Missing The Point

Hanne Oberman

7-9-2020

# Missing The Point: Non-Convergence in Iterative Imputation Algorithms

*Notation in this manuscript:*

* square brackets are comments for myself/stuff I need to review
* bullet points is stuff I need to expand

## Abstract

Iterative imputation is a popular tool to accommodate 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, since there is no consensus on how to evaluate the convergence properties currently. We found that–in the cases considered–inferential validity was 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.

## Intro

Missing data pose a ubiquitous threat to anyone who aims to obtain unbiased, confidence-valid statistical inferences. A popular technique to accommodate missing data is to ‘impute’ (i.e., fill in) any missing values in an incomplete dataset. Imputation procedures like ‘Multiple Imputation by Chained Equations’ (MICE) have proven to be powerful tools to draw valid inference under many missing data circumstances (Rubin, 1987; van Buuren, 2018). To obtain imputations, most imputation software packages rely on iterative algorithms.

With iterative imputation, the validity of the inference depends on the state-space of the algorithm at the final iteration. This 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 using the imputed data? These remain open questions since the convergence properties of iterative imputation algorithms have not been systematically studied (Van Buuren, 2018, 6.5.2). While there is no scientific consensus on how to evaluate the convergence of imputation algorithms (Zhu & Raghunathan, 2015; Takahashi, 2017), the current practice is to visually inspect imputations for signs of non-convergence.

Identifying non-convergence through visual inspection may be undesirable for several reasons: 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, 6.5.2). Therefore, a quantitative diagnostic method to identify non-convergence would be preferred.

In this paper we explore diagnostic methods for iterative imputation algorithms. For reasons of brevity, we focus on the iterative imputation algorithm implemented in the popular mice package (Van Buuren and Groothuis-Oudshoorn 2011) in R (R Core Team 2020). We consider two non-convergence identifiers for iterative algorithms: autocorrelation (conform Lynch, 2007, p. 147) and potential scale reduction factor (conform Vehtari et al., 2019, p. 5). Aside from the usual parameters to monitor—chain means and chain variances—we also investigate convergence in the multivariate state-space of the algorithm.

We propose a novel multivariate parameter to check for non-convergence in iterative algorithms. Our aim is to show which method is the most informative about non-convergence in iterative imputation [explain that method = parameter + diagnostic + interpretation].

## Identifying non-convergence

* Iterative imputation is a sort of MCMC algorithm, so it makes sense to investigate non-convergence in a similar fashion to typical MCMC algorithms.
* What does non-convergence look like? –> non-stationarity + non-mixing in a certain parameter, e.g., chain means.
* What are the consequences? –> bias, under-coverage –> refer to van Buuren (2018) instead of reproducing this
* We consider autocorrelation and rhat, and monitor four parameters: chain means, chain variances, a scientific estimate, and lambda.
* Explain lambda

## Simulation Set-Up

We investigate non-convergence in iterative imputation through model-based simulation in R (version 4.0.2; R Core Team 2020). 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](https://github.com/hanneoberman/MissingThePoint).

**Algorithm 1: simulation set-up (pseudo-code)**

for each simulation repetition (1:1000)  
 1. simulate complete data  
 for each missingness condition (1:9)  
 2. create missingness  
 for each iteration (1:50)  
 3. impute missingness  
 4. perform analysis of scientific interest  
 5. apply non-convergence identifiers  
 6. pool results across imputations  
 7. compute performance measures  
 8. combine outcomes from all iterations  
 9. combine outcomes from all missingness conditions  
10. aggregate outcomes from all simulation repetitions

### Aims

Our aim is to assess the impact of non-convergence on the validity of scientific estimates obtained using mice (Van Buuren and Groothuis-Oudshoorn 2011). Inferential validity is reached when estimates are both unbiased and have nominal coverage across simulation repetitions (). To induce non-convergence we terminate the iterative algorithm at different imputation chain lengths (). We differentiate between nine different missingness scenarios: three missingness mechanisms versus three proportions of incomplete cases [remove here if already defined above/below??].

