelman et al. 2013). Vehtari et al. (2021) proposed a much more stringent threshold of $\widehat{R} > 1.01$. The magnitude of AC-values may be evaluated statistically, using a Wald test with $AC = 0$ as null hypothesis (Box et al. 2015). AC-values that are significantly higher than zero indicate non-stationarity.

Simulation set-up

We investigate non-convergence in iterative imputation through model-based simulation in R (version 4.4.2; R Core Team (2024)). We provide a summary of the simulation set-up in Algorithm 2; the complete script and technical details are available from github.com/hanneoberman/MissingThePoint [TODO: make this a Zenodo DOI]. The number of simulation repetitions $n_{\text{sim}} = 2000$.

[TODO: add different correlations? add different imputation models (incompatible/PMM)?]

Table 1: Missingness scenarios

Missingness conditions	
missing data mechanism	proportion of incomplete cases
×	
MCAR, MAR	25%, 50%, 75%

Note 2: Algorithm 2: simulation set-up (pseudo-code)

```
for (each simulation run) {
  simulate complete data;
  for (each missingness condition) {
    create missingness;
    for (each imputation model iteration) {
      impute missingness;
      estimate quantities of scientific interest;
      apply performance measures to the estimates;
      compute non-convergence diagnostics
} } }
```

Aims

With this simulation, we assess the impact of non-convergence on the validity of scientific estimates obtained using the imputation package `{mice}` (Van Buuren and Groothuis-Oudshoorn 2011). Inferential validity is reached when estimates are both unbiased and have nominal coverage across simulation repetitions ($n_{\text{sim}} = 2000$). To evaluate convergence, we terminate the imputation algorithm after a varying number of iterations ($n_{\text{it}} = 1, 2, ..., 100$, corresponding to τ in each individual imputation run). We differentiate between six different missingness scenarios that are defined in the data generating mechanism (see Table XYZ).

Data generating mechanism

TODO: table captions

In each simulation repetition, we first generate a complete set of $n_{\text{obs}} = 200$ cases, representing person-data in a multiple linear regression problem. The predictor space consists of three multivariately normal random variables,

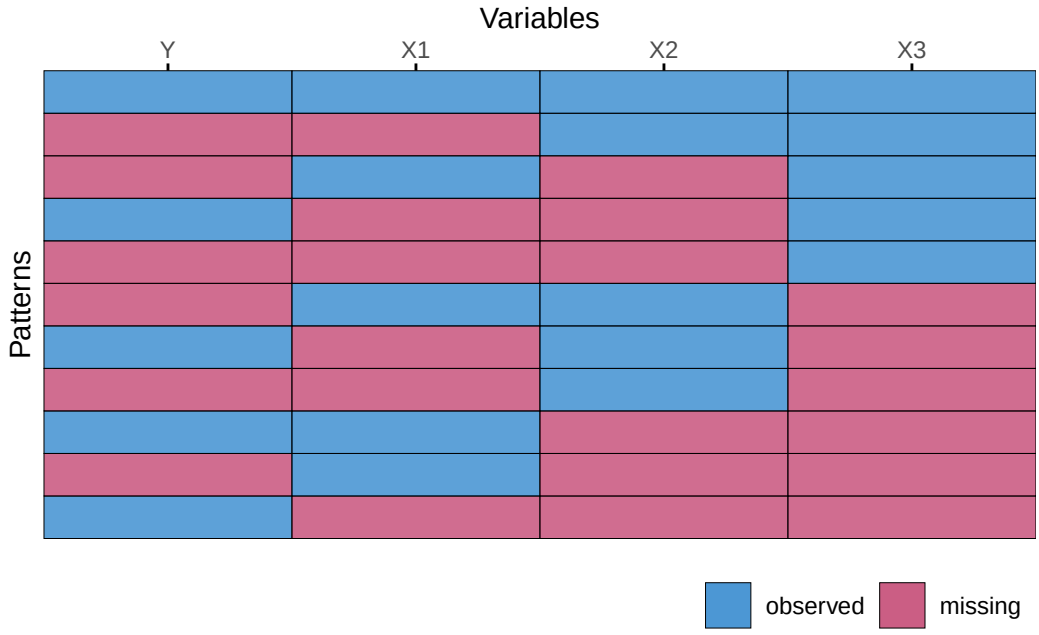
$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \sim \mathcal{N} \left[\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & & \\ 0.5 & 1 & \\ 0.5 & 0.5 & 1 \end{pmatrix} \right].$$

The outcome variable Y is a linear combination of the three predictors, such that for each unit $i = 1, 2, ..., n_{\text{obs}}$,

$$Y_i = X_{1i} + X_{2i} + X_{3i} + \epsilon_i,$$

where $\epsilon \sim \mathcal{N}(0, 1)$. This results in a complete dataset of size $n_{\text{obs}} \times p$, with units $i = 1, 2, ..., n_{\text{obs}}$ and variables $j = Y, X_1, X_2, X_3$). Multivariate normal data are generated using the `mvtnorm` package (Genz and Bretz 2009).

Subsequently, the complete are 'amputed' (i.e., made incomplete) according to six missingness conditions. We use a 2×3 factorial design with two missing data mechanisms and three proportions of incomplete cases, see Table XYZ. We consider all possible multivariate patterns of missingness, as visualized in Figure **??**.



i Missing data mechanism

Missingness mechanisms refer to the probability of being missing for any given entry in a dataset. There are three distinct types of mechanisms, as defined by Rubin (1976). Roughly translated, the probability of being missing may be equal for all entries (MCAR; Missing Completely At Random), may depend on observed information (MAR; Missing At Random), or may depend on *un*observed information, making the missingness non-ignorable (MNAR; Missing Not At Random).

