# Imputation of incomplete multilevel data

AUTHOR
Hanne Oberman and Gerko Vink

## 1 Introduction

Multilevel data are clustered data structures, for example with students clustered in classes or patients clustered in hospitals. Such hierarchical data structures may be accommodated using multilevel modeling—a statistical technique to capture associations in the data both at the unit-level (level-1) as well as the cluster-level (level-2; REF: Hox). ~~Multilevel modeling is~~ a step-by-step procedure (~~cf.~~ Hox et al., 2017). First, a null-model is fitted to serve as a benchmark. Then, level-1 effects, level-2 effects and cross-level interactions can be added sequentially. The final model will then contain parameters ~~to~~ account for ~~(a~~ combination ~~of)~~ unit-level and cluster-level effects in the multilevel data.

The procedure of multilevel modeling can be complicated, especially in the presence of missing values. If there are missing entries in multilevel data, even the null-model cannot be estimated. That is why statistical software often defaults to ignoring any incomplete rows in the data. This method, known as 'list-wise deletion', allows the analyst to estimate statistical models on the subset of complete cases in the data. However, ignoring incomplete cases assumes that the complete cases are representative of the entire sample. If there is a difference between complete and incomplete cases, complete-case analysis will yield biased results (REF: van Buuren). Moreover, with multilevel data structures, missingness can occur at the unit-level (i.e., incomplete cases), but also at the cluster-level (i.e., incomplete or entirely unobserved cluster-level variables). For example, [missing test results for one student in one school is sporadic unit-level missingness, missing hospital size for all patients in one hospital is systematic cluster-level missingness]. If the observations for a variable are systematically missing in a certain cluster, none of the units in this cluster can be included in the analysis of scientific interest after list-wise deletion. Complete-case analysis then is a wasteful way to accommodate missing data, and may even further bias the analysis results. Hence, the default method to handle missingness in multilevel data is not appropriate.

A more appropriate way to handle missingness in multilevel is to impute (i.e., fill in) every missing data entry multiple times. With multiple imputation, several completed versions of the incomplete data are

created, which can be analyzed as if the data were complete. The resulting statistics can then be pooled according to Rubin's rules (REF: Rubin). Since the analysis results may vary across imputations, pooled estimates will reflect the uncertainty due to non-response. Multiple imputation has been established as a statistically valid all-round method to deal with incomplete data.

This tutorial focuses on one 'flavor' of multiple imputation: multiple imputation by chained equations (MICE) as implemented in the popular `R` package {`mice`}. MICE is an algorithmic approach to fill in missing data entries on a variable-by-variable basis. That means that for every incomplete variable in the data, an imputation model should be chosen. This imputation model should be congenial (i.e., statistically matching) the analysis model one is intending to fit after treating the missingness. There is no encompassing imputation model ('joint model') required. The flexibility that this approach gives, may also be a disadvantage. If it is not clear at the imputation stage what the final analysis model will be at the multilevel modeling stage, it may be difficult to select appropriate imputation models. Each imputation model should therefore be compatible with the broadest multilevel model the analyst is intending to fit after imputation. In this tutorial, we will outline how to build imputation models that are in line with the multilevel structure in the data.

The aim of this tutorial is to provide empirical researchers who are faced with incomplete multilevel data guidance in imputing the missing values in their data with {`mice`}. Missing data pose a wicked, but treatable problem. Combined with multilevel structures (another wicked but treatable problem), the solutions are a lot more difficult. This tutorial will aid empirical researchers in validly handling incomplete multilevel data by means of multiple imputation with the `R` package {`mice`}. Other valid methods (such as `smcfcs`) are outside the scope of this tutorial. No experience with missing data and imputation is required. We will assume basic familiarity with multilevel modeling. The notation will follow Hox et al. (2017). The R package {`lme4`} is used for the analysis of scientific interest. Analysis estimates after imputation are pooled with {`miceadds`}. Visualizations for data exploration and evaluation of the imputations will be performed using {`ggplot2`} and {`ggmice`}.

The requirements to follow along with this tutorial are an incomplete dataset with a multilevel structure, and the conviction that there are no reasons for the missingness that cannot be modeled from the data (ignorability assumption, see Box XYZ). Software-wise, this tutorial requires `R` and the packages `mice`, `lme4`, `ggplot2`, `ggmice` and `miceadds`. As a case study, we will use an adapted version of the popularity data, published by Hox et al. (2017). The original dataset does

not contain any missing values, For the purpose of this tutorial, we have created an 'amputed' (i.e. incomplete) version of the data. The complete version will serve as comparative truth in this tutorial. Both the complete and incomplete data can be downloaded from XYZ. By the end of this tutorial, the reader will be able to build an imputation model for each incomplete variable, evaluate the imputation models, impute the data, evaluate the imputations, and analyze the data, and evaluate the effects of the imputations.

**Ignorability**

Missingness is canonically categorized into three missing data mechanisms. The terminology proposed by Rubin (1987). In this paper we will assume MCAR or MAR. MCAR means that there is no relation between the missingness and observed values. MAR means that there are dependencies between the missingness and the data, but these dependencies can be modeled from the data. The case of MNAR is outside of the scope of this tutorial. We will not cover missingness that is related to unseen parts of the data, e.g. the reason for the missingness depends on the missing value itself.

# 2 Theoretical background

Before we dive into the details and feel sorry for all the holes in our data, it is important to go a few steps back. Because it is not just holes in our data. Most often, there is more data than holes. And more data means more information. In this tutorial we frame your mind to consider the flow of that information in such a way that the multilevel structure is taken into account. In other words, we will show you how to solve for the incomplete data to fit your multilevel model.

Regardless of the type of model we are estimating, we need information for our model to be fit on. Some information we can see, such as the data we have collected. Other sources of information may be unavailable, such as missing values or cases that are not part of our data set. It can easily occur that the available information is not sufficient for our model to arrive at the correct conclusion. This is a common scenario in practice and extends far beyond the domain of mere missing values in data sets (see e.g. Hand 2020). In any case, when the available information is not sufficient, our standard modeling practices fall short and we need to adjust the model to arrive at the correct conclusion.

The concept of adjusting models is by itself quite intuitive. When we need to go from data to answer on a dataset that does not tell us everything, there are but three solutions: We can either append the data with the necessary unavailable information, adjust our modeling such that the

necessary unavailable information is taken into account, or do both. In either way, some model is needed to solve for the incompleteness.

Modeling incomplete data sets, however, is not a trivial task. One needs to carefully make assumptions about the nature of the available and unavailable information and explicitly define the models that connect these two information sources. This tasks becomes increasingly challenging when the complexity of the modeling effort increases. For multilevel data sets this is especially the case, because the complexity of modeling can increase exponentially with every additional level.

In this tutorial we aim to provide a practical guide to imputing incomplete multilevel data. We will use the R package mice to illustrate the imputation process and demonstrate how it connects to a larger ecosystem of imputation and combination methods for incomplete multilevel data.

This tutorial is aimed at researchers and analysts who know their way around multilevel analyses, but lack the skills and expertise of dealing with incomplete sources of information. For them we will gently introduce, explain and demonstrate the necessary methodology and its application in mice. For applied researchers and analysts that are both unfamiliar with multilevel modeling and incomplete data analysis, we will try to be as complete as possible, but we may refer to other sources to complement our tutorial.

## 2.1 Dealing with unavailable information

First and foremost, the best way to deal with unavailable information is to make sure that you have none. This is in itself impossible to verify, as one would need the potentially unavailable information to definitively prove that there is no deviation from the conclusions obtained on the available information. Since this is impossible to do in practice, it is far more convenient to assume some model that relates the observed and unobserved parts in terms of the problem at hand. For such models, there are generally three scenarios that we need to consider.

### 2.1.1 Scenario 1: Missingness is independent of any information

The first scenario is that analysis of the available information by itself yields the correct conclusion. This is an ideal scenario, but it is most likely not the case. Rubin (1987) defines this scenario as **missing completely at random** (MCAR). Hand (2020) defines this - perhaps more intuitively - as *not data dependent*. There is, however, a caveat with using Hand's

terminology, as the term *data dependent* may lull the reader into the false sense of security of assuming that data refers to the available data. While Hand is careful in making this distinction in his book, we would like to explicitly highlight the possibility that more data is always available, and that not considering this data may lead to incorrect conclusions. We therefore favor the explicit use of the term *missingness* in Rubin's terminology and like to stick with Rubin's MCAR in our text.

With MCAR, the missingness in the data is unrelated to any observed information (i.e. data points), nor to any unobserved information. In terms of classical statistics, this would mean that a random and sufficient sample from a population would be randomly incomplete in such a way that analysis of the complete cases would yield a sufficient statistic. You may recognize this as the age-old adage *"the sample is representative of the population"*, but now the sample is both representative and incomplete. The bottom line is that we can ignore the two missingness mechanisms that are at play here:

1. First, the population has been sampled, meaning that the information that we have obtained is incomplete. Random sampling makes this mechanism ignorable and analyzing a sufficiently large sample will - in the limit - yield the same conclusion as analyzing the population itself.
2. Second, the sample is now incomplete. This incompleteness is also assumed to be random and thereby ignorable. In other words, we have merely obtained a smaller sample of the information than originally would have been intended. But analyzing this sample, given that it is still sufficiently large, will still yield the same conclusion as analyzing the population.

In terms of probability we could argue that with MCAR, the probability to be observed is the same for every obtained sequence of information. The complement, however, must also hold: the probability to be unobserved is the same for every unobtained sequence of information. We would like to note that this is a very strong assumption and is often not met in practice. We would also like to note that for structured data this does not mean that every cell has an equal probability to be unobserved. The distribution of random probabilities may change between columns in the data, for example, whilst still being fully random. But as long as this distribution is not related to the sampling, we can still assume that the data is MCAR.

### 2.1.2 Scenario 2: Missingness depends on available information

This assumption is violated when the probability of being unobserved - or

observed, for that matter - is dependent on more than one dimension in the available information. For example when the probability of being unobserved is dependent on the value of another variable in the data, we have a clear and proven deviation from MCAR. This would bring us to the second scenario: **missing at random (MAR)** (Rubin, 1976). With MAR the observed information holds the key to unlocking the missing information. In Hand's terminology, this would be *seen data dependent*. The difficulty here is that - unlike with MCAR, where the missingness mechanism may not be dependent on any observed nor unobserved data - with MAR, it may only be observed data dependent.

The observed data dependency of MAR is a very flexible assumption. It allows us to model the unobserved information based on the observed information. In simple terms, if we define a model for all unobserved information, based on all observed information and combine that model with the observed data model, we'd be able to draw correct conclusions. Bayesians would recognize this as a form of conditional probability, where the probability of the unobserved information is conditional on the observed information. Specifying a good prior for the unobserved information is crucial here, as it will determine the outcome of the model. Any sufficient prior will yield a correct posterior predictive distribution and, hence, a correct conclusion. The flexibility in the Bayesian approach is that it allows for straightforward seperation of the missingness and analysis problems for any scenario that is MAR.

### 2.1.3 Scenario 3: Missingness may also relate to unavailable information

But what if your missingness is related to the data that you cannot see? This would mean that no matter what type of analysis you perform on the observed information, your conclusions would be invalid. This is the third scenario: **missing not at random (MNAR)** (Rubin, 1976). Hand (2020) defines this as *unseen data dependent*. This is the most difficult scenario to deal with, as it is impossible to model the missing information based on the observed information alone and some form of *adjustment* is needed to arrive at the correct conclusion.

> ⓘ **Distinguishing beteen missingness mechanisms**
>
> Everyone who learns about missingness will at some point try to distinguish between the three mechanisms. This is a good exercise, but it is important to note that the distinction is not always clear, nor possible.
>
> Situations in which the missingness mechanism is certain are rare. If you lose one case file during data collection, you can argue that the missingness of that case is random and can therefore be ignored. But how does that relate to

> other missingness in the data set? Alternatively, for some data collection efforts it is known that the missingness is not ignorable. For example, when collecting blood pressure, there is often a clear reason why such information is (not) collected. The absense of blood pressure measurements may therefore indicate that there is no reason to assume and measure abnormal bloodpressure, increasing the likelihood of MNAR. Solving for missing blood pressure would then too much the observations and could easily lead to biased estimation of the effect. If mechanisms can be deduced from the context of a study, such mechanisms are extremely valuable in solving for the incomplete data.
>
> It becomes problematic when people aim to infer mechanisms from the data itself. Take for example a structured data set. If we determine that the missingness in one column depends on the observed values in another column, we may be tempted to conclude that the missingness is MAR. In our evaluation, however, we did not take into account that the missingness could also relate to itself, or that the missingness could relate to some outside source of the data. In any case, if we would model the missingness, it would not be hard to conceptualize another model that relates to unseen data and would have equal fit to the model that only relates to seen data. This has been clearly demonstrated by (molenberghs 2008).
>
> Alternatively, we could argue the same for MCAR. The inability to distinguish both MAR and MCAR from MNAR based on the data alone, renders the many MCAR tests that are available in software futile.

# 3 Methods

In this tutorial, we will use the `R` package {`mice`} for multiple imputation by chained equations, {`lme4`} for multilevel analysis, {`miceadds`} to obtain estimates of multilevel effects after imputation, and {`ggplot2`} and {`ggmice`} for data visualization.

The {`mice`} package provides tools for multivariate imputation by chained equations. The main function, `mice()`, can be run on any dataset in tabular form. To fill in the missing values, the function requires an incomplete dataset. All other arguments, such as the number of imputations `m`, are optional. The `mice()` functions has sensible defaults. Some of these defaults, however, are not applicable to multilevel data. Following the recommendations by van Buuren (2018), each incomplete variable should be imputed using an appropriate imputation model. An imputation model is a combination of an imputation method (the functional form of the model) and relevant imputation model predictors (other variables in the data that are associated with the incomplete variable and/or the missingness indicator of the incomplete variable). In the context of multilevel data, the functional form of the imputation

models will likely need a "2l" component. The "2l" methods take the multilevel structure of the data into account. The `method` argument in the `mice()` function can be used to specify multilevel imputation methods for every incomplete variable in the data. Then the `predictorMatrix` argument can be used to supply imputation model predictors. The clustering variable is denoted with the value `-2`, unit-level associations (or level-1 effects) are denoted with `2`, and cluster-level associations (level-2 effects) are denoted with `1`.