### Data generating mechanism

Data are generated in each simulation repetition for a complete set of cases (i.e., before inducing missingness). We define three multivariately normal random variables, let

[Add mvtnorm package here??] The complete set is amputed according to nine missingness conditions. We use a factorial design consisting of three missingness mechanisms and three proportions of incomplete cases. [Briefly describe MCAR, MAR, MNAR + 25, 50, 75% of cases incomplete (not using 5% anymore, because according to Vink, n.d., 25% is more representative of missing data problems in the social sciences). And add the ampute function?]

### Estimands

We impute the missing data five times () using Bayesian linear regression imputation with mice (Van Buuren and Groothuis-Oudshoorn 2011). On each imputed dataset, we perform multiple linear regression to predict outcome variable from the other two variables

where is the expected value of the outcome. Our estimands are the regression coefficient and coefficient of determination that we obtain after pooling the regression results across the imputations.

### Methods

We use eight different diagnostic methods to identify non-convergence: a combination of two non-convergence identifiers—autocorrelation and —and four parameters of interest—chain means, chain variances, a scientific estimate, and the novel parameter we propose, .

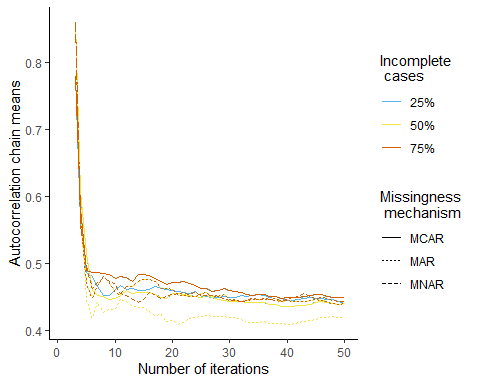
### Performance measures

As recommended by Van Buuren (2018), our performance measures are bias (), average confidence interval width (, where and denote the lower and upper limit of the 95% confidence interval respectively) and empirical coverage rate () of the estimands [or just of the regression estimate?? otherwise add the ciw and cov of r2 too!! and check if definition of ciw is correct].

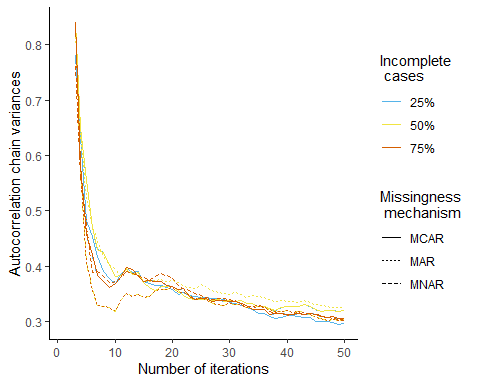
## Simulation Results

The following plots provide an overview of the results for each method and each performance measure, contrasted to the number of iteration in the imputation algorithm. Within the plots, we split the results according to the missingness conditions [missingness mechanisms as line types, and proportion of incomplete cases as colors].

