# 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.

### Aims

- can we diagnose non-convergence quantitatively?
- ~~which parameter should we track?~~
- how many iterations are needed before convergence is reached?
- how many iterations are needed before valid inferences are reached?

## Introduction

Iterative imputation has become the de facto standard to accommodate missing data in social scientific research [ref: e.g., rubin after 18 years paper? murray 2018?]. The technique enables researchers to draw valid inferences when faced with incomplete observations, without resorting to ad hoc solutions such as list-wise deletion, which may bias results. With iterative imputation, every missing cell in an incomplete dataset is imputed (i.e., filled in) algorithmically. And once the dataset is completed, the analysis of scientific interest can be performed. This analysis will yield unbiased and confidence-valid estimates if–and only if–both the missing data problem and the scientific problem are appropriately considered. The validity of this whole process naturally depends on the convergence of the algorithm that was used to generate the imputed values. Yet, there has not been a systematic study on how to evaluate the convergence of iterative imputation algorithms.

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 may produce 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). [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 systematically evaluated on imputation algorithms.

In this study, we explore different methods for identifying non-convergence in iterative imputation algorithms. 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: intrgrate this section] In an empirical study, where $\widehat{R}$ was used to inform the required imputation chain length, it took as many as 50 iterations to overcome the conventional non-convergence threshold $\widehat{R} > 1.2$. Yet, scientific estimates were insensitive to continued iteration from $t = 5$ onward (Lacerda, Ardington, and Leibbrandt 2007). We, therefore, suspect that $\widehat{R}$ may over-estimate signs of non-convergence in iterative imputation algorithms, compared to the validity of estimates. In contrast to this, signs of non-convergence can be under-estimated by $\widehat{R}$, in exceptional cases where the initial values of the algorithm are not appropriately over-dispersed (Brooks and Gelman 1998, 437). In `mice`, initial values are chosen randomly from the observed data, hence we cannot be certain of over-dispersion in the initial values. In practice, we do not expect this to cause problems for identifying non-convergence with $\widehat{R}$.

[TODO: intrgrate this section] If the estimated quantities of scientific interest $\bar{Q}$ are unbiased and confidence-valid estimates of $Q$, we will conclude that the algorithm is sufficiently converged for practical purposes. In contrast to this, we defined *approximate* convergence as the most converged state that the algorithm can obtain. Continued iteration after reaching approximate convergence does not yield better mixing or stationarity (indicated by non-improving $\widehat{R}$- and $AC$-values). Continued iteration after obtaining valid inferences may lead to a more converged state of the algorithm, but not better estimates.

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

## Some notation

Let $y$ denote an $n \times k$ matrix containing the data values on $k$ variables for all $n$ units in a sample. The data value of unit $i$ ($i = 1, 2, \dots, n$) on variable $j$ ($j = 1, 2, \dots, k$) may be either observed or missing. The number of units $i$ with at least one missing value, divided by the total number of units $n$, is called the proportion of incomplete cases $p_{\text{inc}}$ in dataset $y$. The collection of observed data values in $y$ is denoted by $y_{\text{obs}}$; the missing part of $y$ is referred to as $y_{\text{mis}}$. For each datapoint in $y_{\text{mis}}$, we sample $m \times T$ plausible values, where $m$ is the number of imputations ($\ell = 1, 2, \dots, m$) and $T$ is the number of iterations in the imputation algorithm ($t = 1, 2, \dots, T$). The state-space of the algorithm at a certain iteration $t$ may be summarized by a scalar summary $\theta$ (e.g., the average of the imputed values). The collection of $\theta$-values between $t = 1$ (summarizing the state-space of the algorithm at initialization) and $t = T$ (summarizing the state-space for the imputed values) will be referred to as an 'imputation chain'.

## Iterative imputation

Anyone who analyzes person-data may run into a missing data problem. Missing data is not only ubiquitous across science, but treating it can also be a tedious task. If a dataset contains just one incomplete observation, calculations are not defined, and statistical models cannot be fitted to the data. To circumvent this, many statistical packages employ list-wise deletion by default (i.e., ignoring incomplete observations). Unfortunately, this *ad hoc* solution may yield invalid results (Van Buuren 2018). An alternative is to apply imputation. With imputation, we 'fill in' the missing values in an incomplete dataset. Subsequently, the model of scientific interest can be fitted to the completed dataset. By repeating this process several times, a distribution of plausible results may be obtained, which reflects the uncertainty in the data due to missingness. This technique is known as 'multiple imputation' [MI; Rubin (1976)]. MI has proven to be a powerful tool to draw valid inferences from incomplete data under many circumstances (Van Buuren 2018).

Figure **??** provides an overview of the steps involved with MI. The missing part $y_{\text{mis}}$ of an incomplete dataset is imputed $m$ times. This creates $m$ sets of imputed data $\dot{y}_{\text{imp},\ell}$, where $\ell = 1, 2, ..., m$. The imputed data is then combined with the observed data $y_{\text{obs}}$ 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).