The {`lme4`} package allows for multilevel modeling. The `lmer()` function requires a formula to define the multilevel model and a dataset to estimate the model on. After fitting multilevel models, the {miceadds} package can be used to obtain pooled estimates of multilevel effects.

The 'tidyverse' package {`ggplot2`} is a data visualization package. The `ggplot()` function takes a dataset and mapping instructions. With the family of `geom_*()` functions, different plots can be made. The {ggmice} package extends the functionality of {`ggplot2`}. The `ggmice()` function is equivalent to `ggplot()` except from the way it handles missing or imputed data. The {`ggmice`} package also contains specialized functions for the visualization of e.g. imputation models.

Together, these packages provide functionality treat, analyze, and evaluate incomplete multilevel data. Tabel XYZ gives an overview of the steps that we deem necessary in an imputation workflow for multilevel data. Please note that this list is not exhaustive; we propose these steps as a minimum requirement, accompanied by our `R` function suggestions.

1. Inspect incomplete data

a. Explore variables: determine imputation methods with str() and ggmice(),
b. If necessary, add variables for polynomial terms
c. Explore missingness: evaluate missing data patterns with plot_pattern()
d. Determine imputation model predictors using plot_corr(), plot_flux(), and ggmice()

2. Build imputation models

a. Create methods vector with imputation methods using make.methods()
b. Create predictor matrix with imputation model predictors using make.predictorMatrix()
c. Evaluate imputation models using plot_pred()

3. Run imputation algorithm

a. Impute: fill in the missing values with mice()
b. Inspect convergence: visualize algorithmic convergence with plot_trace()
c. If necessary, add iterations

4. Evaluate imputations

a. Inspect imputed values: assess imputed values with ggmice(), PPC?
b. Analyze imputed data: fit the model of scientific interest with lmer()
c. Evaluate missingness metrics with testEstimates()
d. If necessary, add imputations

5. Interpret results and report

## 3.1 Imputation with `mice`

The `R` package `mice` provides a framework for imputing incomplete data on a variable-by-variable basis. The `mice` function allows users to flexibly specify how many times and under what model the missing data should be imputed. This is reflected in the first four function arguments

```
mice(data, m, method, predictorMatrix, ...)
```

where `data` refers to the incomplete dataset, `m` determines the number of imputations, `method` denotes the functional form of the imputation model per variable and `predictorMatrix` specifies the interrelational dependencies between the variables and imputation models (i.e., the set of predictors to be used for imputing each incomplete variable).

The object supplied as `data` should be tabular (e.g. a `data.frame` with $n$ rows and $p$ variables, with missing values coded as `NA`). For multilevel imputation models, a clustering variable is required. The clustering variable should be a factor or an integer vector, with each level representing a unique cluster. Moreover, the data should be in 'long' format, with each row representing a unique unit-level observation. The multilevel imputation functions implemented in `mice` will not work with 'wide' data, where each row represents a cluster and the columns represent the unit-level observations.

The number of imputations `m` should be determined based on the severity of the missing data problem and the intended analysis model of substantive interest. Van Buuren (2018, § 2.8) suggests using the default `m = 5` for imputation model building, and to increase `m` as required after

initial exploration.

The `method` argument specifies the imputation method to be used for each column in data. If not supplied by the user, `method` defaults to convenient standard methods for single level continuous and categorical data. Since these do not take any clustering or multilevel structures into account, valid imputation of incomplete multilevel data will typically require a user-supplied methods vector. The tables 7.2, 7.3 and 7.4 in van Buuren (2018, § 7.6, § 7.7 and § 7.8, respectively) provide an overview of the available methods to perform univariate multilevel imputation. Note that that in the current version of `mice` (v. 3.16) there are no dedicated multilevel imputation methods available for discrete variables with 3 or more unordered categories.

With the `predictorMatix` argument, `mice` users can define which columns should be used as predictors in each imputation model. The default predictor matrix is a square binary matrix with the variables to be imputed in the rows and the imputation model predictors in the columns. The default `predictorMatrix` will not be suitable for multilevel data. Univariate imputation methods for two-level data use other codes than `0` and `1`. In the predictor matrix, `-2` denotes the cluster variable, a value `1` indicates a fixed effect and a value `2` indicates a random effect. An entry of `3` or `4` indicates that the cluster means of the covariates are added as a predictor to the imputation model with fixed or random effects, respectively.

movie sechie!

# 4 Case study

Prerequisites: incomplete dataset and known multilevel modeling strategy (i.e. the most general analytic model to be applied to imputed data), plus an assumed missingness mechanism. This tutorial assumes M(C)AR, for MNAR see Munoz et al.

The most general analytic model to be applied to imputed data (Hox et al, 2018, p. 17, equation 2.12).

$$\begin{aligned}\text{popularity}_{ij} =& \gamma_{00} + \gamma_{10}\text{gender}_{ij} + \gamma_{20}\text{extraversion}_{ij} + \\ & \gamma_{01}\text{experience}_{j} + \gamma_{21}\text{extraversion}_{ij} \times \text{experience}_{j} + \\ & u_{2j}\text{extraversion}_{ij} + u_{0j} + e_{ij}\end{aligned}$$

Setup `R` environment

```
library(mice)         # for imputation
library(miceadds)     # for imputation
library(micemd)       # for imputation
```

```
library(ggmice)       # for visualization
library(ggplot2)      # for visualization
library(dplyr)        # for data wrangling
library(lme4)         # for multilevel modeling
library(broom)        # for tidying model output
library(broom.mixed)  # for tidying model output
```

Load the complete data.

```
load("data/popular.RData")
```

Load the incomplete data.

```
load("data/popular_MAR.RData")
dat <- popular_MAR
```

Load the data, make sure the variables are correctly formatted (e.g. numeric clustering variable)

## 4.1 Comparative truth

Sequentially fit multilevel models to the complete data to obtain the true parameter estimates.

```
mod_true_singlelevel <- lm(
  popularity_ij ~ 1,
  data = popular
  )
```

```
mod_true_singlelevel |> tidy(conf.int = TRUE)
```

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
|---|---|---|---|---|---|---|
| (Intercept) | 5.08 | 0.031 | 164 | 0 | 5.02 | 5.14 |

```
mod_true_intercept <- lmer(
  popularity_ij ~ (1 | cluster_id),
  data = popular,
  REML = FALSE
  )
```

```
mod_true_intercept |> tidy(conf.int = TRUE)
```

| effect | group | term | estimate | std.error | statistic | conf.low | co |
|---|---|---|---|---|---|---|---|
| fixed | NA | (Intercept) | 5.078 | 0.087 | 58.4 | 4.91 | |
| ran_pars | cluster_id | sd__(Intercept) | 0.833 | NA | NA | NA | |
| ran_pars | Residual | sd__Observation | 1.105 | NA | NA | NA | |

idem + log wt wscore NA

```
mod_true_predictors <- lmer(
  popularity_ij ~ gender_ij + extraversion_ij + experience_j +
  data = popular,
  REML = FALSE
  )
```

```
mod_true_predictors |> tidy(conf.int = TRUE)
```

| effect | group | term | estimate | std.error | statistic | conf.low | co |
|---|---|---|---|---|---|---|---|
| fixed | NA | (Intercept) | 0.809 | 0.169 | 4.79 | 0.478 | |
| fixed | NA | gender_ijgirl | 1.254 | 0.037 | 33.65 | 1.181 | |
| fixed | NA | extraversion_ij | 0.454 | 0.016 | 28.13 | 0.423 | |
| fixed | NA | experience_j | 0.088 | 0.009 | 10.19 | 0.071 | |
| ran_pars | cluster_id | sd__(Intercept) | 0.537 | NA | NA | NA | |
| ran_pars | Residual | sd__Observation | 0.769 | NA | NA | NA | |

idem

```
mod_true_randomslope <- lmer(
  popularity_ij ~ gender_ij + extraversion_ij + experience_j +
    (1  + extraversion_ij | cluster_id),
  data = popular,
  REML = FALSE
  )
```

```
mod_true_randomslope |> tidy(conf.int = TRUE)
```

| effect | group | term | estimate | std.error | statistic |
|---|---|---|---|---|---|
| fixed | NA | (Intercept) | 0.738 | 0.195 | 3.78 |
| fixed | NA | gender_ijgirl | 1.252 | 0.037 | 34.26 |
| fixed | NA | extraversion_ij | 0.453 | 0.024 | 18.49 |
| fixed | NA | experience_j | 0.091 | 0.009 | 10.57 |
| ran_pars | cluster_id | sd__(Intercept) | 1.132 | NA | NA |
| ran_pars | cluster_id | cor__(Intercept).extraversion_ij | -0.886 | NA | NA |

idem

| | | | | | |
|---|---|---|---|---|---|
| ran_pars | cluster_id | sd__extraversion_ij | 0.184 | NA | NA |
| ran_pars | Residual | sd__Observation | 0.743 | NA | NA |

```
mod_true_interaction <- lmer(
  popularity_ij ~ gender_ij + extraversion_ij + experience_j +
    extraversion_ij:experience_j + (1  + extraversion_ij | clust
  data = popular,
  REML = FALSE
  )
```

```
mod_true_interaction |> tidy(conf.int = TRUE)
```

| effect | group | term | estimate | std.error | statistic |
|---|---|---|---|---|---|
| fixed | NA | (Intercept) | -1.207 | 0.269 | -4.49 |
| fixed | NA | gender_ijgirl | 1.241 | 0.036 | 34.27 |
| fixed | NA | extraversion_ij | 0.803 | 0.040 | 20.3 |
| fixed | NA | experience_j | 0.226 | 0.017 | 13.60 |
| fixed | NA | extraversion_ij:experience_j | -0.025 | 0.003 | -9.8 |
| ran_pars | cluster_id | sd__(Intercept) | 0.674 | NA | NA |
| ran_pars | cluster_id | cor__(Intercept).extraversion_ij | -0.627 | NA | NA |
| ran_pars | cluster_id | sd__extraversion_ij | 0.069 | NA | NA |
| ran_pars | Residual | sd__Observation | 0.743 | NA | NA |

Idem

## 4.2 Complete-case analysis

Sequentially fit multilevel models to the incomplete data to obtain the parameter estimates after list-wise deletion.

```
mod_cca_singlelevel <- lm(
  popularity_ij ~ 1,
  data = dat
  )
```

```
mod_cca_singlelevel |> tidy(conf.int = TRUE)
```

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
|---|---|---|---|---|---|---|
| (Intercept) | 5.06 | 0.033 | 155 | 0 | 4.99 | 5.12 |

Idem + referee acn truth

```
mod_cca_intercept <- lmer(
  popularity_ij ~ (1 | cluster_id),
  data = dat,
  REML = FALSE
  )
```

```
mod_cca_intercept |> tidy(conf.int = TRUE)
```

| effect | group | term | estimate | std.error | statistic | conf.low | co |
|---|---|---|---|---|---|---|---|
| fixed | NA | (Intercept) | 5.058 | 0.088 | 57.6 | 4.89 | |
| ran_pars | cluster_id | sd__(Intercept) | 0.839 | NA | NA | NA | |
| ran_pars | Residual | sd__Observation | 1.103 | NA | NA | NA | |

```
mod_cca_predictors <- lmer(
  popularity_ij ~ gender_ij + extraversion_ij + experience_j +
    (1 | cluster_id),
  data = dat,
  REML = FALSE
  )
```

```
mod_cca_predictors |> tidy(conf.int = TRUE)
```

| effect | group | term | estimate | std.error | statistic | conf.low | co |
|---|---|---|---|---|---|---|---|
| fixed | NA | (Intercept) | -0.505 | 0.190 | -2.66 | -0.878 | |
| fixed | NA | gender_ij | 1.230 | 0.043 | 28.32 | 1.145 | |
| fixed | NA | extraversion_ij | 0.460 | 0.019 | 23.86 | 0.422 | |
| fixed | NA | experience_j | 0.090 | 0.009 | 9.96 | 0.072 | |
| ran_pars | cluster_id | sd__(Intercept) | 0.549 | NA | NA | NA | |
| ran_pars | Residual | sd__Observation | 0.769 | NA | NA | NA | |

```
mod_cca_randomslope <- lmer(
  popularity_ij ~ gender_ij + extraversion_ij + experience_j +
    (1  + extraversion_ij | cluster_id),
  data = dat,
  REML = FALSE
  )
```

```
Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
control$checkConv, :
Model failed to converge with max|grad| = 0.00472956 (tol =
0.002, component 1)
```

```
mod_cca_randomslope |> tidy(conf.int = TRUE)
```

| effect | group | term | estimate | std.error | statistic |
|---|---|---|---|---|---|
| fixed | NA | (Intercept) | -0.562 | 0.209 | -2.69 |
| fixed | NA | gender_ij | 1.227 | 0.043 | 28.6 |
| fixed | NA | extraversion_ij | 0.447 | 0.026 | 17.03 |
| fixed | NA | experience_j | 0.096 | 0.009 | 10.76 |
| ran_pars | cluster_id | sd__(Intercept) | 1.037 | NA | NA |
| ran_pars | cluster_id | cor__(Intercept).extraversion_ij | -0.858 | NA | NA |
| ran_pars | cluster_id | sd__extraversion_ij | 0.175 | NA | NA |
| ran_pars | Residual | sd__Observation | 0.747 | NA | NA |

Idem

```
mod_cca_interaction <- lmer(
  popularity_ij ~ gender_ij + extraversion_ij + experience_j +
    extraversion_ij:experience_j + (1  + extraversion_ij | clus
  data = dat,
  REML = FALSE
  )
```

```
mod_cca_interaction |> tidy(conf.int = TRUE)
```

| effect | group | term | estimate | std.error | statistic |
|---|---|---|---|---|---|
| fixed | NA | (Intercept) | -2.119 | 0.308 | -6.87 |
| fixed | NA | gender_ij | 1.221 | 0.043 | 28.67 |
| fixed | NA | extraversion_ij | 0.742 | 0.048 | 15.37 |
| fixed | NA | experience_j | 0.204 | 0.019 | 10.70 |
| fixed | NA | extraversion_ij:experience_j | -0.021 | 0.003 | -6.64 |
| ran_pars | cluster_id | sd__(Intercept) | 0.717 | NA | NA |
| ran_pars | cluster_id | cor__(Intercept).extraversion_ij | -0.665 | NA | NA |
| ran_pars | cluster_id | sd__extraversion_ij | 0.091 | NA | NA |
| ran_pars | Residual | sd__Observation | 0.749 | NA | NA |

Idem

# 4.3 Inspect incomplete data

## 4.3.1 Explore observed data

Assess the structure of the data. waarom doe je dit? leg uit dat het crucial is.

```
str(dat)
```

```
'data.frame':   2000 obs. of  7 variables:
 $ unit_id        : num  1 2 3 4 5 6 7 8 9 10 ...
 $ cluster_id     : num  1 1 1 1 1 1 1 1 1 1 ...
 $ popularity_ij  : num  6.3 4.9 5.3 4.7 6 NA 5.9 4.2 5.2 3.9
...
 $ gender_ij      : num  2 1 2 2 2 1 1 1 1 1 ...
 $ extraversion_ij: num  5 7 4 3 NA NA 5 4 5 5 ...
 $ experience_j   : num  24 NA 24 24 24 24 24 24 24 24 ...
 $ assessment_ij  : num  6 NA 6 5 6 NA 5 5 5 3 ...
```

Print the first few rows of the data.

