Multiple Imputation of Missing Data in Simple and More Complex Settings

Nicole Erler

Department of Biostatistics, Erasmus MC
https://nerler.com

FGME 2019, Kiel
15 September, 2019

Erasmus MC
University Medical Center Rotterdam

Outline

Part I: Multiple Imputation

How does multiple imputation work?

- The ideas behind MI
- Understanding sources of uncertainty
- Implementation of MI and MICE

Part II: Multiple Imputation Workflow

How to perform MI with the **mice** package in R, from getting to know the data to the final results.

Practicals: imputation with **mice** & checking imputed data

Outline (cont.)

Part III: When MICE might fail

Introduction to

- settings where standard use of **mice** is problematic
- alternative imputation approaches
- alternative R packages

Practicals: Imputation with non-linear functional forms & multi-level outcomes

Part IV: Multiple Imputation Strategies

Some tips & tricks

Part I
Multiple Imputation

- Developed by **Donald B. Rubin** in the 1970s
- to handle missing values in **public use databases** (e.g., census data provided by the government),
- motivated by the **increase in missing values**, and
- increased **availability of computers**.

Goal: data should be usable by [10]

- a **large number of analysts**, who commonly have to rely on
- standard **software that can only handle complete data**, and usually
- are **not experts in handling incomplete data**.

Rubin's thoughts: [11]

One imputed value can not be correct in general.
➡ We need to represent missing values by a **number of imputations**.

Missing data has a distribution.

To find **sensible values** to fill in, we need some kind of **model**.

This **distribution depends on assumptions** that have been made about the model.

What we want is the
'predictive distribution' of the missing values given the observed values.

How to obtain that predictive distribution?

Idea: assume nonrespondents are just like respondents

- fit a model to the observed data
- obtain for each "nonrespondent" the conditional distribution of the missing data (given the observed data) as if he/she was a respondent

How to represent the multiple imputed values?

- for each set of imputed values, create a dataset
  (those datasets agree in the observed values but imputed values differ)
- analyse each dataset
- combine results from all analyses

➡ We can describe

- the overall results
- how (much) the **results vary between the imputed datasets**

- X: $n \times p$ data matrix with n rows and p variables $x_1, \ldots, x_p$
- X_{obs}: observed data, X_{mis}: missing data
- R: $n \times p$ missing indicator matrix containing 0 (missing) or 1 (observed)

$$\mathbf{X} = \begin{array}{c} \begin{array}{cccc} X_{-2} & X_2 & & X_{-2} \end{array} \\ \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{bmatrix} \end{array} \qquad \mathbf{R} = \begin{bmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,p} \\ R_{2,1} & R_{2,2} & \cdots & R_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ R_{n,1} & R_{n,2} & \cdots & R_{n,p} \end{bmatrix}$$

For example:

$$\mathbf{X} = \begin{array}{cccc} X_1 & X_2 & X_3 & X_4 \\ \hline \checkmark & \text{NA} & \checkmark & \checkmark \\ \checkmark & \checkmark & \text{NA} & \text{NA} \\ \checkmark & \text{NA} & \checkmark & \text{NA} \end{array} \quad \Rightarrow \mathbf{R} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Missing Completely At Random (MCAR)

$$p(R \mid X_{obs}, X_{mis}) = P(R)$$

Missingness is independent of all data

questionnaire got lost in mail

Missing At Random (MAR)

$$p(R \mid X_{obs}, X_{mis}) = P(R \mid X_{obs})$$

Missingness depends only on observed data

overweight participants are less likely to report their chocolate consumption (and we know their weight)

Missing Not At Random (MNAR)

$$p(R \mid X_{obs}, X_{mis}) \neq P(R \mid X_{obs})$$

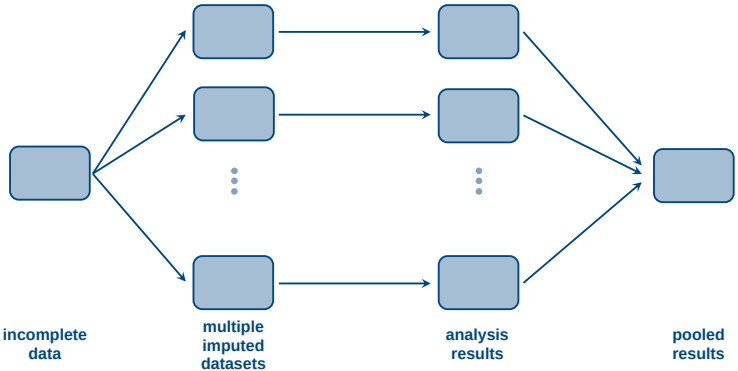
Missingness depends (also) on unobserved data