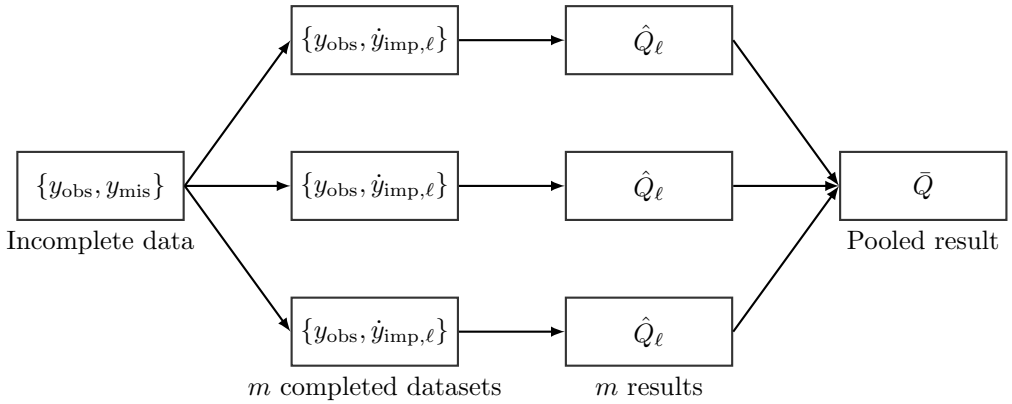



Figure 1: Scheme of the main steps in multiple imputation—from an incomplete dataset, to $m = 3$ multiply imputed datasets, to $m = 3$ estimated quantities of scientific interest $\hat{Q}$, to a single pooled estimate $\bar{Q}$.

A popular method to obtain imputations is to use the 'Multiple Imputation by Chained Equations' algorithm, shorthand 'MICE'(Van Buuren and Groothuis-Oudshoorn 2011). With MICE, imputed values are drawn from the posterior predictive distribution of the missing values. The algorithm is named after its iterative nature: a multivariate distribution is obtained by iterating over a sequence of univariate imputations. Iteration, however, also introduces a potential threat to the validity of the imputations: What if the algorithm has not converged? Are the imputations then to be trusted? And can we rely on the inference obtained on the completed data?

These remain open questions since the convergence properties of iterative imputation algorithms have not been systematically studied (Van Buuren 2018). 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. Iterative imputation algorithms such as MICE 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 (e.g., imputed values) will vary even after convergence (Gelman et al. 2013). 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). Several non-convergence diagnostics exist, but it is not known whether these are appropriate within the imputation framework.

## Algorithmic non-convergence

There are two requirements for convergence of iterative algorithms: mixing and stationarity (Gelman et al. 2013). In iterative imputation algorithms, mixing implies that imputation chains intermingle nicely, and stationarity is characterized by the absence of trending across iterations. If one of the two requirements is not met, we speak of non-convergence. Without mixing, 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.

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 $\theta$ across iterations). The $\theta$ depicted here is the 'chain mean' of variable $j$, which is defined as the average of variable $j$ in the $m$ sets of imputations $\dot{y}_{\text{imp},\ell}$.

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 $\ell$) 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 $\dot{y}_{\text{imp},\ell}$. This effect may translate into the distribution of the $m$ sets of completed data $\{y_{\text{obs}}, \dot{y}_{\text{imp},\ell}\}$, 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.

## Identifying non-convergence

Currently, the recommended practice for evaluating the convergence of the MICE algorithm is through visual inspection. 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 $\theta$ is plotted against the iteration number, as depicted in Figure **??**. The default scalar summaries to inspect for the MICE algorithm are chain means and chain variances. Non-convergence is diagnosed if the imputation chains are not freely intermingled with one another, or if the chains show definite trends (Van Buuren 2018, § 6.5.2).

## 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). 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 both $\widehat{R}$ and $AC$. Other methods are outside the scope of this study because they e.g. 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 2019, Vehtari et al. proposed an updated version of the potential scale reduction factor $\widehat{R}$, originally coined in 1992. The adapted version would be better suited to detect non-mixing in the tails of distributions, and even identify non-stationarity. This version uses three transformations on the scalar summary $\theta$, before computing $\widehat{R}$-values. Namely, rank-normalization, folding, and localization. We follow Vehtari et al.'s formulation (2019, p. 5) to define $\widehat{R}$. Let $m$ be the total number of chains, $T$ the number of iterations per chain (where $T \geq 2$), and $\theta$ the scalar summary of interest. For each chain ($\ell = 1, 2, ..., m$), we estimate the variance of $\theta$, and average these to obtain within-chain variance $W$.

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

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 $\hat{Q}_\ell$). $B$ is defined as the variance of the collection of average $\theta$s per chain:

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

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

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

We can interpret $\widehat{R}$ as potential scale reduction factor since it indicates by how much the variance of $\theta$ 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 $T \to \infty$. 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 $\theta$-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 $\widehat{R}$,

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

We can interpret $AC$-values as a measure of non-stationarity. If there is dependence between subsequent $\theta$-values in imputation chains, $AC$-values are non-zero. Positive $AC$-values occur when $\theta$-values are recurring (i.e., high $\theta$-values are followed by high $\theta$-values, and low $\theta$-values are followed by low $\theta$-values). Recurrence within imputation chains may lead to trending.

Negative $AC$-values occur when $\theta$-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

It is unlikely that iterative algorithms such as MICE will achieve the ideal values of $\widehat{R} = 1$ and $AC = 0$. [TODO: check this! isn't it because of under-dispersion and reliance on observed data?] Because of its convergence to a distribution, the algorithm will show some signs of non-mixing and non-stationarity even in the most converged state. The aim, therefore, is to reach approximate convergence (Gelman et al. 2013). Upon approximate convergence, the imputation chains intermingle such that the only difference between the chains is caused by the randomness induced by the algorithm ($\widehat{R} \gtrapprox 1$), and there is little dependency between subsequent iterations of imputation chains ($AC \gtrapprox 0$). 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 1; the complete script and technical details are available from github.com/hanneoberman/MissingThePoint [TODO: make this a Zenodo DOI]. The number of simulation repetitions $n_{\text{sim}} = 2000$.