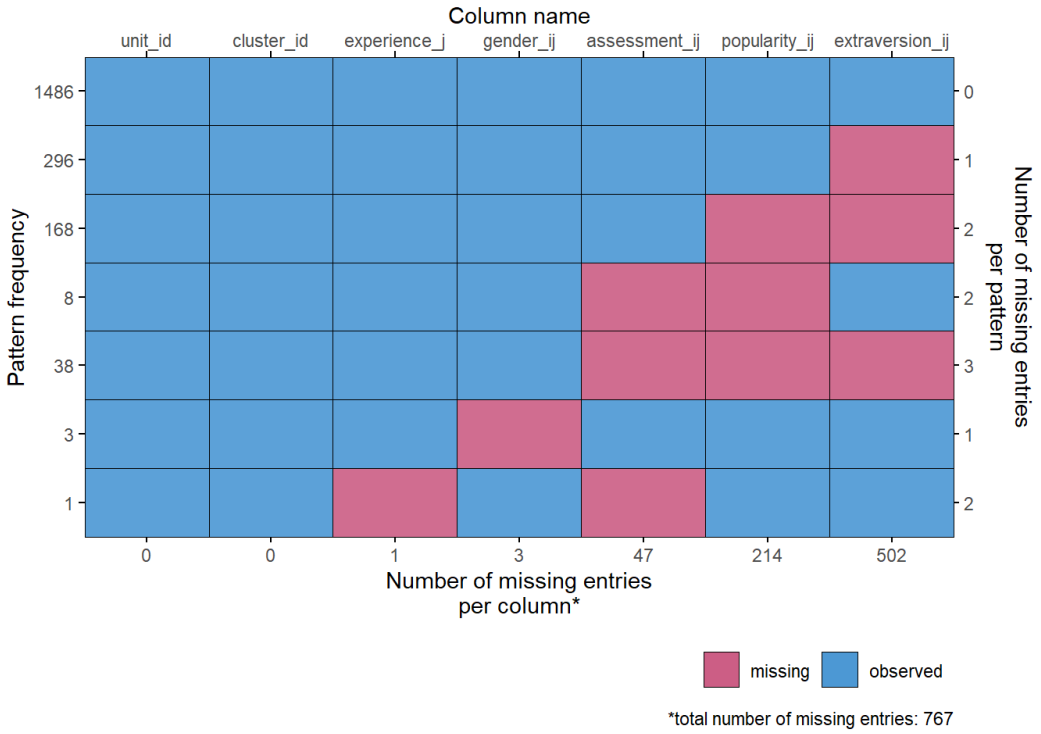
```
head(dat)
```

| unit_id | cluster_id | popularity_ij | gender_ij | extraversion_ij | experience_j | asse |
|---|---|---|---|---|---|---|
| 1 | 1 | 6.3 | 2 | 5 | 24 | |
| 2 | 1 | 4.9 | 1 | 7 | NA | |
| 3 | 1 | 5.3 | 2 | 4 | 24 | |
| 4 | 1 | 4.7 | 2 | 3 | 24 | |
| 5 | 1 | 6.0 | 2 | NA | 24 | |
| 6 | 1 | NA | 1 | NA | 24 | |

## 4.3.2 Explore missingness

Inspect the missing data patterns.

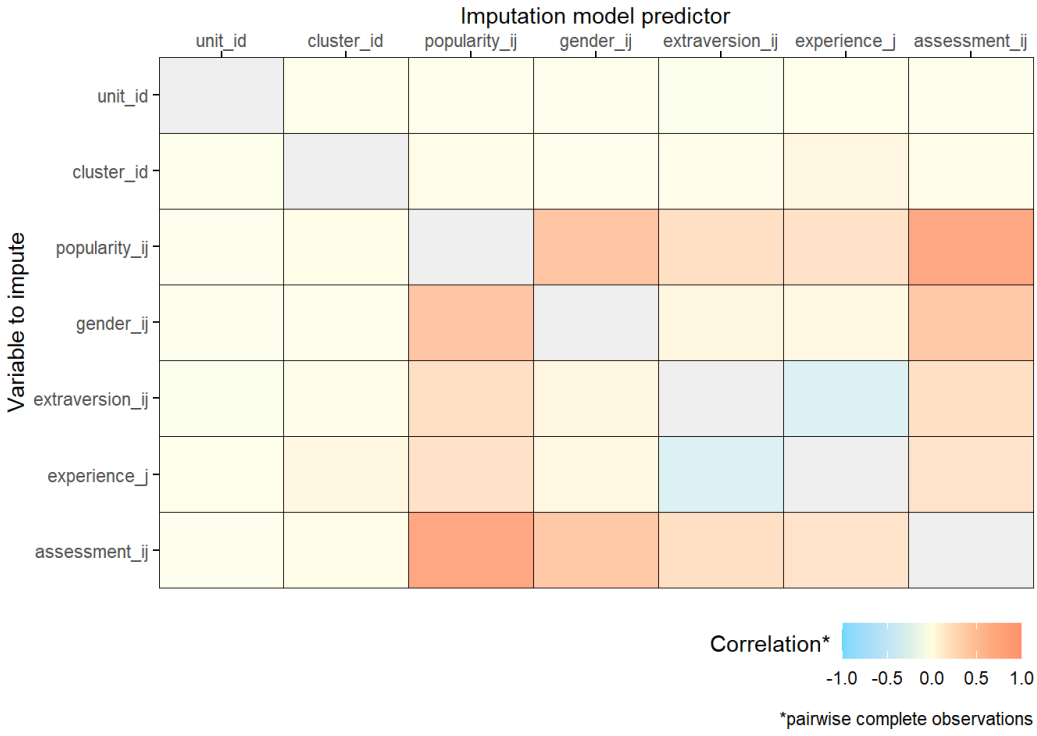
```
plot_pattern(dat, square = FALSE)
```




The unit identifier and cluster identifier are observed for all cases. All other variables are incomplete. The outcome variable 'popularity' is missing for 214 cases. The cluster-level variable 'teacher experience' is missing only once. The unit-level variables 'gender', 'extraversion' and 'teacher assessment' are missing for 3, 502 and 47 cases, respectively.

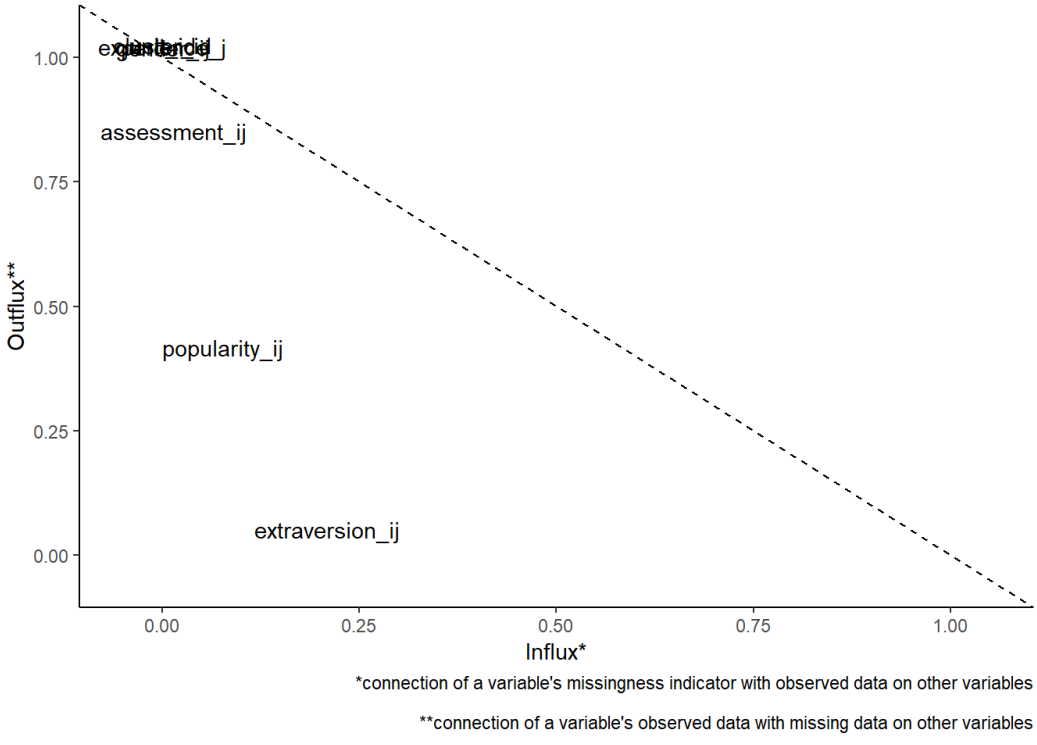
Teacher assessment is not part of the analysis model, but may serve as auxiliary variable in the imputation models. This is advisable if the observed data in the teacher assessment variable is strongly related to the observed data in the incomplete variables *or* their missingness indicators (see below).

```
plot_corr(dat, square = FALSE)
```




The auxiliary variable 'teacher assessment' may be useful in the imputation for the incomplete outcome variable 'popularity'. The unit identifier is not relevant, and will be left out of the imputation models.

```
plot_flux(dat)
```




+ interpretatie

## 4.4 Build imputation models

For each incomplete variable we need to define an imputation model: 1) choose the functional form of the imputation model and 2) select the imputation model predictors.

## 4.4.1 Imputation methods

Create the default methods vector, to be filled in with the appropriate imputation methods.

```
meth <- make.method(dat)
meth
```

```
        unit_id      cluster_id   popularity_ij       gender_ij
extraversion_ij
             ""              ""           "pmm"           "pmm"
"pmm"
   experience_j   assessment_ij
          "pmm"           "pmm"
```

and are thus denoted by an empty method

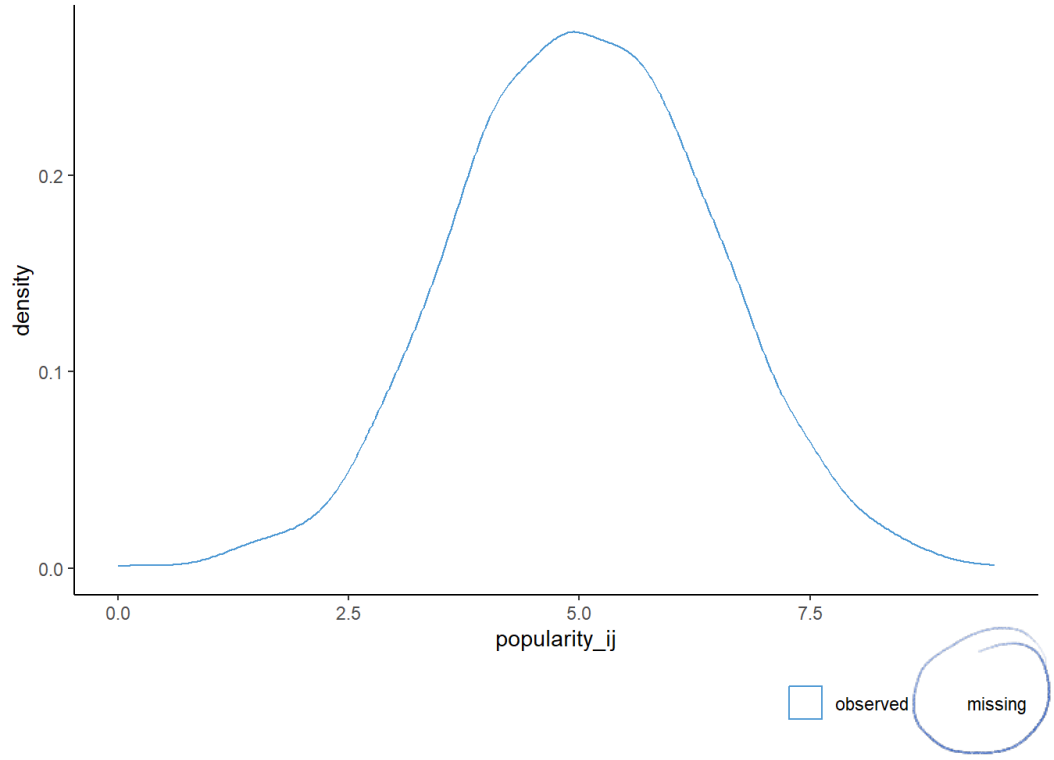
The complete identifier variables do not have to be imputed. The default imputation methods for the incomplete variables are not appropriate because of the multilevel structure of the data. We will fill in the methods vector with more appropriate imputation methods, based on the distribution of the incomplete variables.