overweight participants are less likely to report their weight

- **MCAR** is a special case of **MAR**
- not possible to distinguish **MNAR** from **MAR** with just the observed data
- **Ignorability:**
  If **M(C)AR** and parameters in $p(R \mid X, \psi)$ are (a priori) independent of parameters in $p(X \mid \theta)$ ➡ missingness process does not need to be modelled
- Complete case analysis (mostly) only unbiased in **MCAR**

Here:

- we assume **MAR**,
- focus on missing values in covariates, and
- cross-sectional data (for now)



In summary:

1. **Imputation:** impute multiple times ➡ multiple completed datasets
2. **Analysis:** analyse each of the datasets
3. **Pooling:** combine results, taking into account additional uncertainty

How can we actually get imputed values?

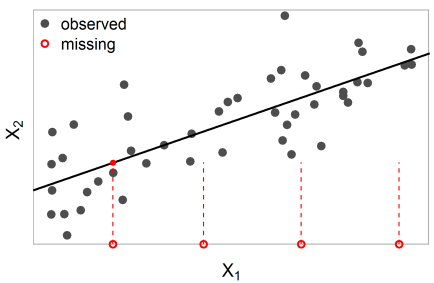
For now: assume only one continuous variable has missing values (**univariate missing data**).

X_1	X_2	X_3	X_4
✓	NA	✓	✓
✓	✓	✓	✓
✓	NA	✓	✓
⋮	⋮	⋮	⋮

Idea: Predict values

Model:
$x_{i2} = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i3} + \beta_3 x_{i4} + \varepsilon_i$

Imputed/predicted value:
$\hat{x}_{i2} = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i3} + \hat{\beta}_3 x_{i4}$



Problem:

- We can obtain **only one imputed value** per missing value, but we wanted a whole distribution.
- The predicted values do not take into account the added **uncertainty** due to the missing values.

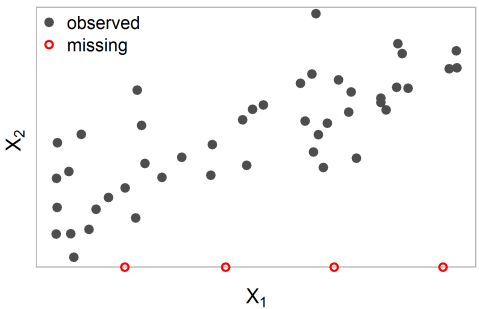
➡ We need to take into account **two sources of uncertainty**:

- The **parameters** are estimated with **uncertainty** (represented by the std. error).
- There is **random variation / prediction error** (variation of the residuals).

Taking into account uncertainty about the parameters β:
We assume that β **has a distribution**, and we can sample realizations of β from that distribution.

When plugging the different realizations of β into the predictive model, we obtain **slightly different regression lines**.

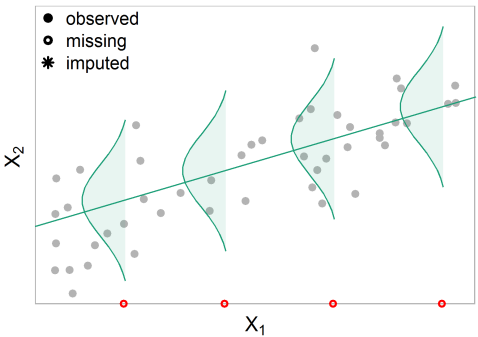
With each set of coefficients, we also get slightly **different predicted values**.



Taking into account the prediction error:
The model does not fit the data perfectly: observations are scattered around the regression lines.

We assume that the **data have a distribution**, where

- the **mean** for each value is given by the **predictive model**, and
- the **variance** is determined by the variance of the residuals ε.



In the end, we obtain one imputed dataset for each color.

Assumption of **distributions** for **parameters** and **missing values**:

Bayesian

Alternative to take into account **uncertainty in parameters**:

Bootstrap

Both require the assumption of a **distribution for the missing values**.

What if none of the standard distribution fits?