**Algorithm 1: 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
```

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

`} } }`

## 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 ($n_{\text{it}} = 1, 2, ..., 100$). We differentiate between six different missingness scenarios that are defined in the data generating mechanism (see Table XYZ).

## Data generating mechanism

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,

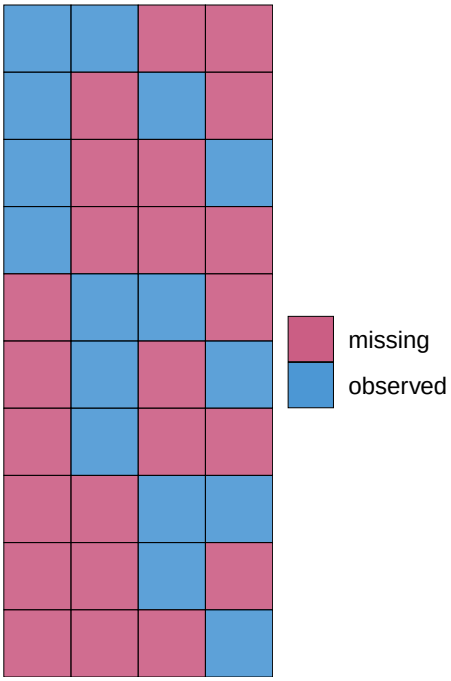
$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \sim \mathcal{N}\left[ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & & \\ 0.5 & 1 & \\ 0.5 & 0.5 & 1 \end{pmatrix} \right].$$

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 $2 \times 3$ 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 **??**.




**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).

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 $\pi_{\text{inc}}$ 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 $\{y_{\text{obs}}, y_{\text{mis}}\}$ for every missingness condition.

[TODO: add left-tailed M(N)AR mechanism, md pattern, and maybe refer to ampute paper or "Dance of…" by Schouten 2018?].

| Performance measures | | |
|---|---|---|
| estimand | | metric |
| | × | |
| $\beta_0$, $\beta_1$, $\beta_2$, $\beta_3$ | | bias, coverage rate, CI width |

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