Performance measures		
estimand		metric
	×	
β_0, β_1, β_2, β_3		bias, coverage rate, CI width

The missing data mechanisms under consideration are 'missing completely at random' [MCAR; Rubin (1987)], where the probability to be missing is the same for all $n_{\text{obs}} \times p$ cells in y, and a right-tailed 'missing at random' (MAR) mechanism, where the probability to be missing is a function of the observed data, and higher values are more likely to be missing. The proportion of incomplete cases π_{inc} is set to 25%, 50%, and 75%. We use the 'mice' package [function `mice::ampute()`; Van Buuren and Groothuis-Oudshoorn (2011)] to obtain an incomplete dataset $\{\mathbf{Y}_{\text{obs}}, \mathbf{Y}_{\text{mis}}\}$ for every missingness condition.

Estimands

We impute the missing data five times ($m = 5$) using Bayesian linear regression imputation with `mice` (Van Buuren and Groothuis-Oudshoorn 2011). Bayesian linear regression is a compatible imputation model for the set of variables at hand. Therefore, the imputation algorithm is guaranteed to converge (van Buuren 2007).

On each imputed dataset, we perform multiple linear regression as our analysis of scientific interest. Our estimands are the regression coefficients β,

$$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3,$$

where $\hat{Y}$ is the predicted value of the outcome. We estimate these coefficients using the `lm()` function (R Core Team 2024) in each imputed dataset, and subsequently pool the estimates across the five imputations using Rubin's rules [function `mice::pool()`; Van Buuren and Groothuis-Oudshoorn (2011)].

Performance measures

We evaluate the pooled estimates against their true population values using the performance measures bias, confidence interval width, and coverage rate, as recommended by van Buuren ((2018) § 2.5.2). We calculate bias as $\bar{Q} - Q$. CR is defined as the proportion of simulation repetitions in which the 95% confidence interval (CI) around $\bar{Q}$ covers the true estimand Q. Finally, we inspect CI width (CIW): the difference between the lower and upper bound of the 95% confidence interval around $\bar{Q}$. CIW is of interest because it is a measure of efficiency. Under nominal coverage, short CIs are preferred, since wider CIs indicate lower statistical power.

Non-convergence diagnostics				
identifier		parameter		variable
	×		×	
AC, $\widehat{R}$		chain means, chain variances		Y, X_1, X_2, X_3

Diagnostic methods

We use two non-convergence identifiers—autocorrelation and $\widehat{R}$—to diagnose non-convergence in the imputation algorithm. These two identifiers are applied to two scalar summaries of the state-space of the algorithm—chain means and chain variances. We compute the two identifiers on four different variables: the outcome variable Y, and the three predictors X_1, X_2, and X_3. In total, we apply the two identifiers on two parameters and four variables, resulting in 16 sets of identifier-parameter-variable pairs.

Simulation results

Our results show the non-convergence metrics and the performance of the MICE algorithm under different missingness conditions at a varying number of iterations. For reasons of brevity, we only discuss results for the worst-performing estimates in terms of bias and coverage, X_3. Full results are available from [TODO: add repo DOI]. [TODO: add tables, with MC SEs, to appendix.]

Diagnostic methods

AC

Figures Figure 2 and Figure 3 show the autocorrelations in the chain means and chain variances, respectively.

[TODO: edit!] Autocorrelation in the chain means decreases rapidly in the first few iterations (see Figure 2). The decrease is substantive until $n_{it} \geq 6$. This means that there is some initial trending within chains, but the average imputed value quickly reaches stationarity. These results hold irrespective of the missingness condition.

Autocorrelation in the chain variances show us something similar (see Figure 3). The number of iterations that is required to reach non-improving autocorrelations is somewhat more ambiguous than for chain means, but generally around $n_{it} \geq 10$. We do not observe a systematic difference between missingness conditions here either.

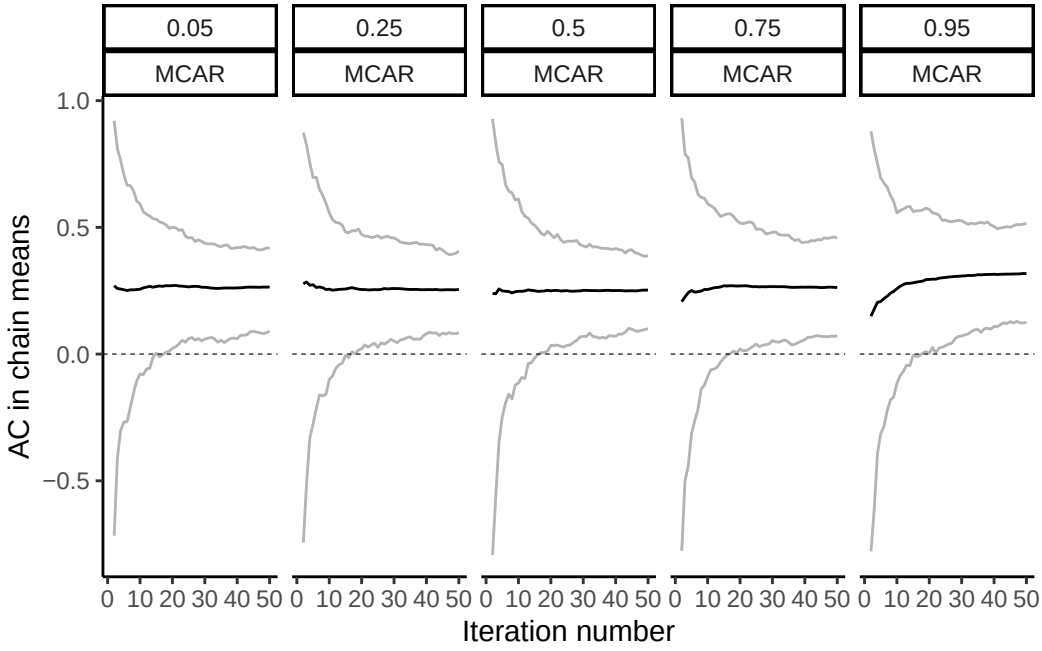


Figure 2: Autocorrelation in the chain means. The dashed line indicates perfect stationarity, $AC = 0$. The solid black line indicates the avarage autocorrelation accross simulation repetitions, with grey lines representing the empirical confidence interval.