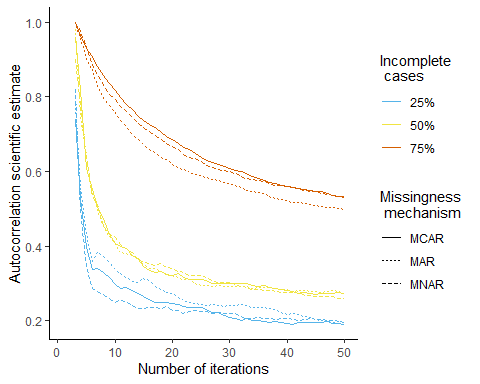
### Diagnostic Methods



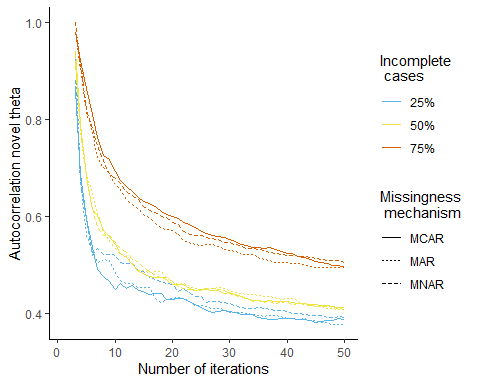
In this plot our diagnostic method is the amount of autocorrelation in the chain means. We can interpret this as the extent to which the average imputed value trends across iterations. We see that autocorrelation in the chain means rapidly decreases until . This means that there is some initial trending within chains, but stationarity does not improve substantively after the first few iterations. These results hold irrespective of the missingness condition.



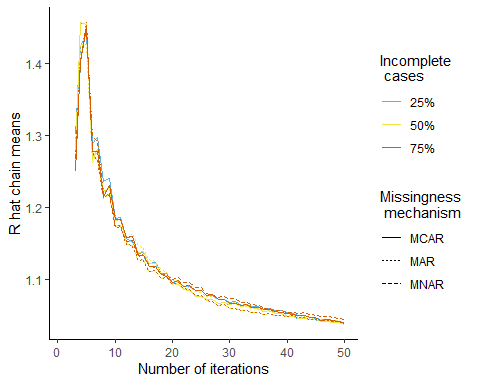
Autocorrelation in the chain variances show us something similar. The number of iterations that is required to reach non-improving autocorrelations is somewhat more ambiguous than for chain means, but generally when . We do not observe a systematic effect of the missingness conditions here either.



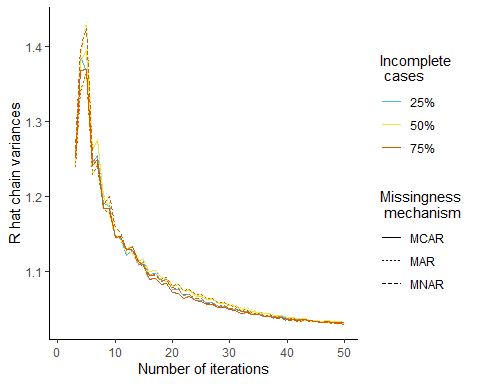
There is more autocorrelation in the scientific estimates than in the univariate parameters (chain means and chain variances). We observe the highest autocorrelations in conditions where 75% of cases are incomplete. Overall, the autocorrelations reach a plateau when to . There is no clear effect of the missingness mechanisms.



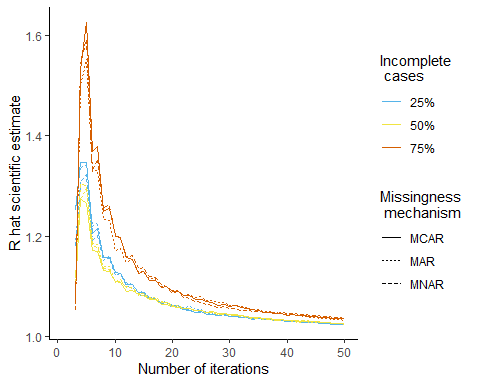
Autocorrelation in the novel parameter exhibit a similar trend to the autocorrelation in the scientific estimates. Trending in this parameter diminishes when .



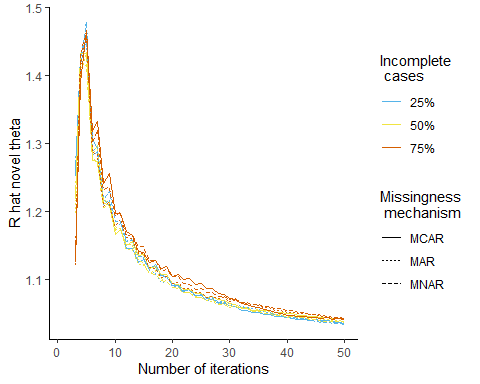
We observe that in the chain means generally decreases as a function of the number of iterations. An exception to this observation is the initial increase when . According to these results, the mixing between chain means generally improves until to . There is no apparent differentiation between the missingness conditions.