## Estimands

We impute the missing data five times ($m = 5$) using Bayesian linear regression imputation with `mice` (Van Buuren and Groothuis-Oudshoorn 2011). On each imputed dataset, we perform multiple linear regression as our analysis of scientific interest. Our estimands are the regression coefficients $\beta$,

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

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 percentage 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.

## Diagnostic methods

- non-convergence in iterative algorithms 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
- identifiers are historically focused on either non-mixing between chains or non-stationarity within chains
- the popular non-mixing identifier rhat has recently been updated by Vehtari et al, and should now work for non-stationarity as well
- just to be sure, we also use autocorrelation to quantify trending within chains
- we apply these identifiers to the two parameters that we typically evaluate after imputation using visual inspection: chain means and chain variances

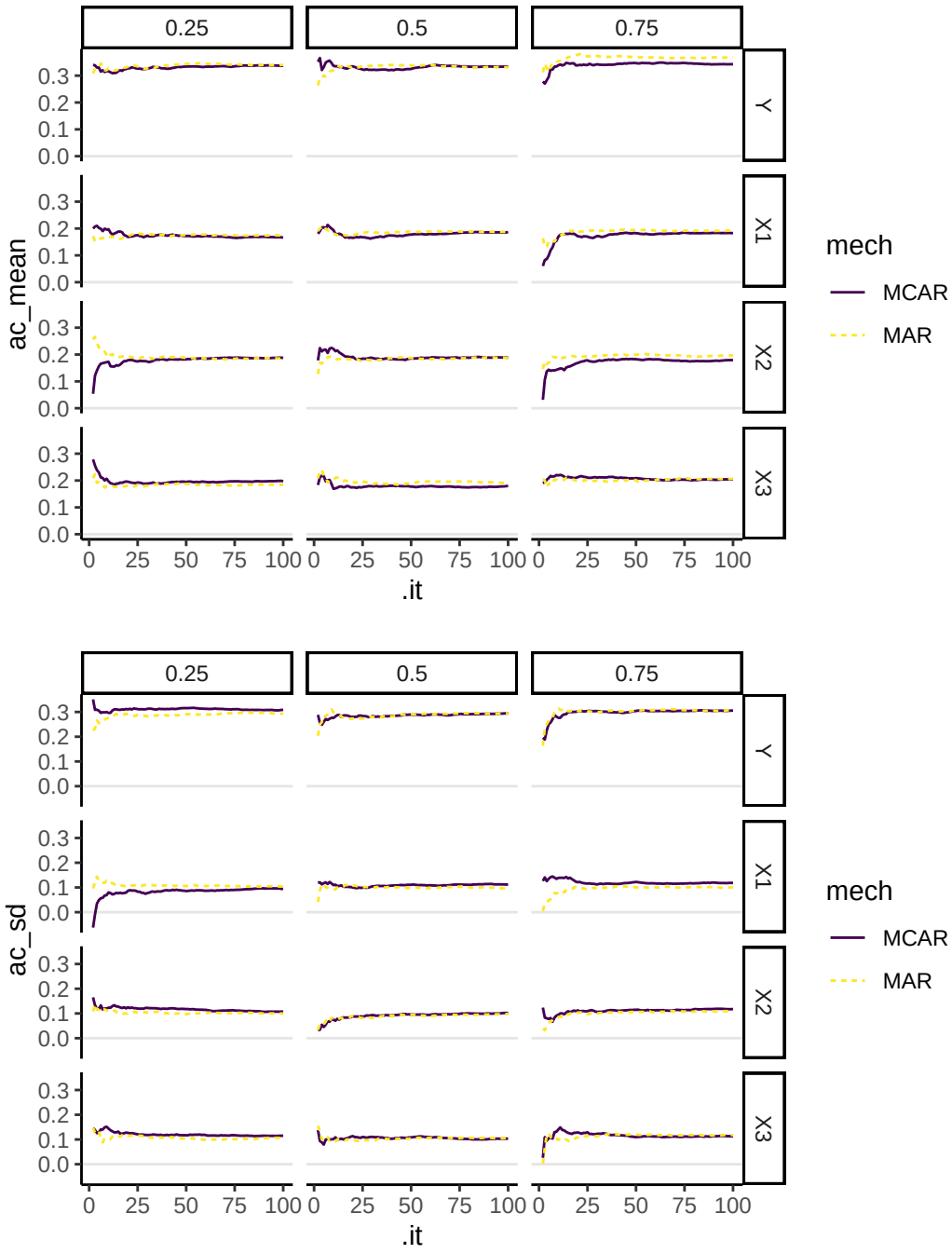
We use two non-convergence identifiers—autocorrelation and $\widehat{R}$—to diagnose non-convergence in the imputation models of the four incomplete variables. For each variable we apply the two identifiers on two parameters—chain means and chain variances—and four variables, resulting in 16 sets of identifier-parameter-variable pairs.