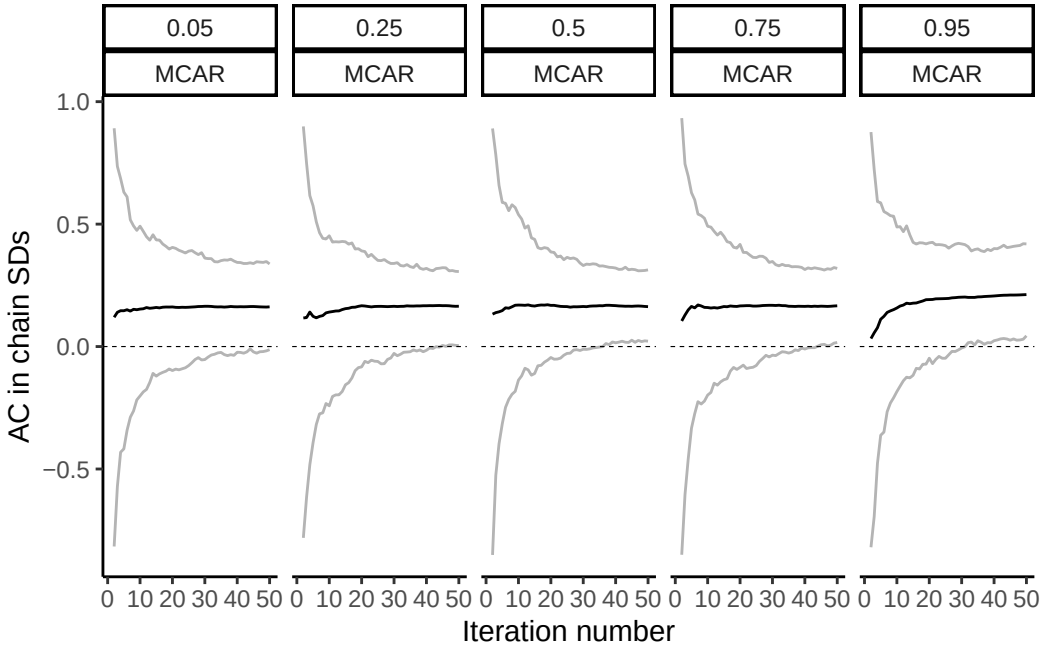


Figure 3: Autocorrelation in the chain variances. The dashed line indicates perfect stationarity, $AC = 0$. The solid black line indicates the avarage autocorrelation accross simulation repetitions, with grey lines representing the empirical confidence interval.

$\widehat{R}$

Figure Figure 4 and Figure 5 show the potential scale reduction factor in the chain means and chain variances respectively.

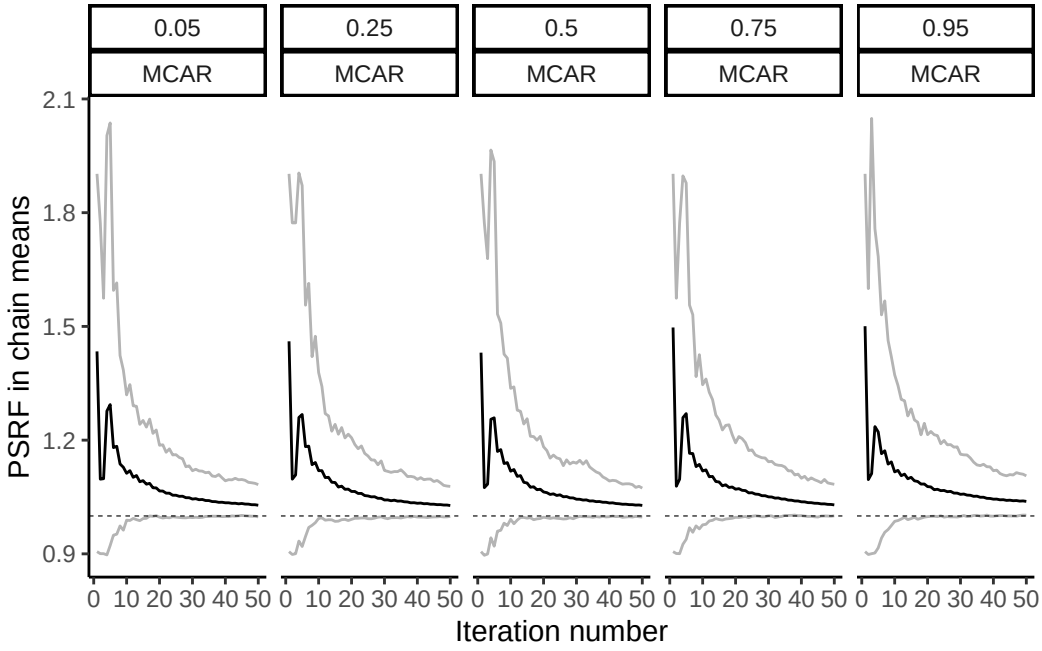


Figure 4: PSRF in the chain means. The dashed line indicates perfect mixing, $\widehat{R} = 0$. The solid black line indicates the avarage PSRF accross simulation repetitions, with grey lines representing the empirical confidence interval.

We observe that $\widehat{R}$-values of the chain means generally decrease as a function of the number of iterations (see Figure 4). An exception to this observation is a steep increase in iterations $3 \leq n_{it} \leq 5$ [TODO: interpret?? due to initialization or is there really more mixing initially??]. After the first couple of iterations, the mixing between chain means generally improves until $n_{it} \geq 30$ to 40. There is no apparent differentiation between the missingness conditions.

The mixing between chain variances mimics the mixing between chain means almost perfectly (see Figure 5. Irrespective of the missingness condition, the $\widehat{R}$-values taper off around $n_{it} \geq 30$.

[TODO: write about: The rhat plots all show some initialization before the fifth iteration: is rhat useful before that??]