The mixing between chain variances mimics the mixing between chain means almost perfectly. Irrespective of the missingness condition, the -values taper off around .



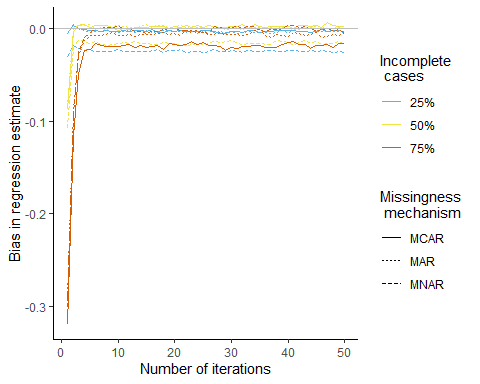
With the scientific estimate as parameter we observe very similar -values again. We do, however, see some differentiation between missingness conditions. Conditions where 75% of the cases are incomplete show more extreme non-mixing. The overall trend remains the same: about 30 iterations are required before mixing stops improving substantially.



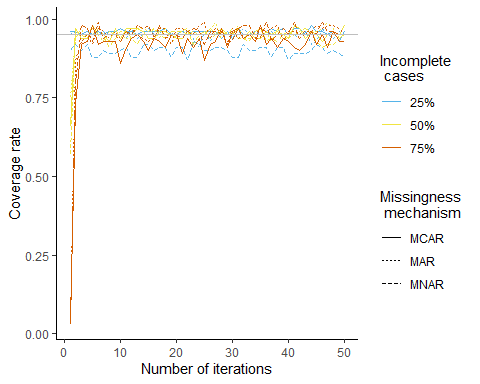
-values in the chains of the novel parameter show a trend similar to the chain means and chain variances. [add something about conditions??]

[These rhat plots all show some initialization before the fifth iteration: is rhat usefull before that??]

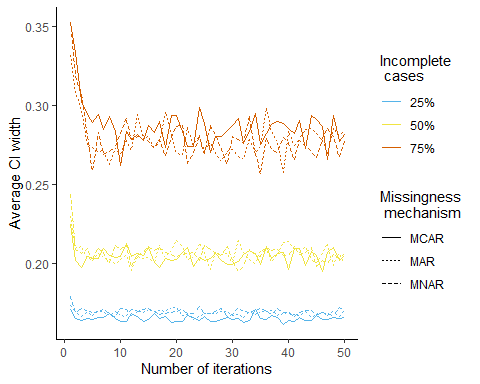
### Performance Measures



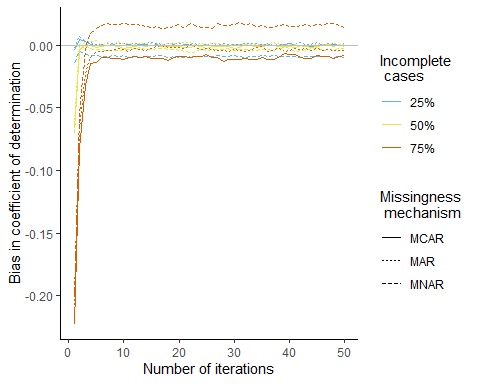
We see that within a few iterations the average bias approaches zero. When , even the worst performing conditions (e.g., with a proportion of incomplete cases of 75%) produce stable, non-improving estimates [regression coefficient is under-estimated because there is less info to estimate the relation??].



Nominal coverage is quickly reached. 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. Depending on the proportion of incomplete cases this takes up-to .

 Equivalent to the bias in the regression estimate, the bias in the coefficient of determination tapers off within a couple of iterations. We observe stable estimates in all conditions when [interpret the over-estimation in MNAR+75% condition?].