## Simulation results

The following figures display the simulation results for the sixteen diagnostic identifier-parameter-variable pairs, and sixteen performance measure-variable pairs, contrasted to the number of iterations in the imputation algorithm. Within the figures, we split the results according to the missingness conditions [TODO: missingness mechanisms as line types, and proportion of incomplete cases as colors?]. Note that these results are averages of the $n_{sim} = 2000$ simulation repetitions.

### Diagnostic Methods

Figure @ref(fig:ac) shows the autocorrelations in the chain means (panel A) and chain variances (panel B).




Autocorrelation in the chain means decreases rapidly in the first few iterations (see @ref(fig:ac)A). 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 @ref(fig:ac)B). 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.

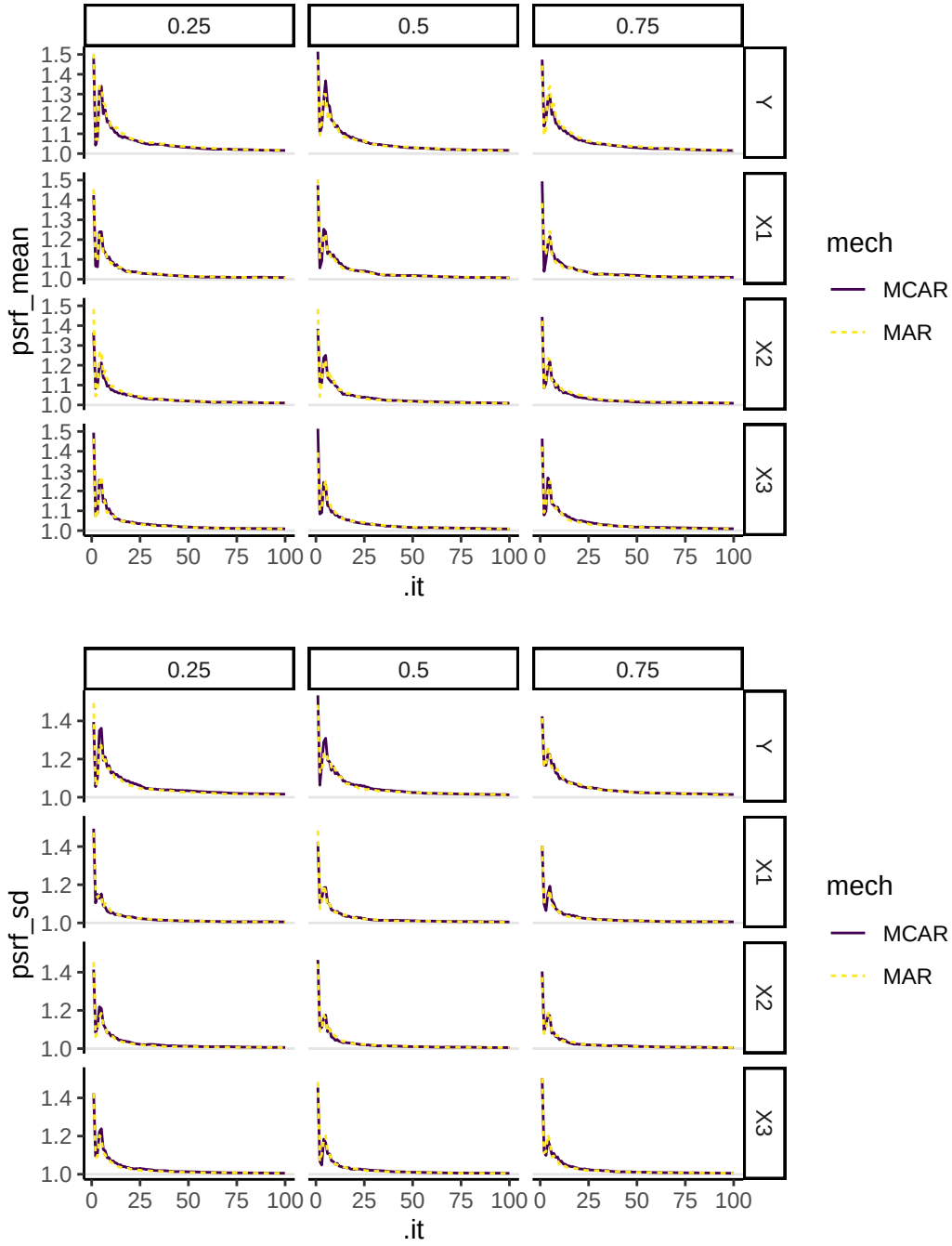
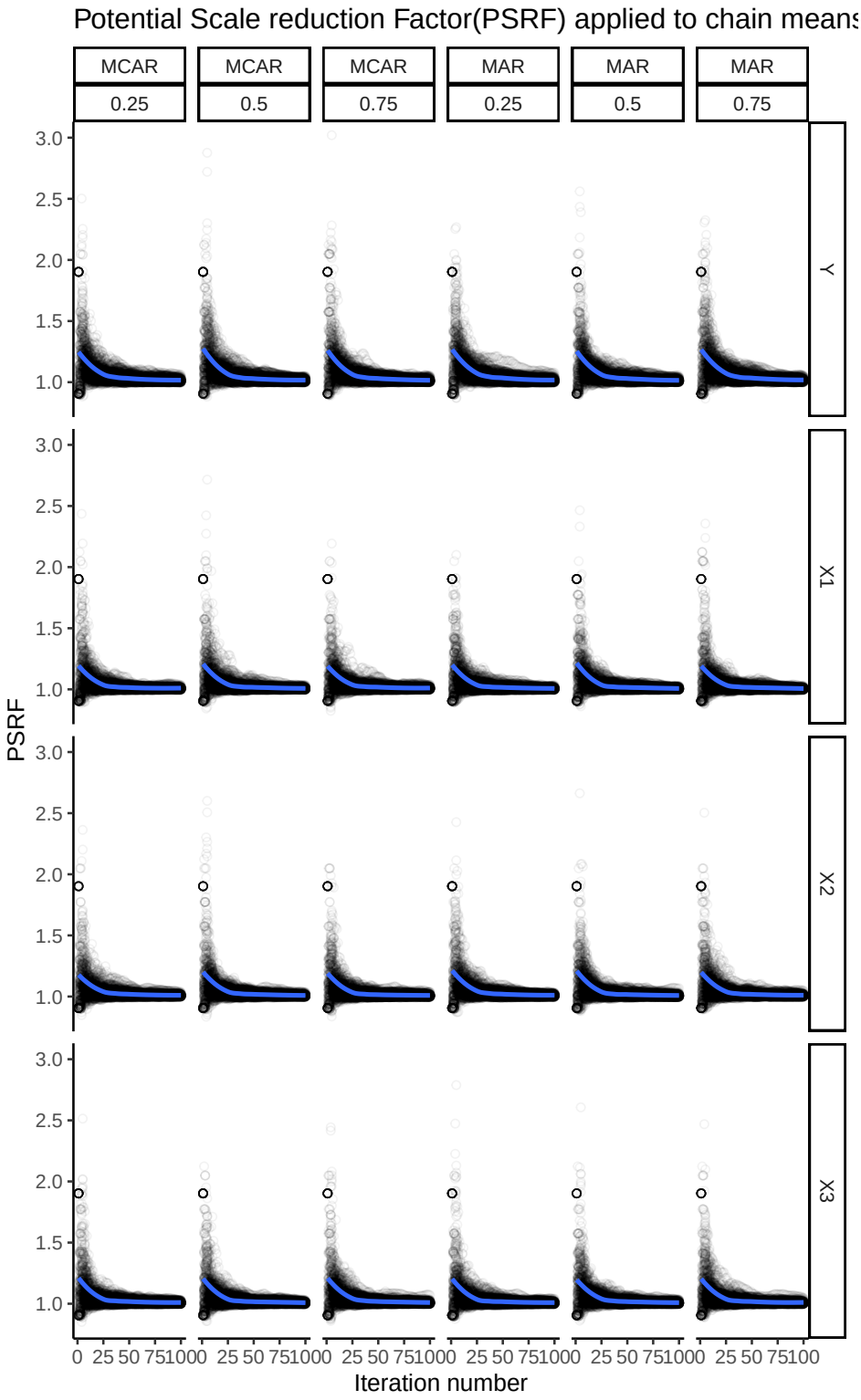
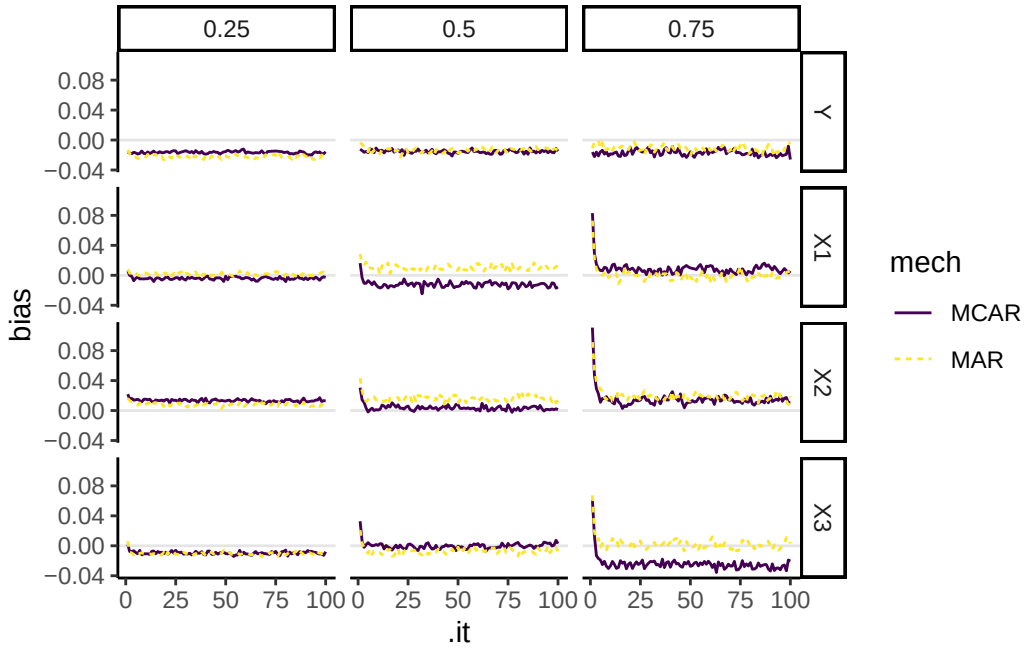


Figure @ref(fig:rh) shows the potential scale reduction factor in the chain means (panel A) and chain variances (panel B).

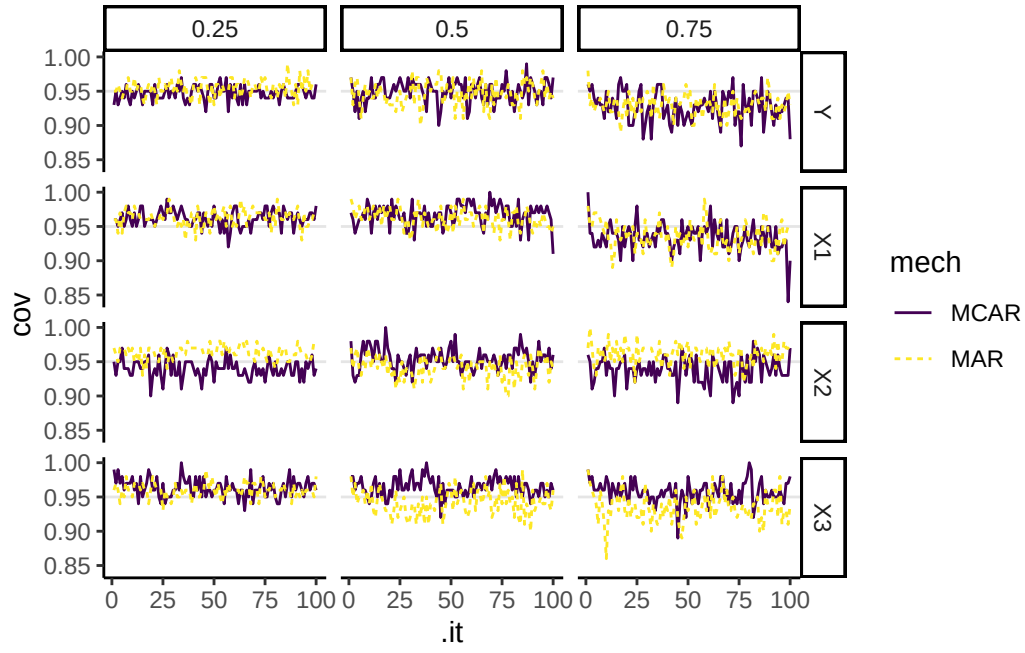
We observe that $\widehat{R}$-values of the chain means generally decrease as a function of the number of iterations (see @ref(fig:rh)A). An exception to this observation is a steep increase in iterations $3 \leq n_{it} \leq 5$ [TODO: interpret?? due to initialization or is there really more mixing initially??]