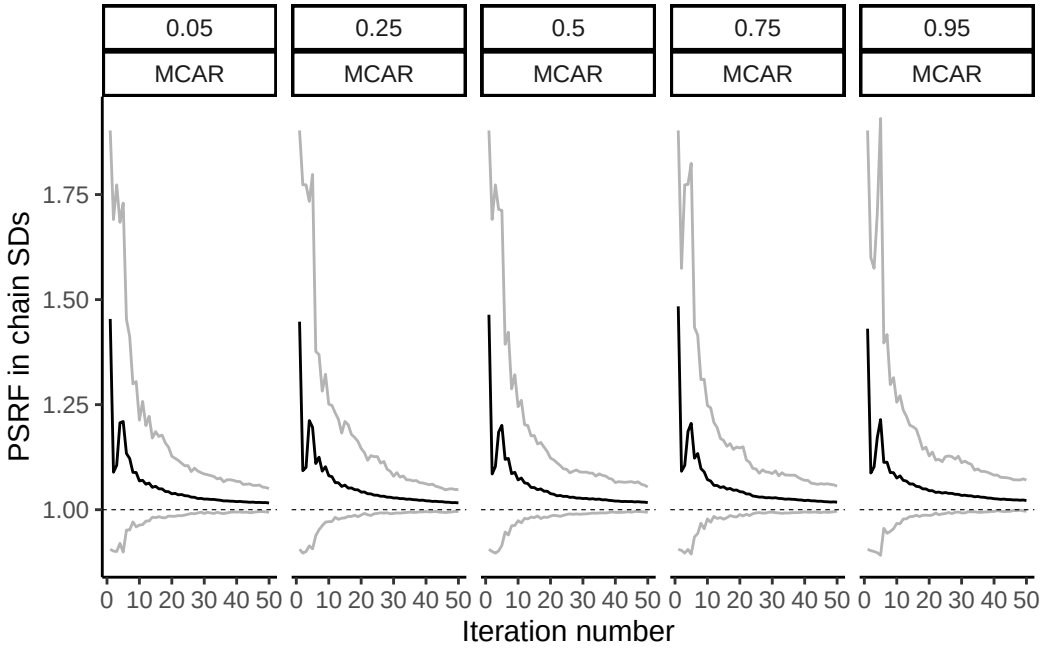


Figure 5: PSRF in the chain variances. The dashed line indicates perfect mixing, $\widehat{R} = 0$. The solid black line indicates the avarage PSRF accross simulation repetitions, with grey lines representing the empirical confidence interval.

Performance measures

In Figures Figure 6 - Figure 8 we show the performance measures: bias in the regression estimate, the empirical coverage rate of the regression estimate and the average confidence interval width of this estimate.

We see that within a few iterations the bias in the regression estimate approaches zero (see Figure 6). When $n_{it} \geq 6$, even the worst-performing conditions (e.g., with a proportion of incomplete cases of 75%) produce stable, non-improving estimates. [TODO: check if regression coefficients are first underestimated because there is less info to estimate the relation??] [TODO: write about: more missingness = more extreme bias, but also steeper decrease over iterations].

Nominal coverage is quickly reached (see Figure 7). After just three iterations, the coverage rates are non-improving in every missingness condition.

The average confidence interval width decreases quickly with every added iteration until a stable plateau is reached (see Figure 8). Depending on the proportion of incomplete cases this takes up-to $n_{it} \geq 9$. [TODO: write about: more missingness = wider CIs, with steeper decrease in CIW, but not stababiling at same value between missingness conditions.]

Summary

Overall, the methods to diagnose non-convergence perform as expected: they indicate more signs of non-convergence in conditions with worse performance in terms of bias and confidence-validity. We observe approximately unbiased estimates after at most seven iterations (for any π_{inc} considered). This implies that the algorithm produces stable, non-improving estimates when $\tau \geq 7$. The number of iterations necessary to obtain approximately unbiased, confidence-valid estimates corresponds to a $\widehat{R}$ threshold of ≈ 1.2.

Discussion

Identifying non-convergence may be a crucial aspect of drawing valid statistical inferences from incomplete data. In this simulation study, we have shown that non-convergence in iterative imputation algorithms does indeed go hand in hand with biased, invalid estimates. Our study , however, found that inferential validity was achieved after five to ten iterations, much earlier than indicated by non-convergence identifiers. While convergence diagnostics kept improving substantially until $\tau \approx 20 - 30$, performance measures did not improve after $\tau = 9$. Of course, it never hurts to iterate longer, but such calculations hardly brought added value.

The main finding of this study is that valid inferences may be obtained much quicker than approximate algorithmic convergence is reached. Under the current specifications, approximately unbiased, confidence-valid estimates were obtained after a maximum of nine iterations.