## Discussion

* Convergence diagnostics keep improving susbstantially until 20-30 it
* Performance measures do not improve after it = 9
* Univariate thetas may under-estimate non-convergence.
* Determining non-stationarity with lambda is more difficult than with qhat :(

methodological explanation is that rhat and ac have a lag (few it to inform your statistic) –> will always indicate conv slower than the est

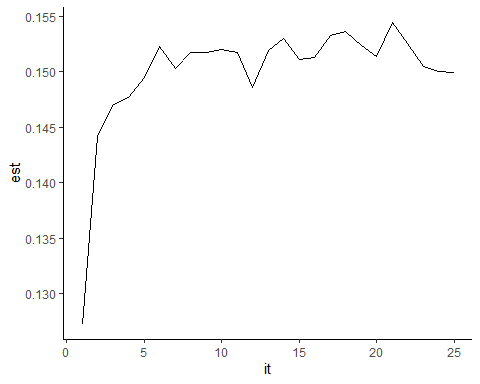
calc rhat and cov for each block of 5 it

disc: conf val is likely to happen much sooner

as soon as we start down-weighting the first few it from the calc, the memory effect would

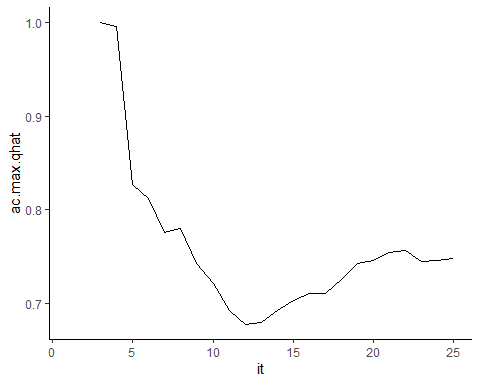
## Example data

# load data  
load("Data/example.Rdata")  
load("Data/example\_mids.Rdata")  
  
# plot each var of example data  
plotfun <- function(v, ...){  
ggplot(example, aes(x = .data[["it"]], y = .data[[v]])) +  
 geom\_line() +  
 theme\_classic()  
}  
  
plotfun(v="est")



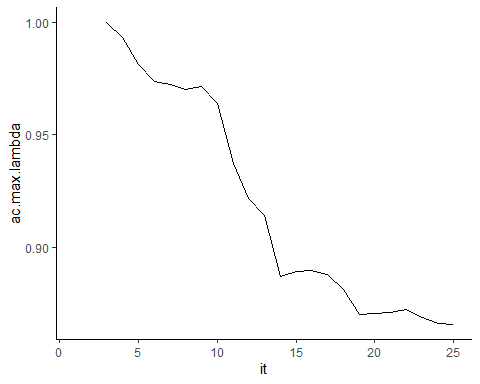
plotfun(v="ac.max.qhat") #+ list(scale\_y\_continuous(limits = c(0.65,1)))

## Warning: Removed 2 row(s) containing missing values (geom\_path).



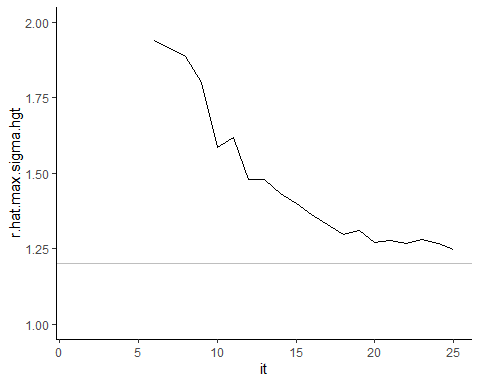
plotfun(v="ac.max.lambda")