. 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 @ref(fig:rh)B). Irrespective of the missingness condition, the $\widehat{R}$-values taper off around $n_{it} \geq 30$. [The rhat plots all show some initialization before the fifth iteration: is rhat useful before that??]


Potential Scale reduction Factor(PSRF) applied to chain means
MCAR
0.25
MCAR
0.5
MCAR
0.75
MAR
0.25
MAR
0.5
MAR
0.75
Y
X1
X2
X3
PSRF
3.0
2.5
2.0
1.5
1.0
0 25 50 751000 25 50 751000 25 50 751000 25 50 751000 25 50 751000 25 50 75100
Iteration number

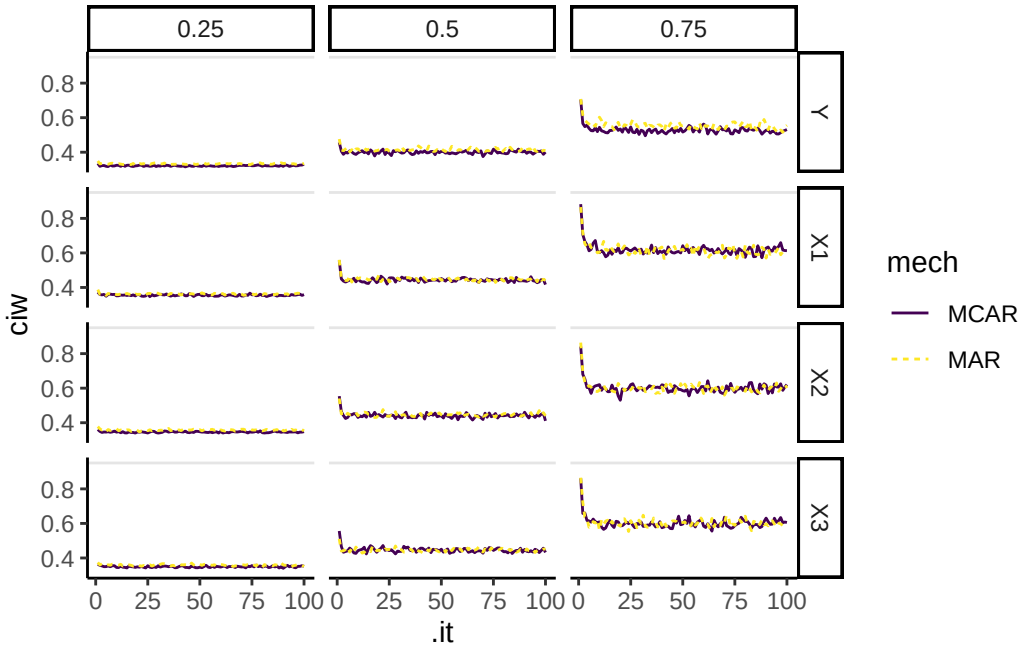
## Performance Measures




- more missingness = more extreme bias, but also steeper decrease over iterations



- no clear trend




- more missingness = wider CIs, with steeper decrease in CIW, but not stababiling at same value between missingness conditions.

In Figure @ref(fig:perf) we show the performance measures: bias in the regression estimate (panel A), the empirical coverage rate of the regression estimate (panel C) and the average confidence interval width of this estimate (panel D).

We see that within a few iterations the bias in the regression estimate approaches zero (see @ref(fig:perf)A). 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 [regression coefficient is underestimated because there is less info to estimate the relation??].