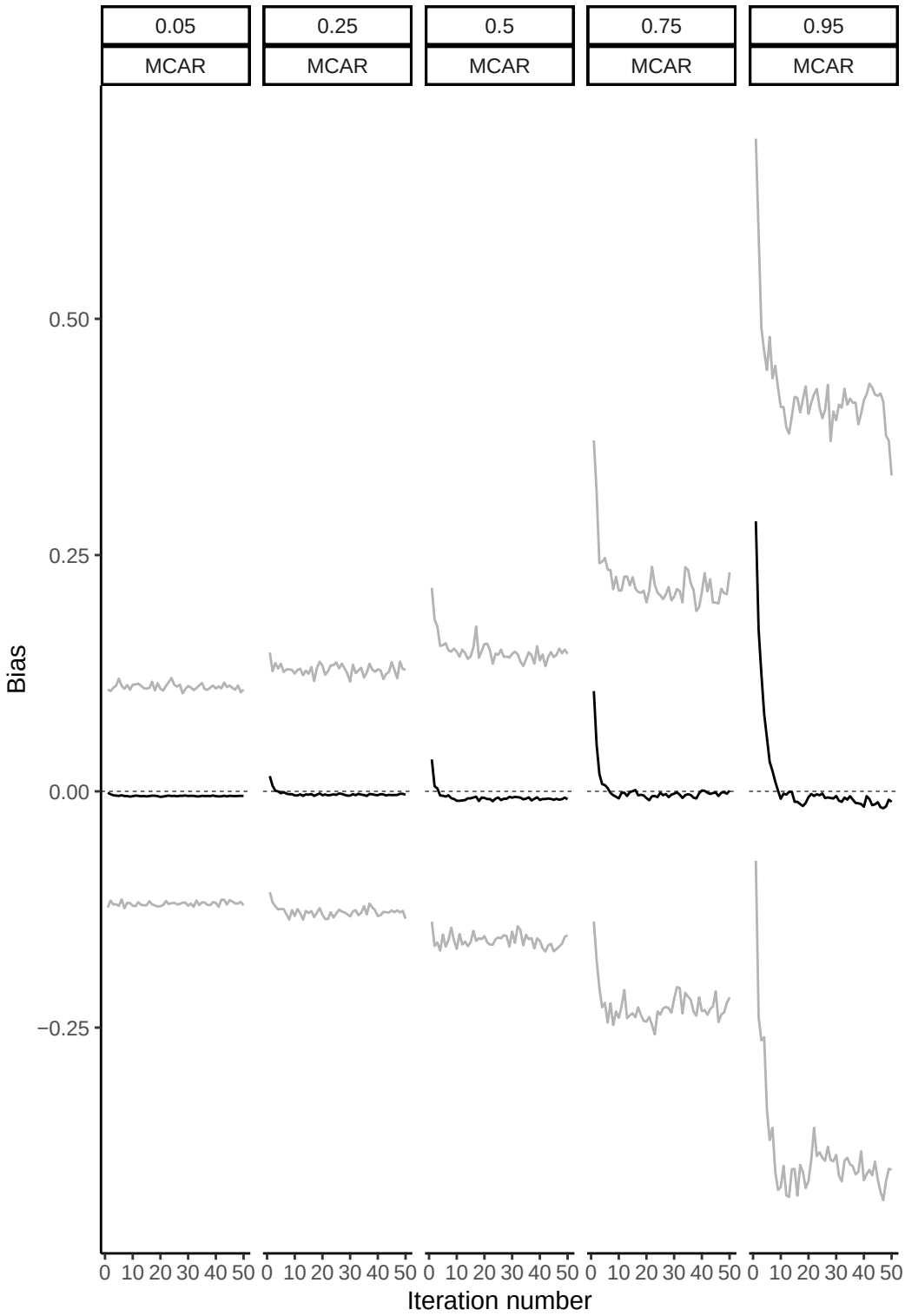


Figure 6: Bias in regression coefficient $\beta = 1$. The dashed line indicates no bias. The solid black line indicates the avarage bias accross simulation repetitions, with grey lines representing the empirical confidence-interval.

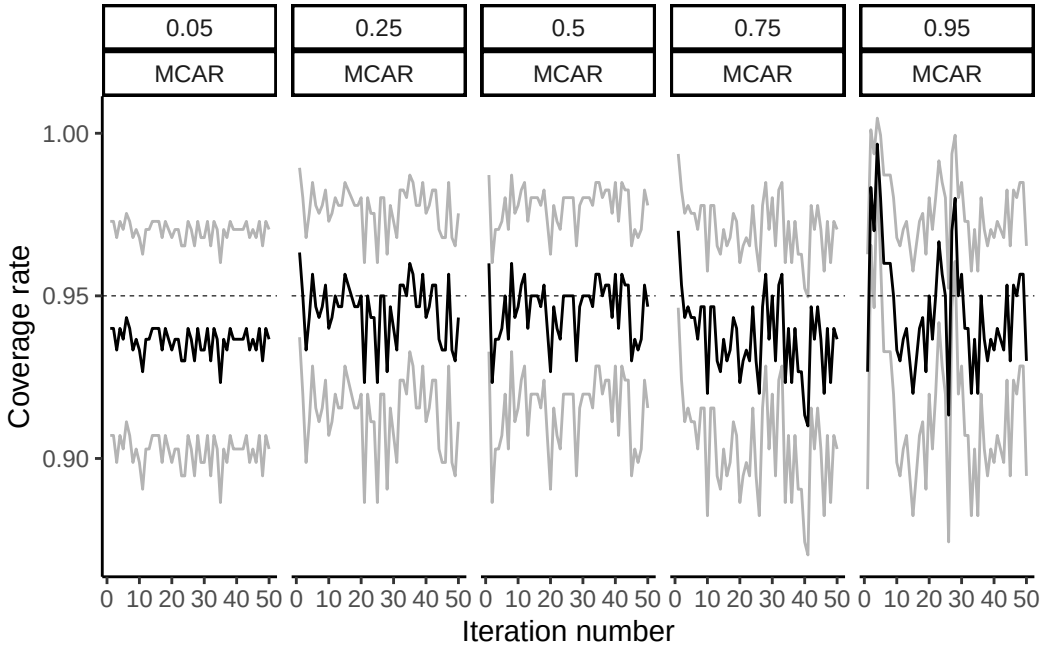


Figure 7: Coverage rate of regression coefficient $\beta = 1$. The dashed line indicates confidence-validity. The solid black line indicates the avarage coverage rate accross simulation repetitions, with grey lines representing the empirical confidence interval.