## Warning: Removed 2 row(s) containing missing values (geom\_path).



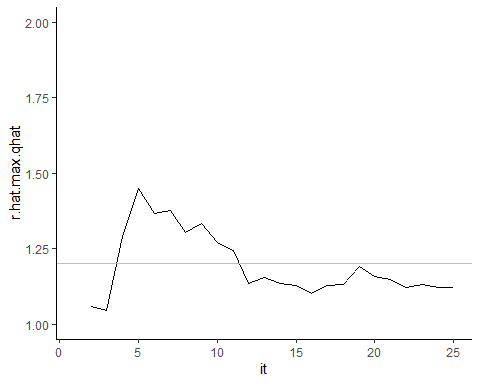
plotfun(v="r.hat.max.sigma.hgt")+list(geom\_hline(yintercept = 1.2, color = "grey"), scale\_y\_continuous(limits = c(1,2)))

## Warning: Removed 5 row(s) containing missing values (geom\_path).



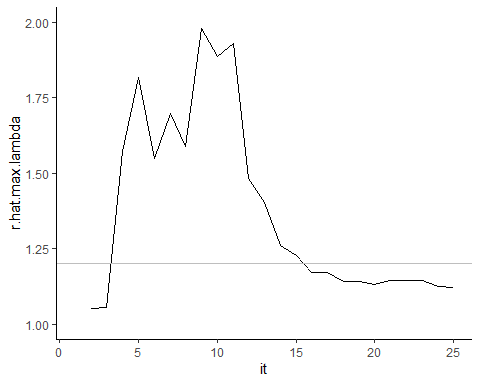
plotfun(v="r.hat.max.qhat")+list(geom\_hline(yintercept = 1.2, color = "grey"), scale\_y\_continuous(limits = c(1,2)))

## Warning: Removed 1 row(s) containing missing values (geom\_path).

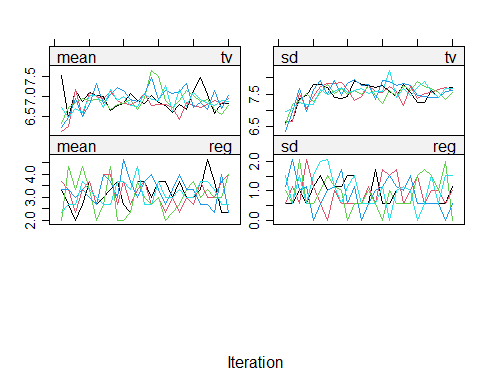
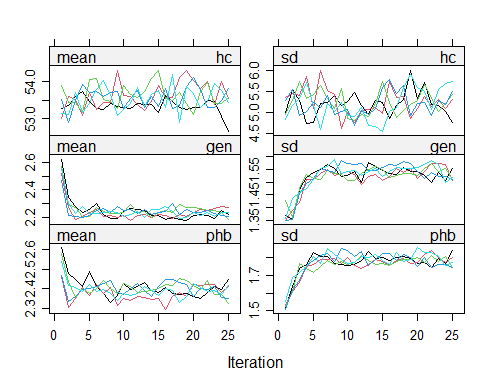
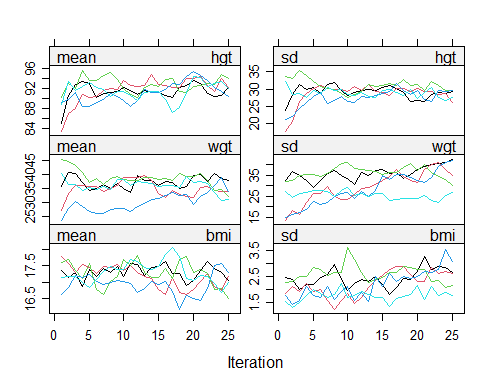


plotfun(v="r.hat.max.lambda")+list(geom\_hline(yintercept = 1.2, color = "grey"), scale\_y\_continuous(limits = c(1,2)))

## Warning: Removed 1 row(s) containing missing values (geom\_path).



plot(mids)



# References (incomplete)

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria.

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>.