Nominal coverage is quickly reached (see @ref(fig:perf)C). After just three iterations, the coverage rates are non-improving in every missingness condition [but MNAR with 5% incomplete cases does not reach nominal coverage –> due to bias in the estimate in combination with very narrow CI (see CIW!)].

The average confidence interval width decreases quickly with every added iteration until a stable plateau is reached (see @ref(fig:perf)D). Depending on the proportion of incomplete cases this takes up-to $n_{it} \geq 9$.

## Discussion

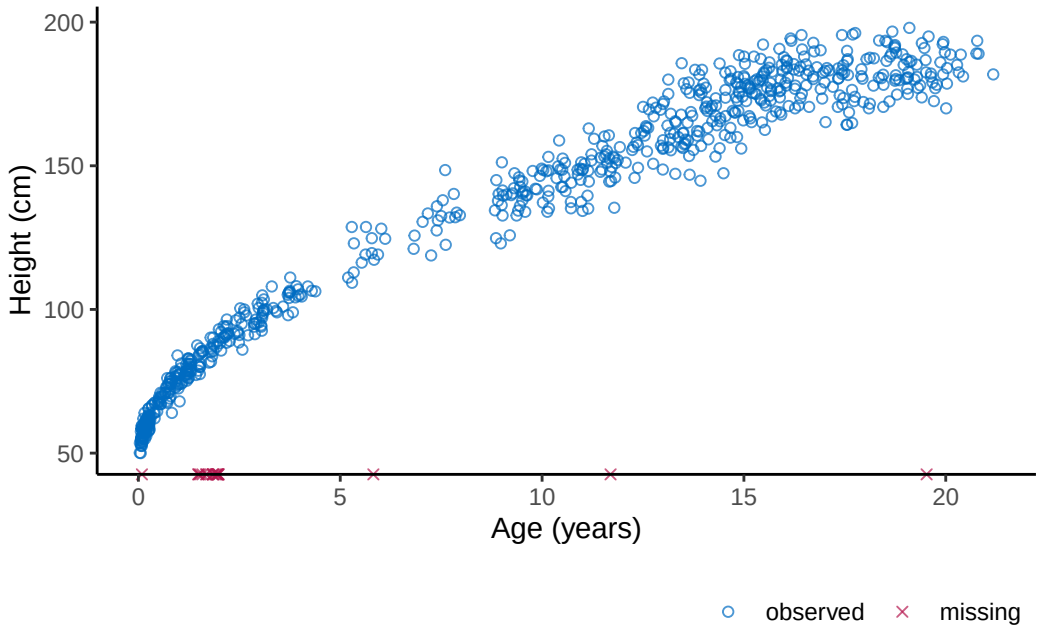
Our study found that—in the cases considered—inferential validity was achieved after five to ten iterations, much earlier than indicated by the non-convergence identifiers. Of course, it never hurts to iterate longer, but such calculations hardly bring added value.

- Convergence diagnostics keep improving substantially until n_it = 20-30
- Performance measures do not improve after n_it = 9
- [methodological explanation is that rhat and ac have a lag (few it to inform your statistic) –> will always indicate convergence slower than inferential validity is reached]

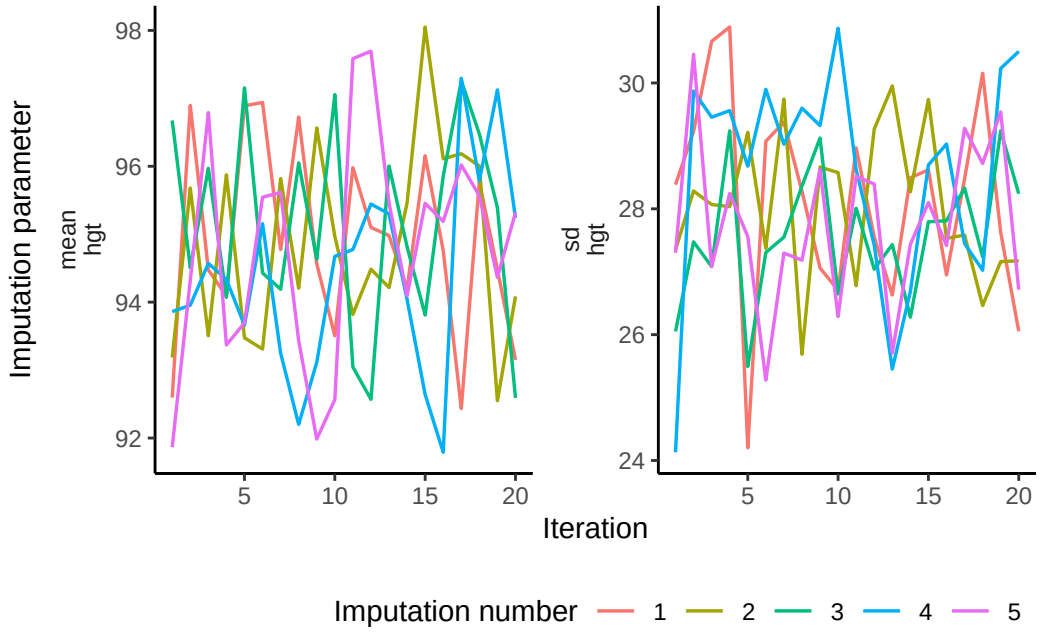
In short, the validity of iterative imputation stands or falls with algorithmic convergence—or so it's thought. We have shown that iterative imputation algorithms can yield correct outcomes even when a converged state has not yet formally been reached. Any further iterations just burn computational resources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