### 4.4.1.1 Popularity

The outcome variable 'popularity' is approximately normally distributed aggregated over clusters.

```
ggmice(dat, aes(popularity_ij)) +
  geom_density()
```

```
Warning: Removed 214 rows containing non-finite outside the
scale range
(`stat_density()`).
```

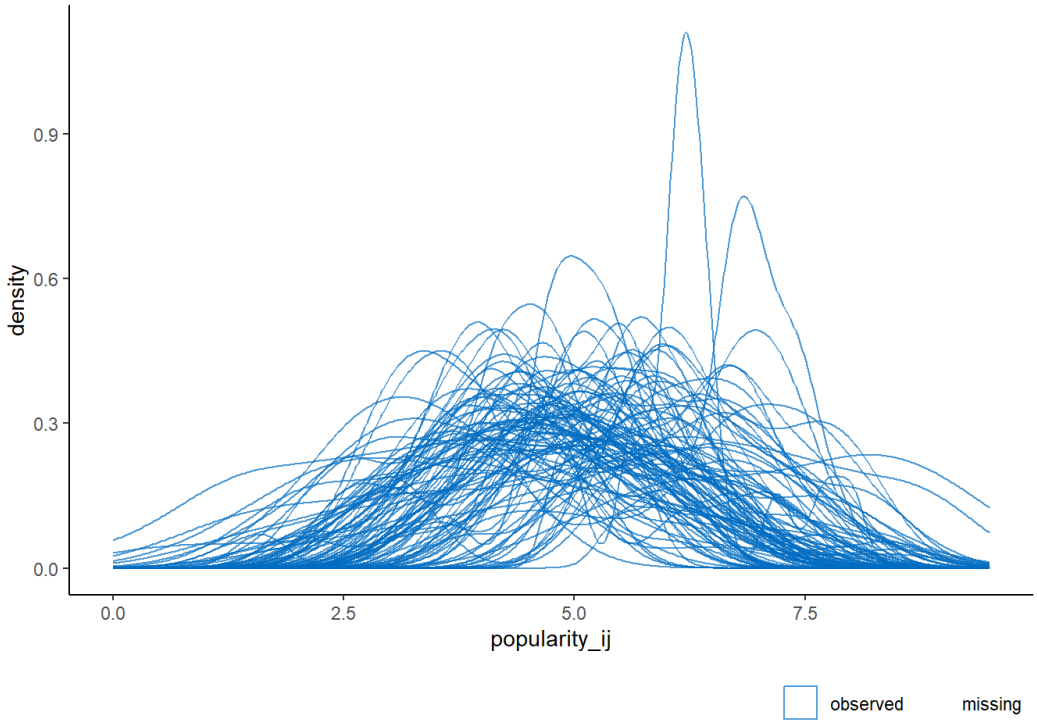



is dit niet
voorkeur?
default om weg
te laten als
1 dimensie in
ggmice?

This density plot of the incomplete outcome variable 'popularity' shows that aggregated across clusters, the distribution is approximately normal. Within clusters, however, there is more variability.

```
ggmice(dat, aes(popularity_ij, group = cluster_id)) +
  geom_density()
```

```
Warning: Removed 214 rows containing non-finite outside the
scale range
(`stat_density()`).
```

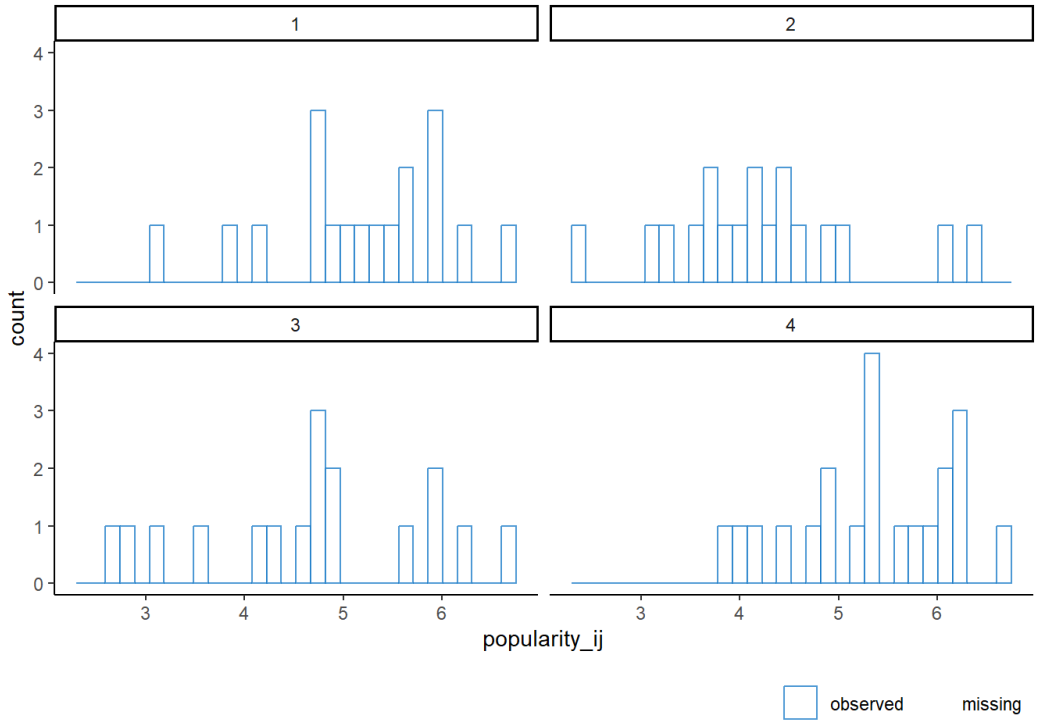



Further inspect the distribution of the outcome variable in the first few clusters.

```
dat |>
  filter(cluster_id %in% unique(dat$cluster_id)[1:4]) |>
  ggmice(aes(popularity_ij)) +
    geom_histogram(fill = "white") +
    facet_wrap(~cluster_id)
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```

```
Warning: Removed 7 rows containing non-finite outside the scale
range
(`stat_bin()`).
```




The variable is not normally distributed within clusters. We will use `2l.pmm`: a semi-parametric multilevel method using the linear mixed model with homogeneous error variances.

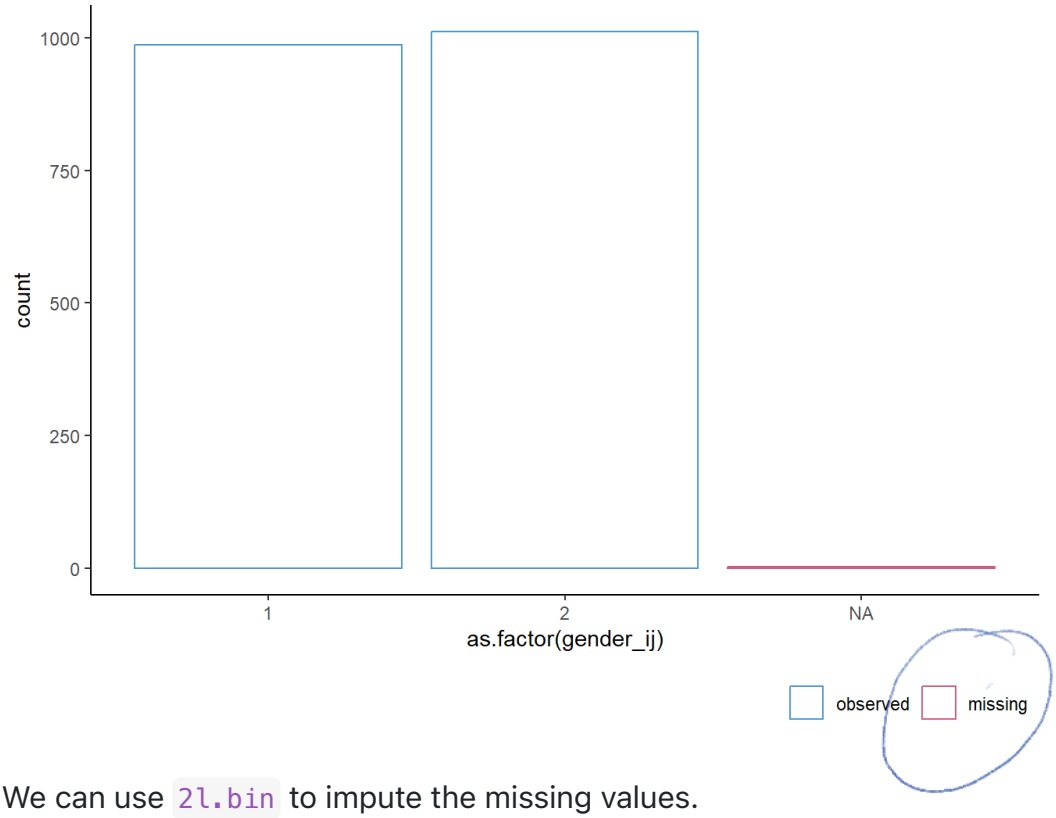
```
meth["popularity_ij"] <- "2l.pmm"
```

#### 4.4.1.2 Gender

The unit-level predictor variable 'gender' is dichotomous.

```
ggmice(dat, aes(as.factor(gender_ij))) +
  geom_bar(position = "stack", fill = "white")
```

```
Warning: Mapping variable 'as.factor(gender_ij)' recognized
internally as gender_ij.
Please verify whether this matches the requested mapping
variable.
```




We can use `2l.bin` to impute the missing values.

```
meth["gender_ij"] <- "2l.bin"
```

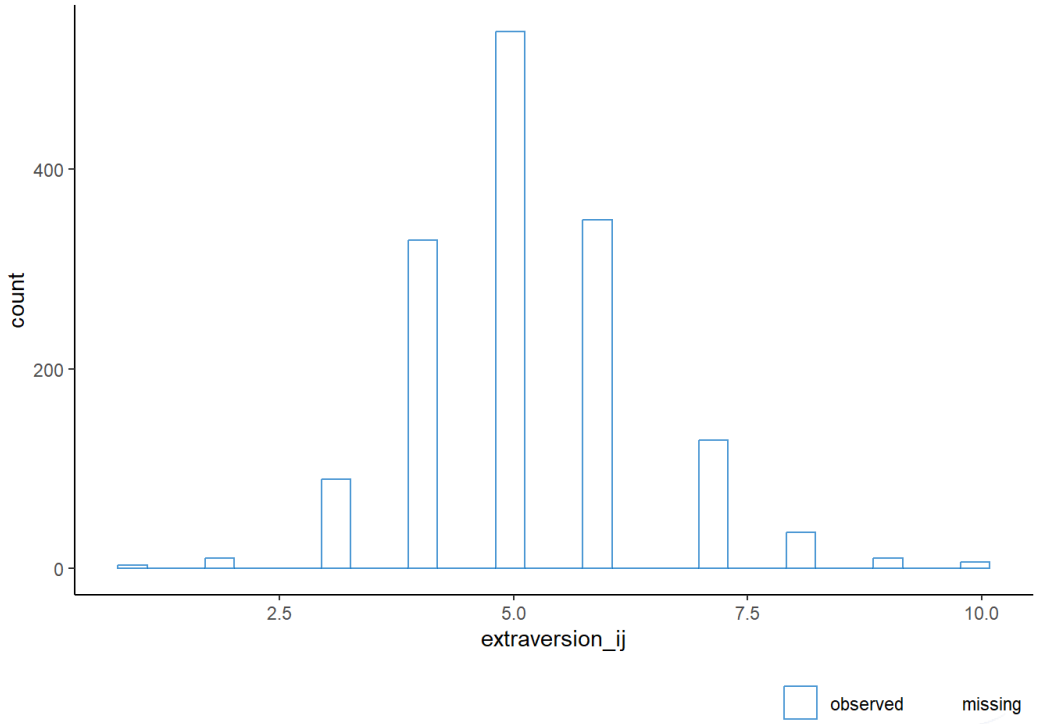
### 4.4.1.3 Extraversion

The unit-level predictor variable 'extraversion' is approximately normally distributed, but not truly continuous.

```
ggmice(dat, aes(extraversion_ij)) +
  geom_histogram(fill = "white")
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```

```
Warning: Removed 502 rows containing non-finite outside the
scale range
(`stat_bin()`).
```




We will use `2l.pmm` for this variable too.

```
meth["extraversion_ij"] <- "2l.pmm"
```

waarom geen norm?
- leg uit dat dit ook zou kunnen
- misschien voor/nadelen 2l.pmm kort behandelen tov 2l.norm

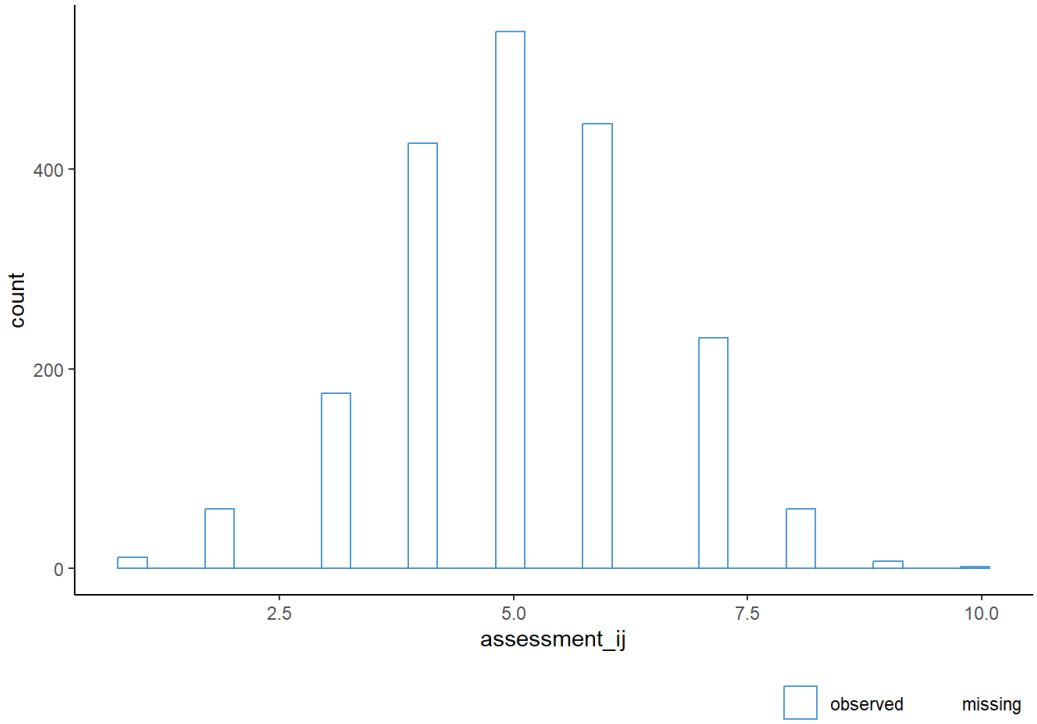
### 4.4.1.4 Teacher assessment

The unit-level auxiliary variable 'teacher assessment' is approximately normally distributed but not continuous either.

```
ggmice(dat, aes(assessment_ij)) +
  geom_histogram(fill = "white")
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.

Warning: Removed 47 rows containing non-finite outside the
scale range
(`stat_bin()`).
```