Predictive Mean Matching (PMM)

- semi-parametric approach to imputation
- developed for settings where the normal distribution is not a good choice for the predictive distribution [7, 8]

Idea:

- **find cases** in the observed data that are **similar** to the cases with missing values
- **fill in** the missing value with the **observed value** from one of those cases

To find similar cases, the **predicted values** of complete and incomplete cases are compared.

The steps in PMM:

1. Obtain parameter estimates for $\hat{\boldsymbol{\beta}}$ and $\hat{\sigma}$.
2. Calculate the predicted values for the observed cases

$$\hat{\mathbf{y}}_{obs} = \mathbf{X}_{obs}\hat{\boldsymbol{\beta}}.$$

3. Calculate the predicted value for the missing cases

$$\hat{\mathbf{y}}_{mis} = \mathbf{X}_{mis}\hat{\boldsymbol{\beta}}.$$

4. For each missing value, find d donor candidates that fulfill a given criterium.
5. Randomly select one of the donors.

Several **criteria to select donors** have been proposed:

1. The donor is the **(one) case with the smallest absolute difference**
2. Donor candidates are the d **cases with the smallest absolute difference**. The donor is selected randomly from the candidates.
3. Donor candidates are those cases for which the **absolute difference is smaller than some limit** η. The donor is selected randomly from the candidates.
4. Select candidates like in 2. or 3., but select the donor from the candidates with probability that depends on the absolute difference.[14]

Potential issues with donor selection

- If the same donor is chosen in many/all imputations (e.g., because only a few similar observed cases are available), the **uncertainty about the missing values will be underestimated**.
- Therefore, using one donor is not a good idea. On the other hand, using too many candidates can lead to bad matches.

➡ PMM may be **problematic** when

- the **dataset is very small**,
- the **proportion of missing values is large**, or
- one/some **predictor variable(s) are strongly related to the missingness**.

For the **sampling of the parameters** (step 1 on slide 17), different approaches have been introduced in the literature:

Type-0	$\hat{\beta}_{LS/ML}$ (least squares or maximum likelihood) are used in both prediction models
Type-I	$\hat{\beta}_{LS/ML}$ to predict $\hat{y}_{obs}$; $\tilde{\beta}_{B/BS}$ (Bayesian or bootstrap) to predict $\hat{y}_{mis}$
Type-II	$\tilde{\beta}_{B/BS}$ to predict $\hat{y}_{obs}$ as well as $\hat{y}_{mis}$
Type-III	different draws $\tilde{\beta}^{(1)}_{B/BS}$ and $\tilde{\beta}^{(2)}_{B/BS}$ to predict $\hat{y}_{obs}$ and $\hat{y}_{mis}$, respectively

The use of Type-0 and Type-I matching **underestimates the uncertainty** about the regression parameters.

Another point to consider:
the **choice of the set of data used to train the prediction models**.

In the version presented on slide 17, the same set of data (all cases with observed y) is used to train the model and to produce predicted values of y_{obs}.

The predictive model will likely fit the observed cases better than the missing cases, and, hence, **variation will be underestimated**.

Alternatives:

- the **model could be trained on the whole data** (using previously imputed values)
- use a **leave-one-out approach** on the observed data

mice (in R):

- **PMM** via `mice.impute.pmm()`
  - specification of number of donors d (same for all variables)
  - Type-0, Type-I, Type-II matching
- **PMM** via `mice.impute.midastouch()`
  - allows leave-one-out estimation of the parameters
  - distance based donor selection
  - Type-0, Type-I, Type-II matching
- **bootstrap** linear regression via `mice.impute.norm.boot()`
- **bootstrap** logistic regression via `mice.impute.logreg.boot()`
- **Bayesian** linear regression via `mice.impute.norm()`
- ...

Multivariate missing data:
What if we have **missing values in more than one variable**?

In case of **monotone missing values** we can use the technique for univariate missing data in a chain:
impute x_4 given x_1
impute x_3 given x_1 and x_4
impute x_2 given x_1, x_4 and x_3

X_1	X_4	X_3	X_2
✓	✓	✓	NA
✓	✓	NA	NA
✓	NA	NA	NA
⋮	⋮	⋮	⋮

When we have **non-monotone missing data** there is no sequence without conditioning on unobserved values.