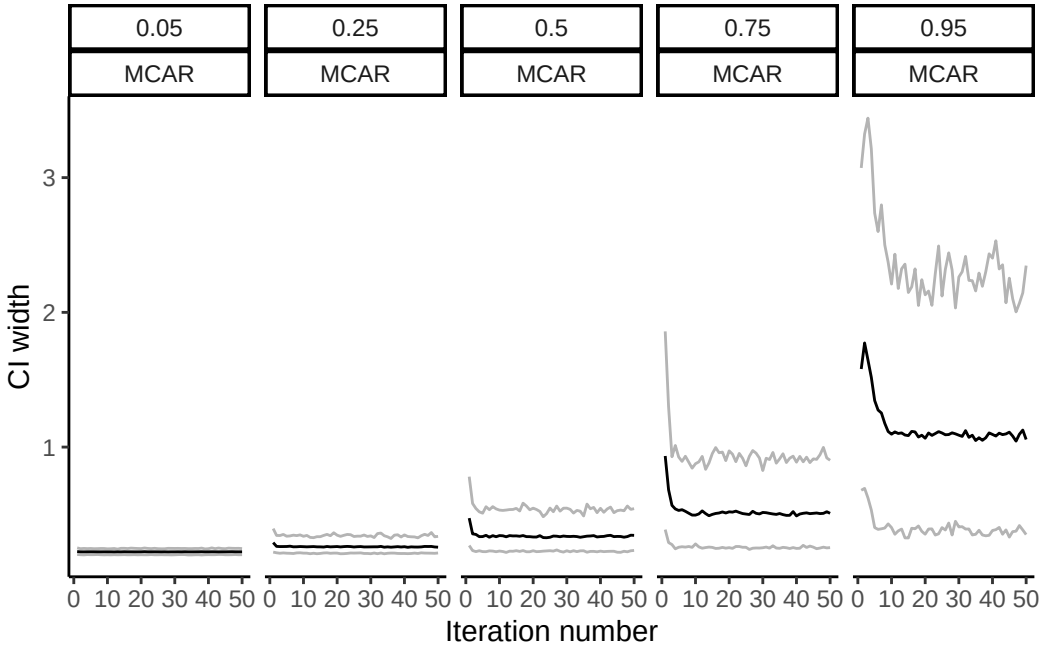


Figure 8: Confidence interval width of regression coefficient $\beta = 1$. The solid black line indicates the avarage CIW accross simulation repetitions, with grey lines representing the empirical confidence interval.

Continued iterations beyond $\tau = 9$ did not yield better estimates. In contrast, $\widehat{R}$-values decreased substantially until 20 to 30 iterations, which implies that mixing in the algorithm still improved with each additional iteration. AC-values only improved until about six iterations, suggesting minimal improvement in trending was obtained beyond $\tau = 6$ in the current simulation set-up.

[TODO: add segway]

[TODO: write about: methodological explanation is that rhat and ac have a lag (too few iterations to inform your statistic) –> will always indicate convergence slower than inferential validity is reached]

The potential scale reduction factor metric assumes over-dispersed starting values of the iterative algorithm. The MICE algorithm does not start in an over-dispersed state. MICE does not rely on starting values for parameters at all: instead, the algorithm is initialized based on a 'zeroth' iteration, where missing values are filled in using a random draw from observed values. Under MCAR, this would typically not yield an over-dispersed state at all, depending on the parameter of interest: means and variances are not affected, regression coefficients and other multivariate parameters start biased downwards (because sampling starting values distorts multivariate structures in the data). Whether this accrues to over-dispersion is questionable. Under MAR (or MNAR), some over-dispersion might occur when the observed data and imputed data differ impactfully. The algorithm should 'escape' the initial state (based on the starting values drawn from observed data alone). Over iterations, all parameters of interest (e.g. means, variances, multivariate estimands) should converge towards a stable state. Moreover, the MICE algorithm is initialized with more information than a typical MCMC algorithm: parameter estimates depend not only on imputed data, but also on observed data that does not change over iterations. Therefore, the initial state of the algorithm is 'too good' to observe a relative decrease in $\widehat{R}$.

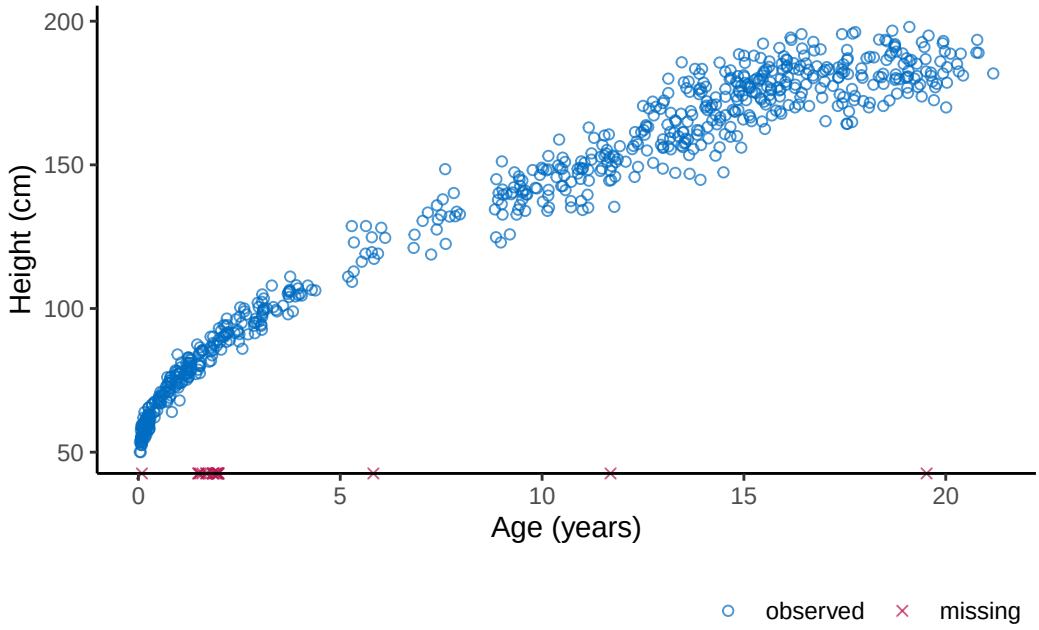
Implications

In short, the validity of iterative imputation stands or falls with algorithmic convergence—or so it's thought. We have shown that iterative imputation algorithms can yield correct outcomes even when a converged state has not yet formally been reached. Any further iterations just burn computational resources without improving the statistical inferences.