We will use `2l.pmm` to impute the missing values.

```
meth["assessment_ij"] <- "2l.pmm"
```

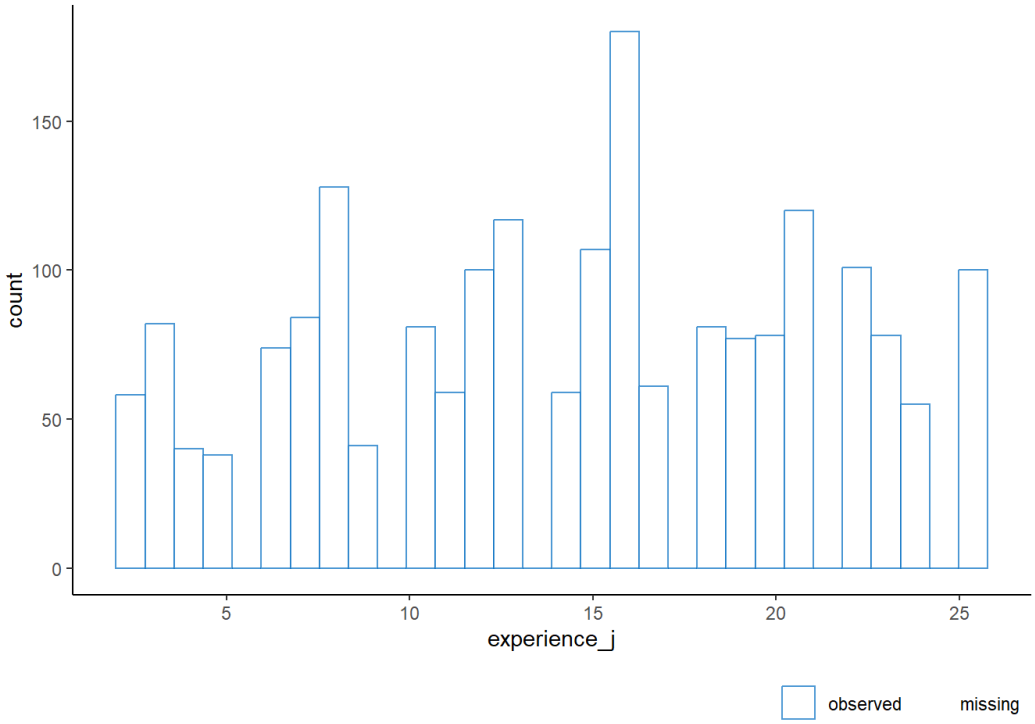
#### 4.4.1.5 Teacher experience

The cluster-level variable 'teacher experience' is continuous.

```
ggmice(dat, aes(experience_j)) +
  geom_histogram(fill = "white")
```

```
`stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.

Warning: Removed 1 row containing non-finite outside the scale
range
(`stat_bin()`).
```




We can use `2lonly.mean` to impute the missing values if we are sufficiently certain that this variable... Otherwise, `2lonly.pmm` is a good alternative.

```
meth["experience_j"] <- "2lonly.pmm"
```

#### 4.4.1.6 Methods vector

Re-evaluate the methods vector

```
meth
```

```
        unit_id      cluster_id   popularity_ij       gender_ij
extraversion_ij
             ""              ""        "2l.pmm"        "2l.bin"
"2l.pmm"
   experience_j   assessment_ij
   "2lonly.pmm"        "2l.pmm"
```

All incomplete variables have a multilevel imputation method.

### 4.4.2 Imputation model predictors

Imputation model predictors are specified with the `predictorMatrix` argument in `mice`. The default predictor matrix is not appropriate for multilevel imputation models.

```
pred <- make.predictorMatrix(dat)
```

The cluster identifier `cluster_id` will serve as clustering variable for all imputation models, whereas the unit identifier will not be part of any imputation model.

```
pred[, "cluster_id"] <- -2
pred[, "unit_id"] <- 0
```

#### 4.4.2.1 Popularity

For the outcome variable 'popularity', we will include all unit-level variables that appear in the most general analytic model to be applied to imputed data. We will include a fixed effect for 'gender` and a random effect for 'extraversion'.

```
pred["popularity_ij", "gender_ij"] <- 1
pred["popularity_ij", "extraversion_ij"] <- 2
```

The next step is to include the disaggregated cluster means of all level-1 variables, which means we have to edit the predictor matrix.

```
pred["popularity_ij", "gender_ij"] <- 3
pred["popularity_ij", "extraversion_ij"] <- 4
```

There are no unit-level interactions in the analytic model, so we will continue to the next step: include all level-2 predictors.

```
pred["popularity_ij", "experience_j"] <- 1
```

There are no cluster-level interactions required either, but there is a cross-level interaction implied by the analytic model. We will use passive imputation to impute the interaction effect for extraversion and experience. This is done by adding an empty column to the data for the interaction term and letting `mice` calculate the interaction effect on the fly.

```
dat$extraversion_experience <- dat$extraversion_ij * dat$experi
```

We need to edit the methods vector to accommodate this additional variable. Set the imputation method for the interaction term to passive imputation.

```
meth["extraversion_experience"] <- "~ I(extraversion_ij * exper
```

Add the additional variable to the predictor matrix as well.

```
pred <- rbind(pred, extraversion_experience = 0)
pred <- cbind(pred, extraversion_experience = 0)
```

Further edit the predictor matrix so the interaction term is included in the imputation model for 'popularity'.

```
pred["popularity_ij", "extraversion_experience"] <- 1
```

And finally, include a predictor related to the missingness and the target: the auxiliary variable teacher assessment.

```
pred["popularity_ij", "assessment_ij"] <- 1
```

The imputation model predictors for popularity now are as follows.

```
pred["popularity_ij", ]
```

```
                unit_id              cluster_id
popularity_ij
                      0                      -2
0
             gender_ij         extraversion_ij
experience_j
                      3                       4
1
          assessment_ij extraversion_experience
                      1                       1
```

#### 4.4.2.2 Gender

The analytic model implies a relation between gender and popularity. So we need to include the outcome variable in the imputation model.

```
pred["gender_ij", "popularity_ij"] <- 2
```

#### 4.4.2.3 Extraversion

The analytic model implies a relation between gender and popularity. So we need to include the outcome variable in the imputation model.

```
pred["extraversion_ij", "popularity_ij"] <- 2
```

#### 4.4.2.4 Experience

The analytic model implies a relation between the cluster-level variable teacher experience and the unit-level variable popularity.

```
pred["experience_j", "popularity_ij"] <- 2
```

#### 4.4.2.5 Teacher assessment

The analytic model does not imply any relation with teacher assessment. This variable is only used as auxiliary variable.

```
pred["assessment_ij", "popularity_ij"] <- 4
```

#### 4.4.2.6 Predictor matrix

Finally, clean up the predictor matrix by setting the diagonal to `0`, to exclude any terms involving the target.

```
diag(pred) <- 0
```

Our final predictor matrix is as follows.

```
pred
```

```
                       unit_id cluster_id popularity_ij
gender_ij
unit_id                      0         -2             1
1
cluster_id                   0          0             1
1
popularity_ij                0         -2             0
3
gender_ij                    0         -2             2
0
extraversion_ij              0         -2             2
1
experience_j                 0         -2             2
1
assessment_ij                0         -2             4
1
extraversion_experience      0          0             0
0
                       extraversion_ij experience_j
assessment_ij
unit_id                              1            1
1
cluster_id                           1            1
1
```

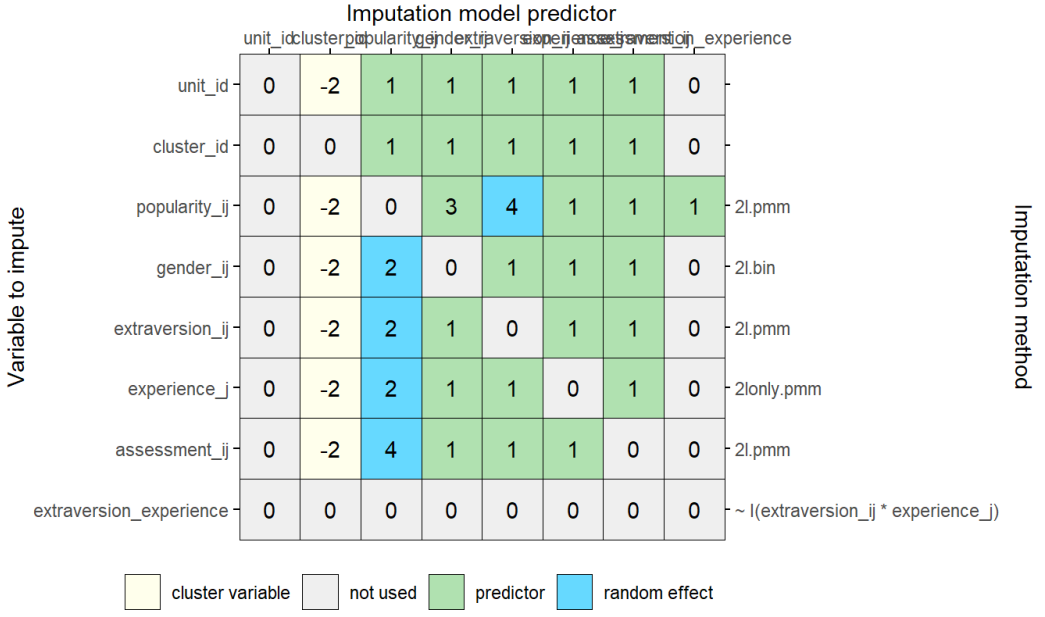
```
popularity_ij                    4           1
1
gender_ij                        1           1
1
extraversion_ij                  0           1
1
experience_j                     1           0
1
assessment_ij                    1           1
0
extraversion_experience          0           0
0
                      extraversion_experience
unit_id                                     0
cluster_id                                  0
popularity_ij                               1
gender_ij                                   0
extraversion_ij                             0
experience_j                                0
assessment_ij                               0
extraversion_experience                     0
```

### 4.4.3 Evaluate imputation models

The imputation models can be visualized as follows.

that has been constructed by specifying the predictormatrix and method vector

```
plot_pred(pred, meth = meth)
```



# 4.5 Run imputation algorithm

## 4.5.1 Impute

Impute the data using the specified imputation methods and predictor matrix.

```
imp <- mice(
  dat,
  m = 1,
  method = meth,
  predictorMatrix = pred,
  maxit = 1)
```

waarom niet

Does not run. Set imputation method for experience to `2lonly.mean` and re-run.

geef aan dat de predictormatrix specificatie nu redundant is en genegeerd wordt.

```
meth["experience_j"] <- "2lonly.mean"
imp <- mice(
  dat,
  method = meth,
  predictorMatrix = pred,
  maxit = 1,
  printFlag = FALSE)
```

```
Warning in mice.impute.2l.bin(y = c(2, 1, 2, 2, 2, 1, 1, 1, 1,
1, 2, 2, : glmer
does not run. Simplify imputation model

boundary (singular) fit: see help('isSingular')

Warning in mice.impute.2l.bin(y = c(2, 1, 2, 2, 2, 1, 1, 1, 1,
1, 2, 2, : glmer
does not run. Simplify imputation model

Warning in mice.impute.2l.bin(y = c(2, 1, 2, 2, 2, 1, 1, 1, 1,
1, 2, 2, : glmer
does not run. Simplify imputation model

Warning in mice.impute.2l.bin(y = c(2, 1, 2, 2, 2, 1, 1, 1, 1,
1, 2, 2, : glmer
does not run. Simplify imputation model

Warning in mice.impute.2l.bin(y = c(2, 1, 2, 2, 2, 1, 1, 1, 1,
1, 2, 2, : glmer
does not run. Simplify imputation model
```

wat moet de lezer hiermee?