## Case Study

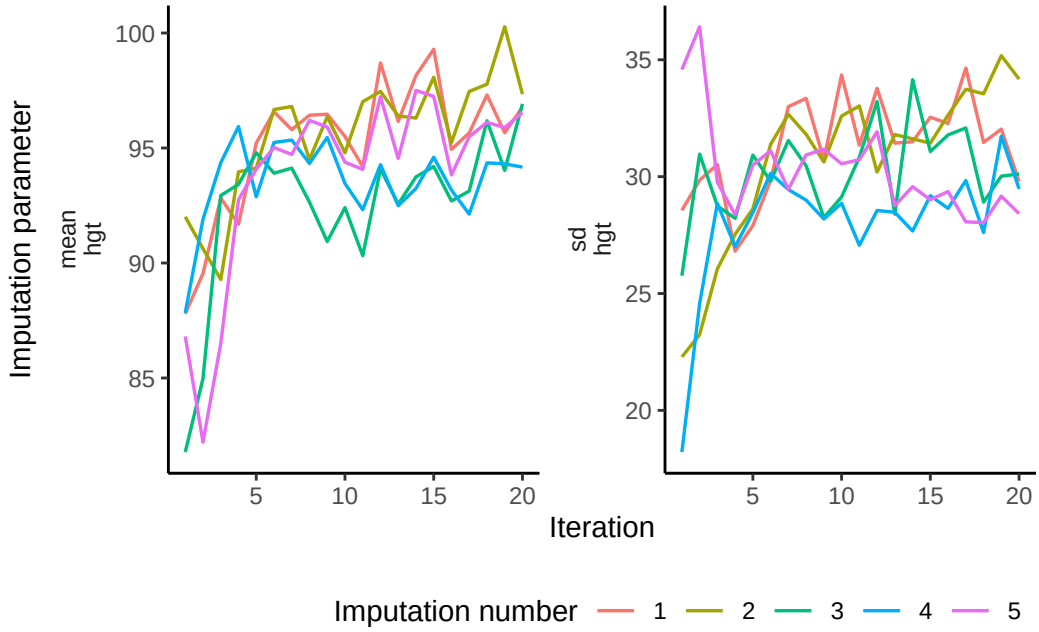
We use empirical incomplete data: the `boys` dataset from the `mice` package, which contains health-related data for 748 Dutch boys (Van Buuren and Groothuis-Oudshoorn 2011). Say, we're interested in the relation between children's heights and their respective ages, we could use a linear regression model 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 @ref(fig:case)A) it seems that mixing improves up-to 10 iterations, while trending is only apparent in the first three iterations.

This figure (@ref(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 @ref(fig:case)C).

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

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 $\theta$ in iteration $t$ and $\theta$ 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).


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.

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.

Lacerda, Miguel, Cally Ardington, and Murray Leibbrandt. 2007. "Sequential Regression Multiple Imputation for Incomplete Multivariate Data Using Markov Chain Monte Carlo." University of Cape Town, South Africa.

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. 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.