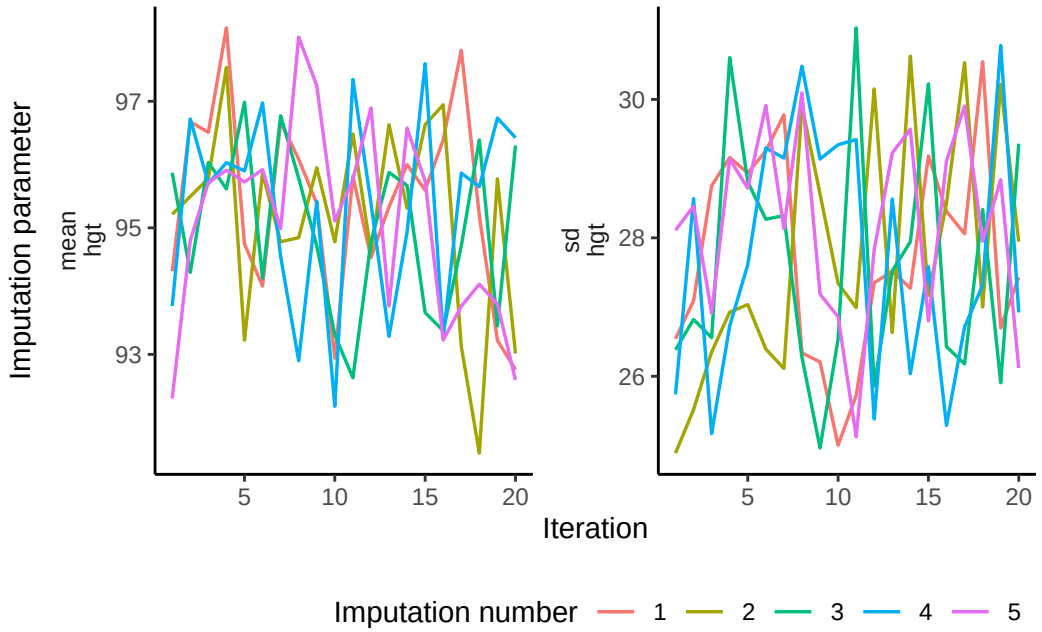
Case Study

We use empirical incomplete data: the `boys` dataset from the `mice` package, which contains health-related data for 748 Dutch boys (Van Buuren and Groothuis-Oudshoorn 2011). Say, we're interested in the relation between children's heights and their respective ages, we could use a linear regression model to predict `hgt` from `age`. However, as figure XYZ shows, the

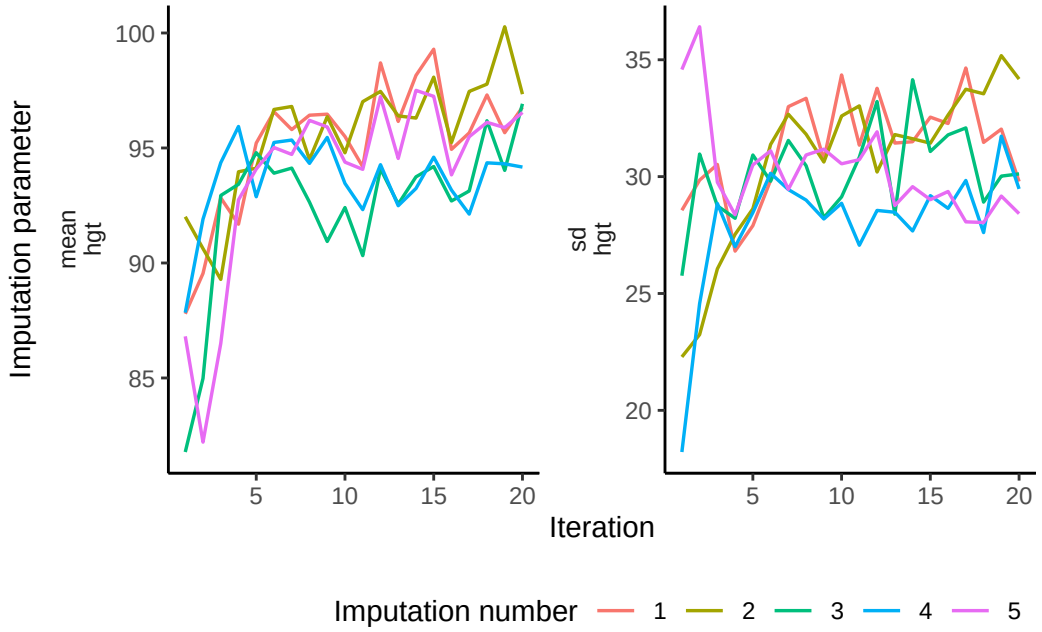
variable `hgt` is not completely observed. To be able to analyze these data, we need to solve the missing data problem first.



The incomplete height variable can be imputed based on auxiliary variables, such as weight. After imputation with `mice`, one would conventionally inspect the trace plots for signs of non-convergence.



A mis-specified imputation model can lead to non-convergence in the imputation algorithm.



- We use real data: the `boys` dataset from the `mice` package

- We are interested in predicting age from the other variables, in particular in the regression coefficient of `hgt`
- We compare non-convergence identified using visual inspection versus rhat in the chain variances, scientific estimate and lambda.
- The figures show results of a `mice` run with 20 iterations but otherwise default settings.

From the traceplot of the chain means (see **?@fig-case**)A) it seems that mixing improves up-to 10 iterations, while trending is only apparent in the first three iterations.

This figure (**?@fig-case**)B) shows that 7 iterations are required before the $\widehat{R}$-values of the chain means drop below the threshold for non-convergence.

The $\widehat{R}$-values for the scientific estimate reaches the threshold much sooner, when $n_{it} = 14$ (see **?@fig-case**)C).

According to the $\widehat{R}$-values with λ as parameter, at least 15 iterations are required (see **?@fig-case**)D).

To illustrate what non-mixing and non-stationarity look like in iterative imputation algorithms, we reproduce an example from van Buuren (2018, § 6.5.2). Figure **??** displays two scenarios from the example. The panel on the left-hand side of the figure shows typical convergence of an iterative imputation algorithm. The right-hand side displays pathological non-convergence, induced by purposefully mis-specifying the imputation model. Each line portrays one imputation chain (i.e., the values of a scalar summary θ across iterations). The θ depicted here is the 'chain mean' of variable j, which is defined as the average of variable j in the m sets of imputations $\dot{\mathbf{Y}}_{\text{imp},\ell}$.