This solves the issue in imputing teacher experience. Now edit the method for gender.

```
meth["gender_ij"] <- "2l.pmm"
imp <- mice(
  dat,
  method = meth,
  predictorMatrix = pred,
  maxit = 1,
  printFlag = FALSE)
```

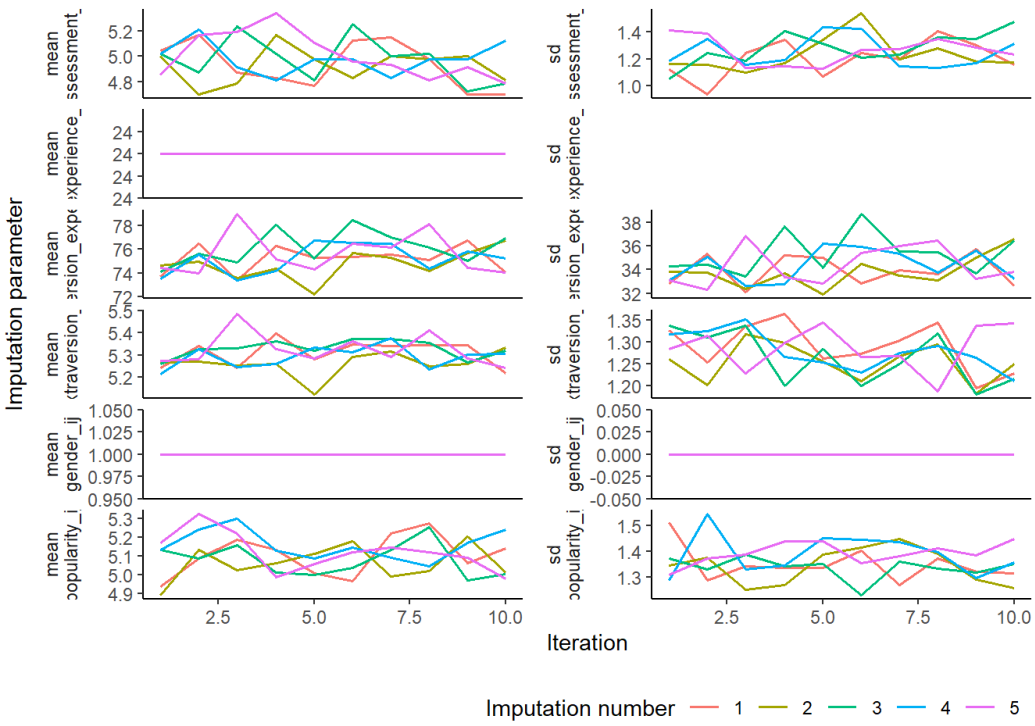
This code runs! Add iterations to reach algorithmic convergence.

```
imp <- mice.mids(imp, maxit = 9, printFlag = FALSE)
```

## 4.5.2 Evaluate convergence

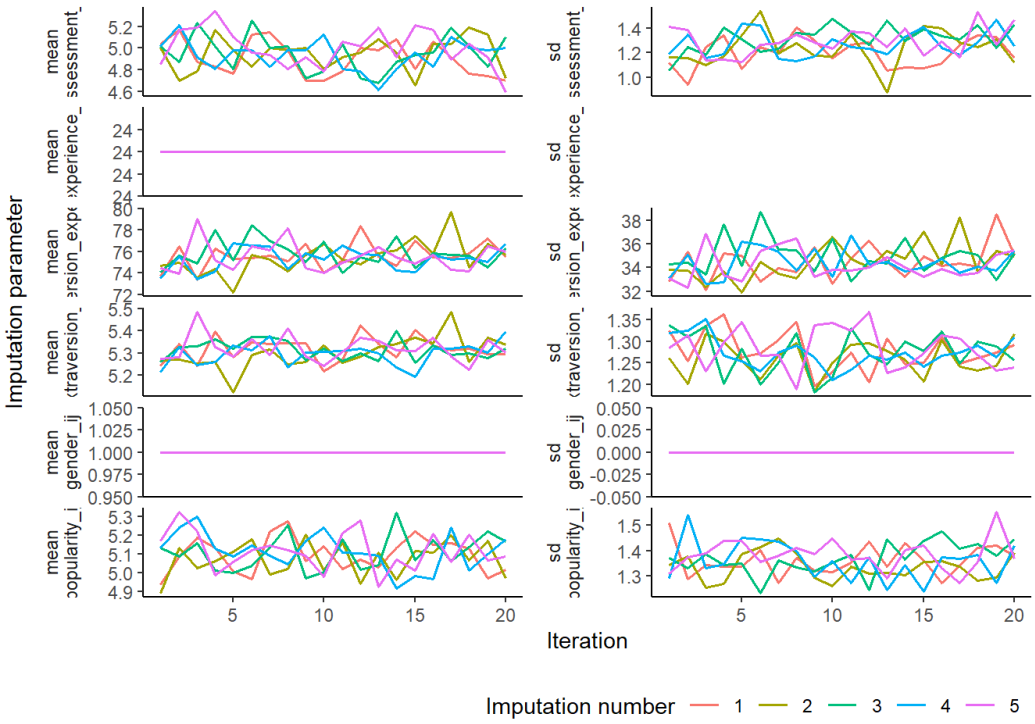
Check for algorithmic non-convergence.

```
plot_trace(imp)
```




The trace plots for teacher experience and gender do not exhibit the typical characteristics of convergence: the lines do not intermingle nicely, and thus require further inspection. The traceplot of the interaction term also requires further inspection, as there appears to be some non-stationarity in the standard deviation of the imputations. We will append some extra iterations to evaluate the potential upward trend.

```
imp <- mice.mids(imp, maxit = 10, printFlag = FALSE)
plot_trace(imp)
```



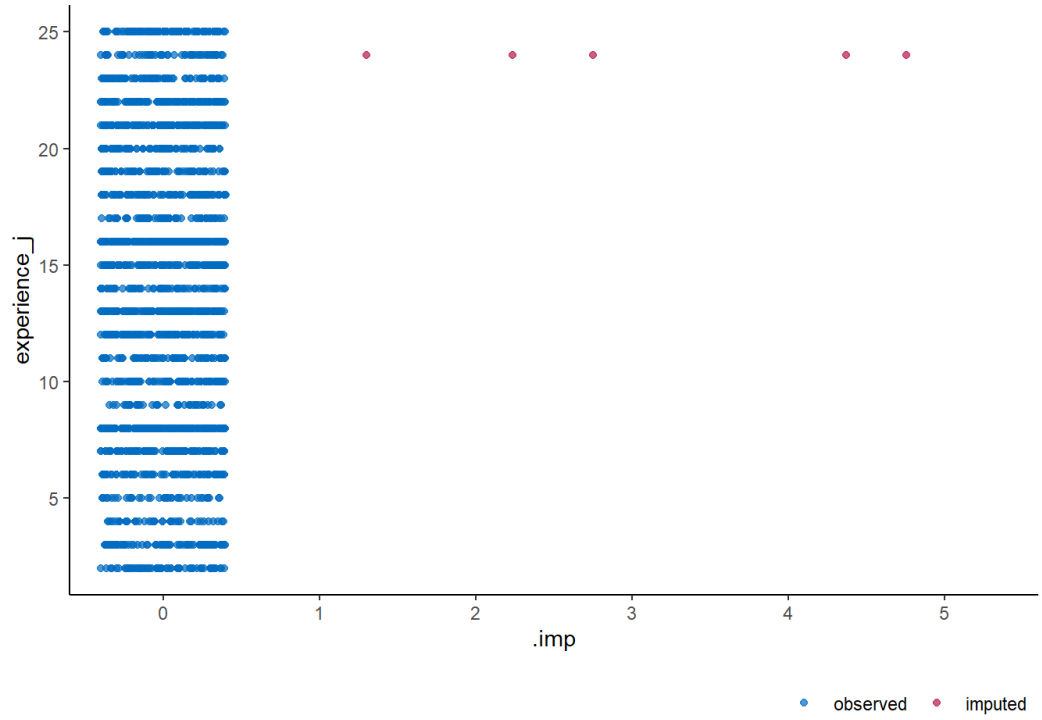

After 20 iterations, there is no apparent trending visible in the trace plots.

## 4.6 Evaluate imputations

### 4.6.1 Inspect imputed values

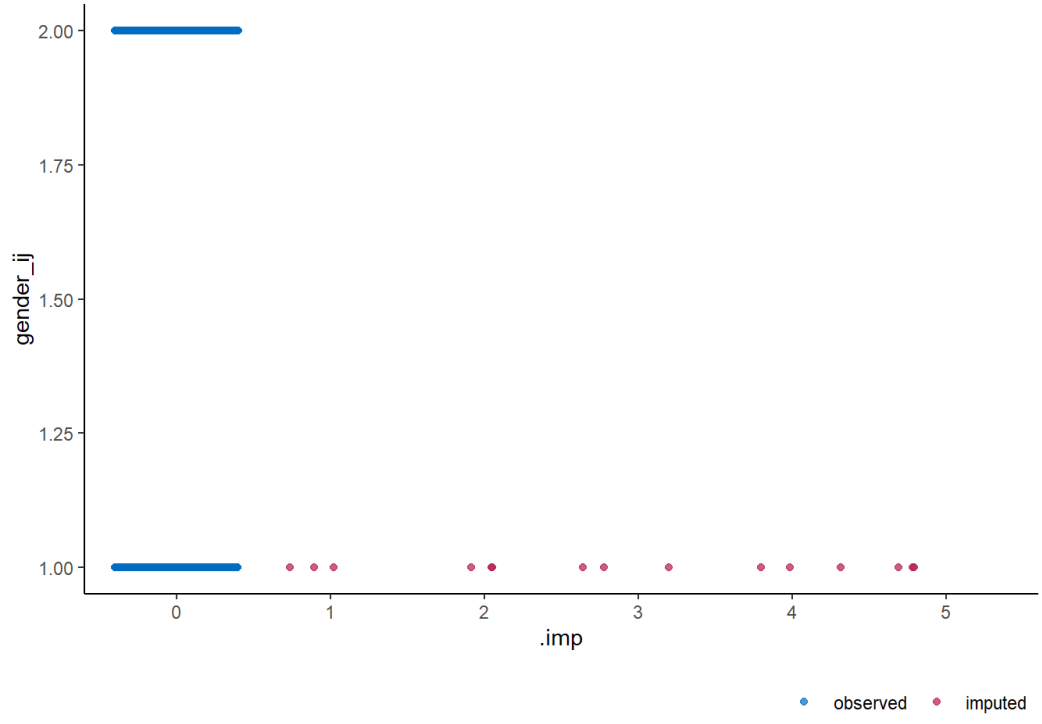
We will first evaluate the imputations of the variables with atypical trace plots.

```
ggmice(imp, aes(.imp, experience_j)) +
  geom_jitter(height = 0)
```




This graphs shows that there is only one imputed value and no variability between the imputations, which explains the atypical trace plots for this variable.

```
ggmice(imp, aes(.imp, gender_ij)) +
  geom_jitter(height = 0)
```

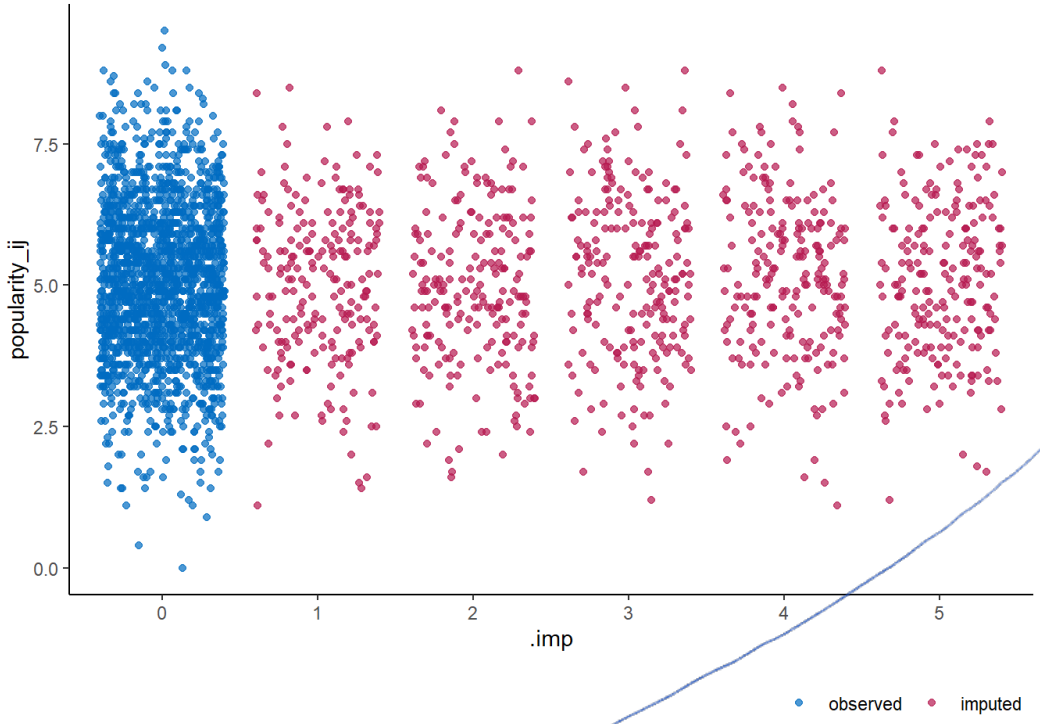



Equivalently, for gender, there is no variability in the imputed values. This leaves no more apparent issues with convergence. We can now evaluate

the imputations of the other variables.

#### 4.6.1.1 Popularity

```
ggmice(imp, aes(.imp, popularity_ij)) +
  geom_jitter(height = 0)
```



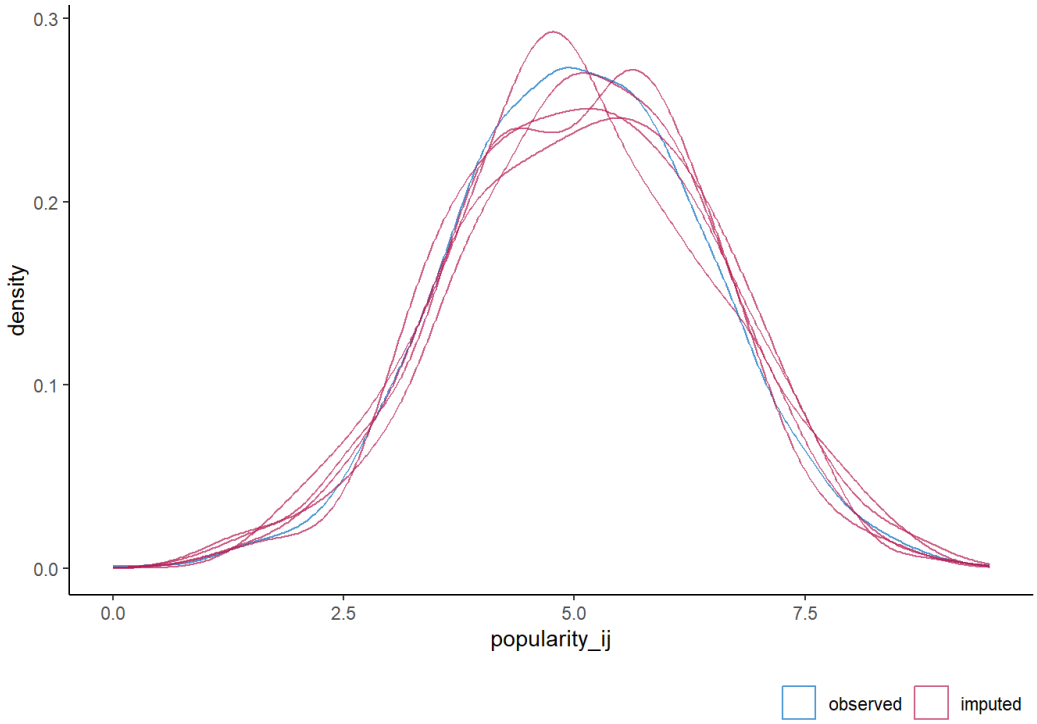

There is one imputed value clearly outside of the range of observed values.

```
range(mice::complete(imp)$popularity_ij)
```

```
[1] 0.0 9.5
```

If a negative value for popularity is impossible, one might decide to re-impute the data using a different imputation method. For our analyses, a negative value falls outside the range of observations, but does not require re-imputation.

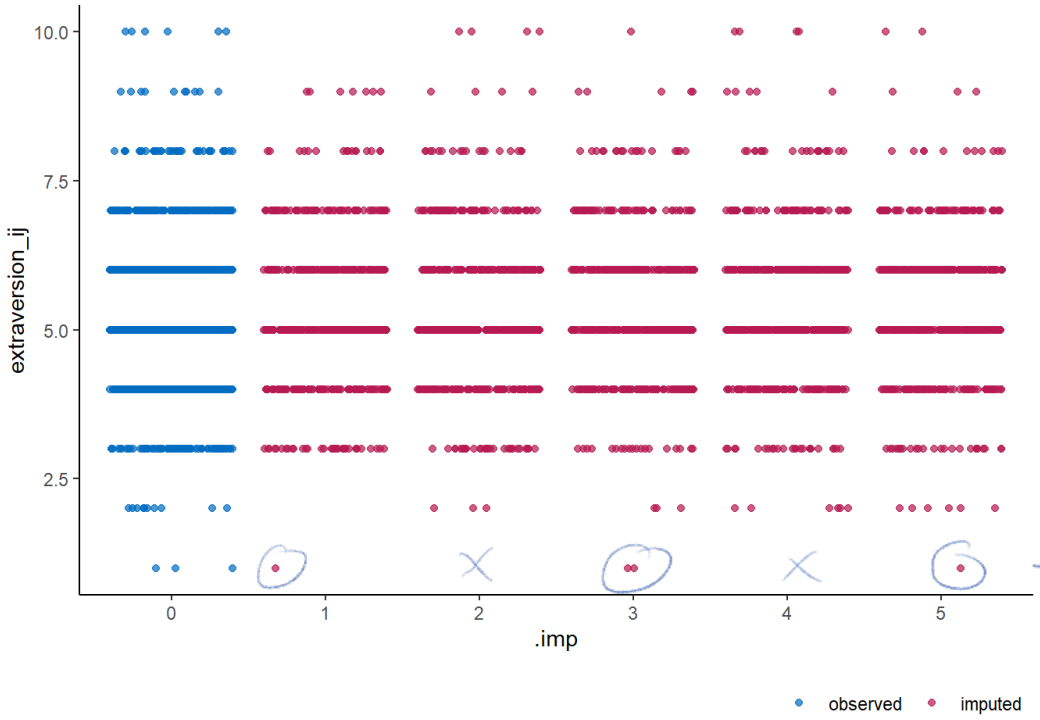
```
ggmice(imp, aes(popularity_ij, group = .imp)) +
  geom_density()
```




### 4.6.1.2 Extraversion

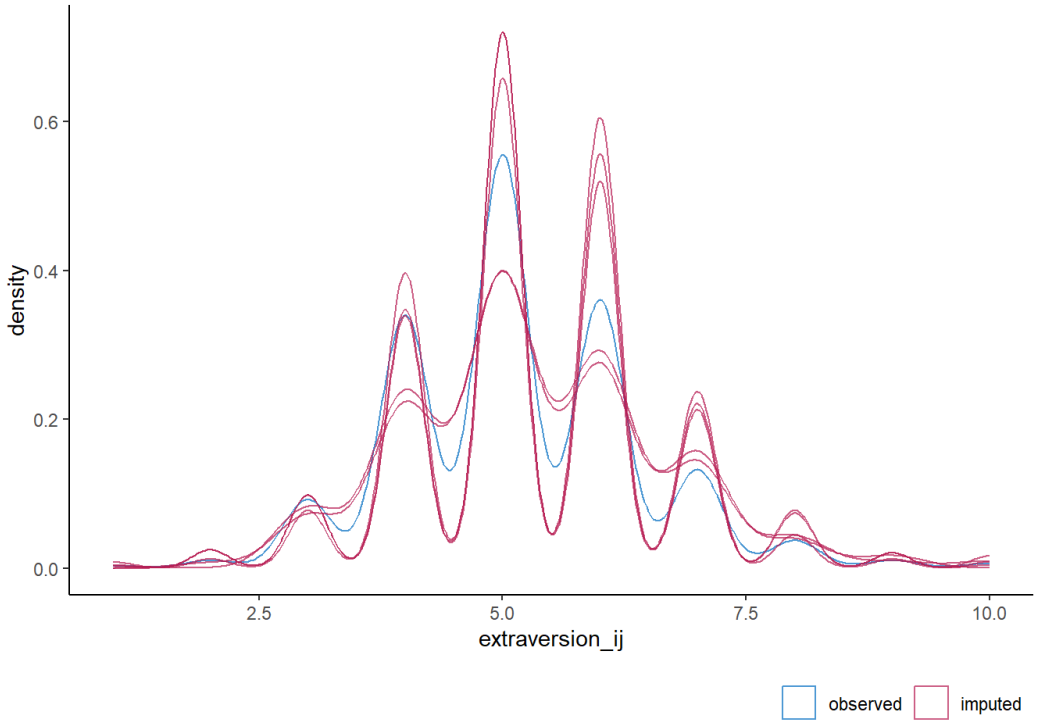
vanaf hier mist de interpretatie.

```
ggmice(imp, aes(.imp, extraversion_ij)) +
  geom_jitter(height = 0)
```




→ meer om imputatie variantie te highlighten

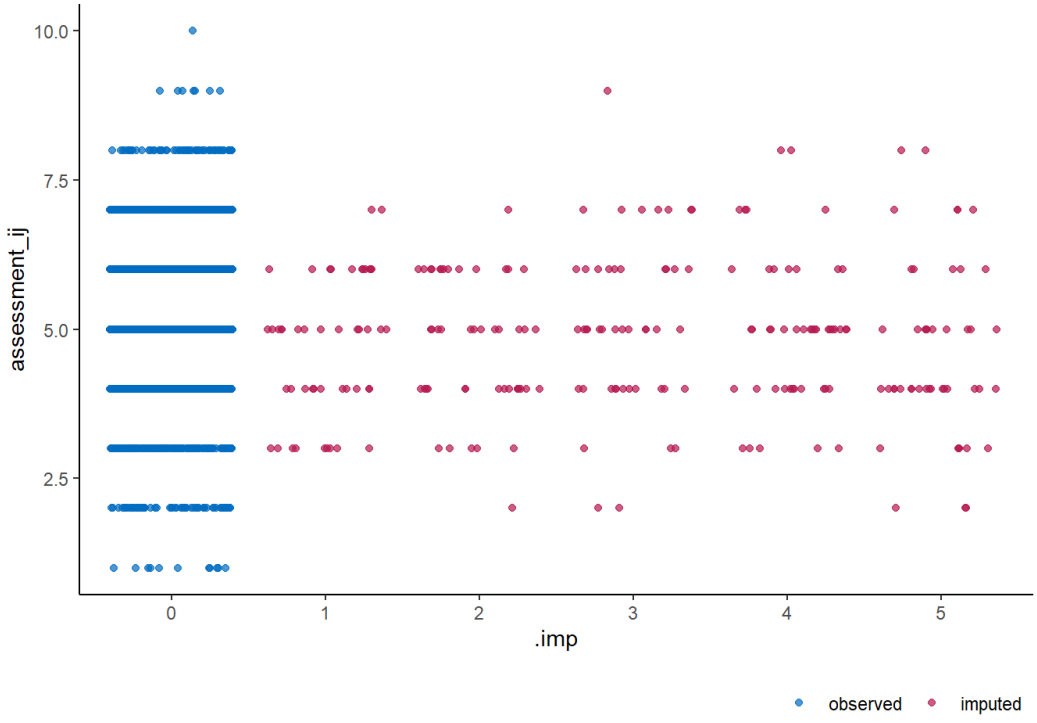
```
ggmice(imp, aes(extraversion_ij, group = .imp)) +
  geom_density()
```




Vingerpatroon verstoord door imputatiemethode

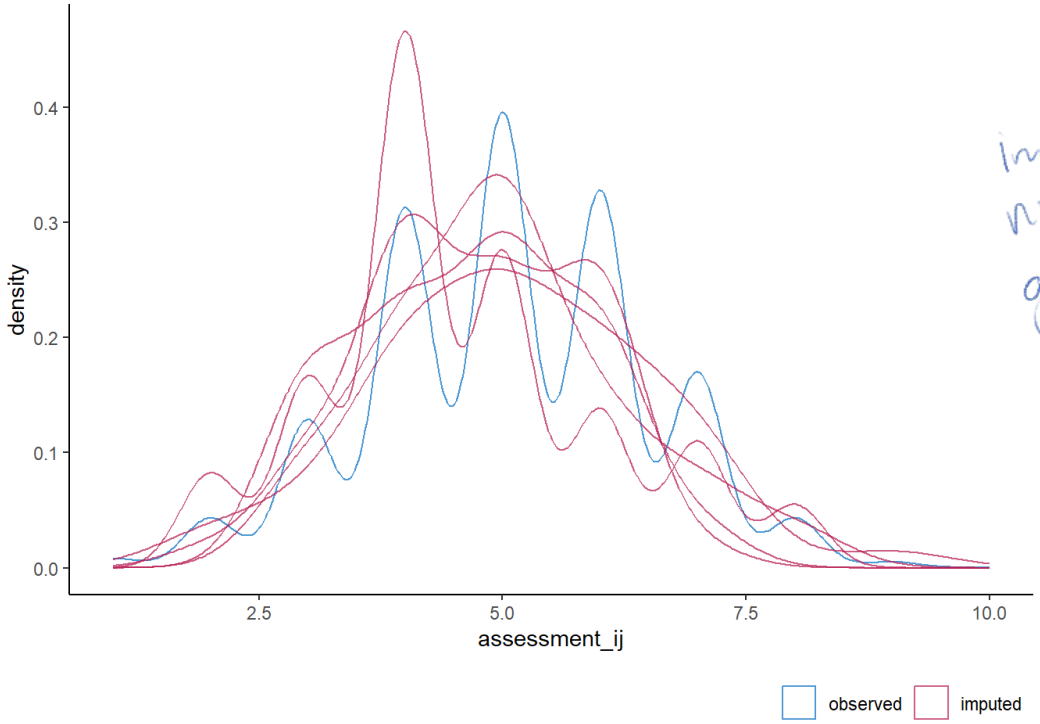
### 4.6.1.3 Teacher assessment

```
ggmice(imp, aes(.imp, assessment_ij)) +
  geom_jitter(height = 0)
```




```
ggmice(imp, aes(assessment_ij, group = .imp)) +
```

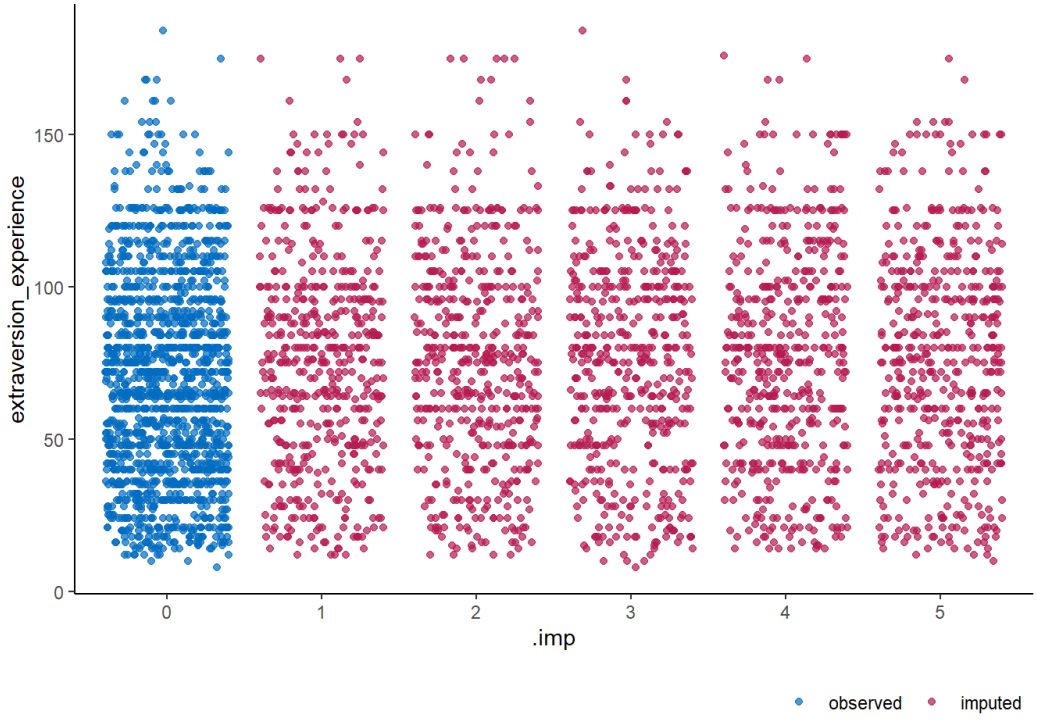
```
  geom_density()
```



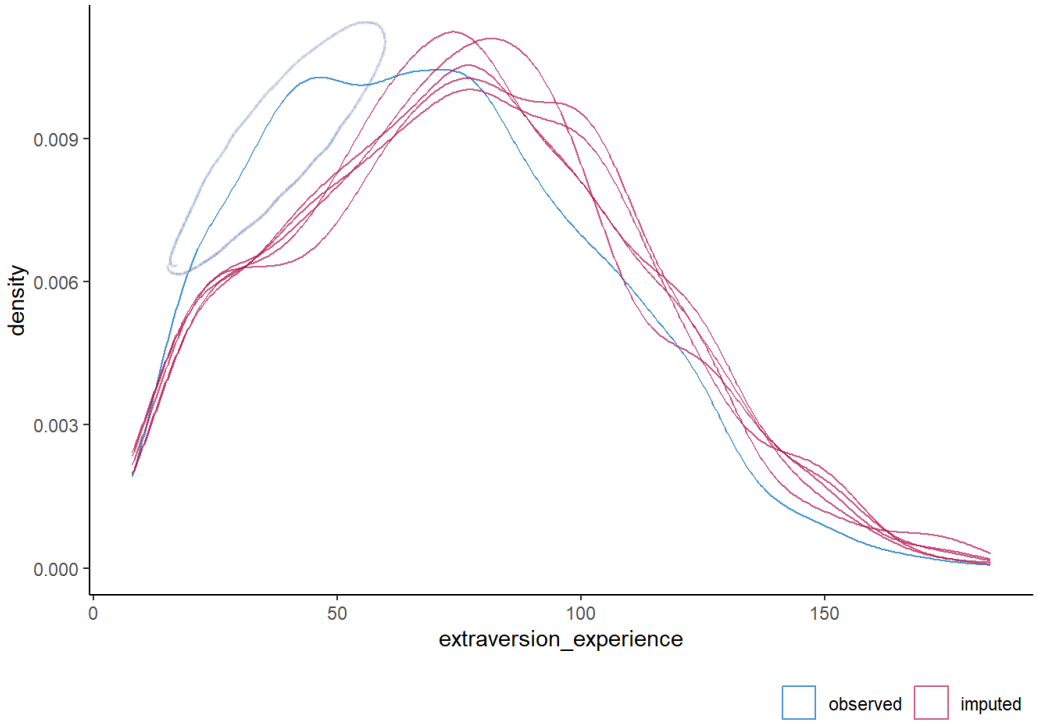

imputatiemethode niet integer. goed om aan te geven

### 4.6.1.4 Interaction term

```
ggmice(imp, aes(.imp, extraversion_experience)) +
  geom_jitter(height = 0)
```



```
ggmice(imp, aes(extraversion_experience, group = .imp)) +
  geom_density()
```




iets minder massa hier dan bij de obs

is deze data MAR?

zo ja, dan waarschijnlijk MAR right.