In the typical convergence scenario, the imputation chains intermingle nicely and there is little to no trending. In the non-convergence scenario, there is a lot of trending and some chains do not intermingle. Importantly, the chain means at the last iteration (the imputed values per imputation ℓ) are very different between the two scenarios. The algorithm with the mis-specified model yields imputed values that are on average a factor two larger than those of the typically converged algorithm. It is obvious that non-convergence in this example impacts the distribution of the imputed values per imputation $\dot{\mathbf{Y}}_{\text{imp},\ell}$. This effect may translate into the distribution of the m sets of completed data $\{\mathbf{Y}_{\text{obs}}, \dot{\mathbf{Y}}_{\text{imp},\ell}\}$, which consequently affects the estimated quantities of scientific interest $\hat{Q}_\ell$, and finally the pooled estimate $\bar{Q}$. $\bar{Q}$ is then a biased, invalid estimate of Q. Therefore, it is important to reach algorithmic convergence in iterative imputation algorithms.

Before we evaluate the performance of the non-convergence diagnostics $\widehat{R}$ and AC quantitatively through simulation, we first assess their appropriateness qualitatively. We do this by applying them to the example of pathological non-convergence that we reproduced form Van Buuren (2018). Ideally, the diagnostics are as informative as, or better than visual inspection of the traceplots. The methods should at least indicate worse performance (higher $\widehat{R}$-

and AC-values) for the scenario with pathological non-convergence, compared to the typically converged algorithm.

Each panel in Figure **??** depicts one method to identify non-convergence, applied to the two scenarios from Figure **??**. Panel A consists of the two traceplots that may be evaluated through visual inspection. Panel B shows two versions of the AC: the default calculation with `R` function `stats::acf()` (R Core Team 2024), and manual calculation as the correlation between θ in iteration τ and θ in iteration $t+1$. Panel C displays the traditional computation of $\widehat{R}$ conform Gelman and Rubin (original), and in panel D we see $\widehat{R}$ as computed by implementing Vehtari et al.'s recommendations (adapted).

References

Box, George E. P., Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. 2015. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons.

Brooks, Stephen P., and Andrew Gelman. 1998. "General Methods for Monitoring Convergence of Iterative Simulations." *Journal of Computational and Graphical Statistics* 7 (4): 434–55. https://doi.org/10.1080/10618600.1998.10474787.

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

Cowles, Mary Kathryn, and Bradley P Carlin. 1996. "Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review." *Journal of the American Statistical Association* 91 (434): 883–904.

El Adlouni, Salaheddine, Anne-Catherine Favre, and Bernard Bobée. 2006. "Comparison of Methodologies to Assess the Convergence of Markov Chain Monte Carlo Methods." *Computational Statistics & Data Analysis* 50 (10): 2685–2701. https://doi.org/10.1016/j.csda.2005.04.018.

Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. 2013. *Bayesian Data Analysis*. Philadelphia, PA, United States: CRC Press LLC.

Gelman, Andrew, and Donald B. Rubin. 1992. "Inference from Iterative Simulation Using Multiple Sequences." *Statistical Science* 7 (4): 457–72. https://doi.org/10.1214/ss/1177011136.

Genz, Alan, and Frank Bretz. 2009. *Computation of Multivariate Normal and t Probabilities*. Lecture Notes in Statistics. Heidelberg: Springer-Verlag.

Hoff, Peter D. 2009. *A First Course in Bayesian Statistical Methods*. Springer Texts in Statistics. New York, NY: Springer New York. https://doi.org/10.1007/978-0-387-92407-6.

Lynch, Scott M. 2007. *Introduction to Applied Bayesian Statistics and Estimation for Social Scientists*. Springer Science & Business Media.

Murray, Jared S. 2018. "Multiple Imputation: A Review of Practical and Theoretical Findings." *Statistical Science* 33 (2): 142–59. https://doi.org/10.1214/18-STS644.

R Core Team. 2024. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Raghunathan, Trivellore, and Irina Bondarenko. 2007. "Diagnostics for Multiple Imputations." {{SSRN Scholarly Paper}} ID 1031750. Rochester, NY: Social Science Research Network.

Rubin, Donald B. 1976. "Inference and Missing Data." *Biometrika* 63 (3): 581–92. https://doi.org/10.2307/2335739.

———. 1987. *Multiple Imputation for Nonresponse in Surveys*. Wiley Series in Probability and Mathematical Statistics Applied Probability and Statistics. New York, NY: Wiley.

———. 1996. "Multiple Imputation After 18+ Years." *Journal of the American Statistical Association* 91 (434): 473–89. https://doi.org/10.2307/2291635.

Schafer, Joseph L. 1997. *Analysis of Incomplete Multivariate Data*. Chapman and Hall/CRC.

Takahashi, Masayoshi. 2017. "Statistical Inference in Missing Data by MCMC and Non-MCMC Multiple Imputation Algorithms: Assessing the Effects of Between-Imputation Iterations." *Data Science Journal* 16 (0): 37. https://doi.org/10.5334/dsj-2017-037.

van Buuren, Stef. 2007. "Multiple Imputation of Discrete and Continuous Data by Fully Conditional Specification." *Statistical Methods in Medical Research* 16 (3): 219–42. https://doi.org/10.1177/0962280206074463.

Van Buuren, Stef. 2018. *Flexible Imputation of Missing Data*. Chapman and Hall/CRC.

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

Vehtari, Aki, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. 2021. "Rank-Normalization, Folding, and Localization: An Improved R^ for Assessing Convergence of MCMC." *Bayesian Analysis* -1 (-1): 1–38. https://doi.org/10.1214/20-BA1221.

Zhu, Jian, and Trivellore E. Raghunathan. 2015. "Convergence Properties of a Sequential Regression Multiple Imputation Algorithm." *Journal of the American Statistical Association* 110 (511): 1112–24. https://doi.org/10.1080/01621459.2014.948117.