## 4.7 Analyze imputed data

After successful imputation, we can analyze the imputed data and pool the results for further analysis.

Single-level model.

```
mod_imp_singlelevel <- with(
  imp,
  lm(popularity_ij ~ 1)
  )
```

```
pool(mod_imp_singlelevel) |> tidy(conf.int = TRUE)
```

| term | estimate | std.error | statistic | p.value | conf.low | conf.high | b | df | df |
|---|---|---|---|---|---|---|---|---|---|
| (Intercept) | 5.06 | 0.033 | 155 | 0 | 4.99 | 5.12 | 0 | 301 | |

Intercept-only model.

```
mod_imp_intercept <- with(
  imp,
```

```
  lmer(
    popularity_ij ~ (1 | cluster_id),
    REML = FALSE)
  )
```

```
pool(mod_imp_intercept) |> tidy(conf.int = TRUE)
```

| term | estimate | std.error | statistic | p.value | conf.low | conf.high | b | df | c |
|---|---|---|---|---|---|---|---|---|---|
| (Intercept) | 5.06 | 0.084 | 60.5 | 0 | 4.9 | 5.22 | 0 | 1765 | |

```
mitml::testEstimates(as.mitml.result(mod_imp_intercept), extra.
```

```
Call:

mitml::testEstimates(model =
as.mitml.result(mod_imp_intercept),
    extra.pars = TRUE)

Final parameter estimates and inferences obtained from 5
imputed data sets.

             Estimate Std.Error   t.value        df   P(>|t|)
RIV       FMI
(Intercept)     5.060     0.084    60.542 17388.669     0.000
0.015     0.015

                                 Estimate
Intercept~~Intercept|cluster_id    0.622
Residual~~Residual                 1.299
ICC|cluster_id                     0.324

Unadjusted hypothesis test as appropriate in larger samples.
```

Model with explanatory variables.

```
mod_imp_predictors <- with(
  imp,
  lmer(
    popularity_ij ~ gender_ij + extraversion_ij + experience_j
    REML = FALSE)
  )
```

```
pool(mod_imp_predictors) |> tidy(conf.int = TRUE)
```

| term | estimate | std.error | statistic | p.value | conf.low | conf.high | b |
|---|---|---|---|---|---|---|---|
| (Intercept) | -0.442 | 0.221 | -2.00 | 0.054 | -0.891 | 0.007 | 0.014 |
| gender_ij | 1.153 | 0.082 | 13.99 | 0.000 | 0.958 | 1.349 | 0.004 |
| extraversion_ij | 0.482 | 0.026 | 18.34 | 0.000 | 0.426 | 0.539 | 0.000 |
| experience_j | 0.088 | 0.010 | 8.49 | 0.000 | 0.067 | 0.109 | 0.000 |

```
mitml::testEstimates(as.mitml.result(mod_imp_predictors), extra
```

```
Call:

mitml::testEstimates(model =
as.mitml.result(mod_imp_predictors),
    extra.pars = TRUE)

Final parameter estimates and inferences obtained from 5
imputed data sets.

                Estimate Std.Error   t.value        df
P(>|t|)       RIV       FMI
(Intercept)       -0.442     0.221    -1.997    35.674
0.054     0.503     0.369
gender_ij          1.153     0.082    13.993     6.980
0.000     3.116     0.806
extraversion_ij    0.482     0.026    18.342    13.906
0.000     1.157     0.591
experience_j       0.088     0.010     8.490    40.072
0.000     0.462     0.348

                               Estimate
Intercept~~Intercept|cluster_id    0.274
Residual~~Residual                 0.707
ICC|cluster_id                     0.280

Unadjusted hypothesis test as appropriate in larger samples.
```

For many explanatory variables, a large part of the variance is due to the missingness. For example, the relative increase in variance compared to the sampling variance for the effect of gender is more than 90%. This means that there is a lot of uncertainty in the estimate of the effect. We could increase the number of imputations.

```
# #this code does not add imputations, only iterations. should
# mice.mids(imp, m = 15, printFlag = FALSE)
```

Model with explanatory variables, extraversion slope random.

```
mod_imp_randomslope <- with(
  imp,
  lmer(
    popularity_ij ~ gender_ij + extraversion_ij + experience_j
      (1  + extraversion_ij | cluster_id))
  )
```

```
Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
control$checkConv, :
Model failed to converge with max|grad| = 0.00274122 (tol =
0.002, component 1)
```

Analysis model does not converge.

```
mod_imp_randomslope <- with(
  imp,
  lmer(
    popularity_ij ~ gender_ij + extraversion_ij + experience_j
      (1  + extraversion_ij | cluster_id),
    control = lmerControl(optimizer = "nloptwrap", optCtrl = lis
    )
  )
```

```
Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
control$checkConv, :
Model failed to converge with max|grad| = 0.00274122 (tol =
0.002, component 1)
```

A different optimizer or extra iterations do not help.

```
mod_imp_randomslope <- with(
  imp,
  lmer(
    popularity_ij ~ gender_ij + extraversion_ij + experience_j
      (1  + extraversion_ij | cluster_id),
    control = lmerControl(optCtrl = list(maxfun = 2e5)), REML =
    )
  )
```

```
Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
control$checkConv, :
Model failed to converge with max|grad| = 0.00271058 (tol =
0.002, component 1)

Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
```

```
control$checkConv, :
Model failed to converge with max|grad| = 0.00509317 (tol =
0.002, component 1)

Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
control$checkConv, :
Model failed to converge with max|grad| = 0.00350807 (tol =
0.002, component 1)
```

Still no convergence. Use different mixed modeling technique.

```
library(nlme)
```

```
Attaching package: 'nlme'

The following object is masked from 'package:lme4':

    lmList

The following object is masked from 'package:dplyr':

    collapse
```

```
mod_imp_randomslope <- with(
  imp,
  lme(
    popularity_ij ~ gender_ij + extraversion_ij + experience_j,
    random = ~ 1  + extraversion_ij | cluster_id))
```

```
pool(mod_imp_randomslope) |> tidy(conf.int = TRUE)
```

| term | estimate | std.error | statistic | p.value | conf.low | conf.high | b |
|---|---|---|---|---|---|---|---|
| (Intercept) | -0.508 | 0.239 | -2.12 | 0.039 | -0.989 | -0.028 | 0.013 |
| gender_ij | 1.146 | 0.080 | 14.33 | 0.000 | 0.957 | 1.334 | 0.004 |
| extraversion_ij | 0.476 | 0.030 | 15.61 | 0.000 | 0.414 | 0.537 | 0.000 |
| experience_j | 0.093 | 0.010 | 9.20 | 0.000 | 0.073 | 0.113 | 0.000 |

```
mitml::testEstimates(as.mitml.result(mod_imp_randomslope), extra
```

```
Call:
```

```
mitml::testEstimates(model =
as.mitml.result(mod_imp_randomslope),
    extra.pars = TRUE)

Final parameter estimates and inferences obtained from 5
imputed data sets.

                 Estimate Std.Error   t.value        df
P(>|t|)       RIV       FMI
(Intercept)        -0.508     0.239    -2.125    50.493
0.038     0.392     0.308
gender_ij           1.146     0.080    14.328     7.180
0.000     2.943     0.796
extraversion_ij     0.476     0.030    15.608    41.156
0.000     0.453     0.343
experience_j        0.093     0.010     9.203    51.555
0.000     0.386     0.305

                                          Estimate
Intercept~~Intercept|cluster_id              1.106
extraversion_ij~~extraversion_ij|cluster_id  0.031
Intercept~~extraversion_ij|cluster_id       -0.161
Residual~~Residual                           0.674
ICC|cluster_id                               0.617

Unadjusted hypothesis test as appropriate in larger samples.
```

Model with with cross-level interaction

```
mod_imp_interaction <- with(
  imp,
  lmer(
    popularity_ij ~ gender_ij + extraversion_ij + experience_j +
      extraversion_ij:experience_j + (1  + extraversion_ij | clu
    REML = FALSE)
  )
```

```
Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
control$checkConv, :
Model failed to converge with max|grad| = 0.00972292 (tol =
0.002, component 1)

Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
control$checkConv, :
Model failed to converge with max|grad| = 0.00336617 (tol =
0.002, component 1)

Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
```

```
control$checkConv, :
Model failed to converge with max|grad| = 0.0134534 (tol =
0.002, component 1)
```

Does not converge. Use `nmle` again.

```
mod_imp_interaction <- with(
  imp,
  lme(
    popularity_ij ~ gender_ij + extraversion_ij + experience_j +
      extraversion_ij:experience_j,
    random = ~extraversion_ij | cluster_id))
```

Does not converge either. Add iterations.

```
mod_imp_interaction <- with(
  imp,
  lme(
    popularity_ij ~ gender_ij + extraversion_ij + experience_j +
      extraversion_ij:experience_j,
    random = ~extraversion_ij | cluster_id,
    control = lmeControl(maxIter = 100)
    )
  )
```

```
pool(mod_imp_interaction) |> tidy(conf.int = TRUE)
```

| term | estimate | std.error | statistic | p.value | conf.low | cor |
|---|---|---|---|---|---|---|
| (Intercept) | -2.087 | 0.443 | -4.71 | 0 | -3.044 | |
| gender_ij | 1.140 | 0.079 | 14.44 | 0 | 0.954 | |
| extraversion_ij | 0.770 | 0.062 | 12.34 | 0 | 0.639 | |
| experience_j | 0.203 | 0.025 | 8.02 | 0 | 0.150 | |
| extraversion_ij:experience_j | -0.021 | 0.004 | -5.21 | 0 | -0.029 | |

```
mitml::testEstimates(as.mitml.result(mod_imp_interaction), extra
```

```
Call:

mitml::testEstimates(model =
as.mitml.result(mod_imp_interaction),
    extra.pars = TRUE)
```

```
Final parameter estimates and inferences obtained from 5
imputed data sets.

                            Estimate Std.Error   t.value
df   P(>|t|)       RIV       FMI
(Intercept)                   -2.087     0.443    -4.709
13.222     0.000     1.222     0.605
gender_ij                      1.140     0.079    14.437
7.236      0.000     2.899     0.794
extraversion_ij                0.770     0.062    12.344
19.835     0.000     0.815     0.497
experience_j                   0.203     0.025     8.023
17.460     0.000     0.918     0.530
extraversion_ij:experience_j  -0.021     0.004    -5.206
20.178     0.000     0.803     0.493

                                          Estimate
Intercept~~Intercept|cluster_id              0.528
extraversion_ij~~extraversion_ij|cluster_id  0.009
Intercept~~extraversion_ij|cluster_id       -0.049
Residual~~Residual                           0.675
ICC|cluster_id                               0.435

Unadjusted hypothesis test as appropriate in larger samples.
```

## 4.8 Compare estimates

```
est_true <-
  mod_true_interaction |> tidy() |> select(term, true = estimate
est_cca <-
  mod_cca_interaction |> tidy() |> select(cca = estimate, cca_se
est_imp <- pool(mod_imp_interaction) |> tidy() |> select(imp = e
cbind(est_true[1:5,], est_cca[1:5,], est_imp)
```

| term | true | true_se | cca | cca_se | imp | imp_se |
|---|---|---|---|---|---|---|
| (Intercept) | -1.207 | 0.269 | -2.119 | 0.308 | -2.087 | 0.443 |
| gender_ijgirl | 1.241 | 0.036 | 1.221 | 0.043 | 1.140 | 0.079 |
| extraversion_ij | 0.803 | 0.040 | 0.742 | 0.048 | 0.770 | 0.062 |
| experience_j | 0.226 | 0.017 | 0.204 | 0.019 | 0.203 | 0.025 |
| extraversion_ij:experience_j | -0.025 | 0.003 | -0.021 | 0.003 | -0.021 | 0.004 |

The estimates after imputation are more biased than the CCA estimates after list-wise deletion.

# 5 Discussion

This tutorial has outlined how to treat missingness in multilevel data using multiple imputation. Multiple imputation with the R package {mice} is a powerful tool to obtain valid estimates of multilevel model effects from incomplete data. Although the procedure of imputation is less effortless than the default missing data method—list-wise deletion—imputation will lead to less biased estimates and confidence-valid inferences. Moreover, data visualization may aid the analyst in building and evaluating imputation models. Since there is software available to facilitate the exploration-imputation-evaluation-analysis pipeline, it is worth the investment. Sometimes a sub-optimal solution has to be preferred, e.g. because of convergence issues. Still, doing something with the missingness is better than ignoring the incomplete cases altogether. In short, using multiple imputation to treat incomplete multilevel data is valid and easy with the right software tools.

# 6 References