X_1	X_2	X_3	X_4
✓	NA	✓	✓
NA	✓	NA	NA
✓	NA	✓	NA
⋮	⋮	⋮	⋮

There are **two popular approaches** for the imputation step in **multivariate non-monotone** missing data:

Fully Conditional Specification

- Multiple Imputation using Chained Equations (**MICE**)
- sometimes also: sequential regression
- implemented in SPSS, R, Stata, SAS, ...
- our focus here

Joint Model Imputation

(see for example Carpenter & Kenward [2])

Algorithm 1 MICE algorithm [15] for **one** imputed dataset

for j in $1, \ldots, p$: ▷ Setup
  Specify imputation model for variable X_j
  $p(X_j^{mis} \mid X_j^{obs}, X_{-j}, R)$
  Fill in starting imputations $\dot{X}_j^0$ by random draws from X_j^{obs}.
end for

for t in $1, \ldots, T$: ▷ loop through iterations
  for j in $1, \ldots, p$: ▷ loop through variables
    Define currently complete data except X_j
    $\dot{X}_{-j}^t = \left(\dot{X}_1^t, \ldots, \dot{X}_{j-1}^t, \dot{X}_{j+1}^{t-1}, \ldots, \dot{X}_p^{t-1}\right)$.
    Draw parameters $\dot{\theta}_j^t \sim p(\theta_j^t \mid X_j^{obs}, \dot{X}_{-j}^t, R)$.
    Draw imputations $\dot{X}_j^t \sim p(X_j^{mis} \mid \dot{X}_{-j}^t, R, \dot{\theta}_j^t)$.
  end for
end for

The imputed values from the **last iteration**,

$$\left(\dot{X}_1^T, \ldots, \dot{X}_p^T\right),$$

are then used to replace the missing values in the original data.

One run through the algorithm ➡ one imputed dataset.

➡ To obtain m imputed datasets: **repeat m times**

- The **sequence of imputations** for one missing value (from starting value to final iteration) is called a **chain**.
- Each run through the MICE algorithm produces one chain per missing value.

Why iterations?

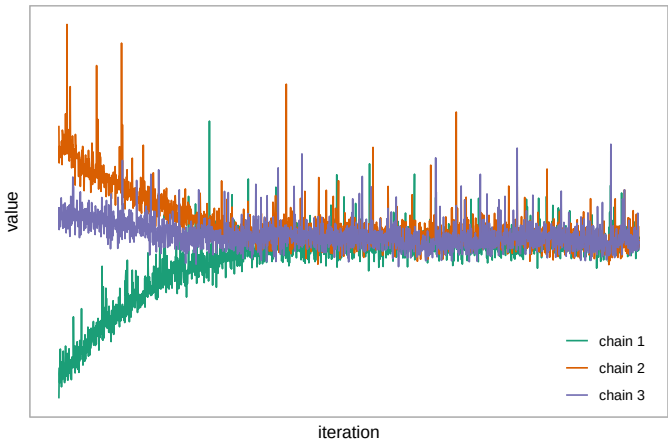
- Imputed values in one variable depend on the imputed values of the other variables (Gibbs sampling).
- If the starting values (random draws) are far from the actual distribution, imputed values from the first few iterations are not draws from the distribution of interest.

How many iterations?

Until **convergence**
= when the sampling distribution does not change any more
(Note: the imputed value will still vary between iterations.)

How to evaluate convergence?

The **traceplot** (x-axis: iteration number, y-axis: imputed value) should show a horizontal band.



Each chain is the sequence of imputed values (from starting value to final imputed value) for the same missing value.

3. Analysis step

Multiple imputed datasets:

X_1	X_2	X_3	X_4
1.4	9.2	1.8	2.0
0.5	12.4	2.3	0.1
-0.5	10.7	2.6	-1.6
⋮	⋮	⋮	⋮

X_1	X_2	X_3	X_4
1.4	13.3	1.8	2.0
0.5	12.4	2.1	0.6
-0.5	10.2	2.6	-1.7
⋮	⋮	⋮	⋮

X_1	X_2	X_3	X_4
1.4	10.0	1.8	2.0
0.5	12.4	2.2	-1.4
-0.5	8.6	2.6	-1.0
⋮	⋮	⋮	⋮

Analysis model of interest, e.g.,

$$x_1 = \beta_0 + \beta_1 x_2 + \beta_2 x_3 + \beta_3 x_4$$

Multiple sets of results:

	est.	se
β_0	-0.15	0.22
β_1	0.16	0.02
β_2	-0.59	0.03
β_3	0.28	0.03

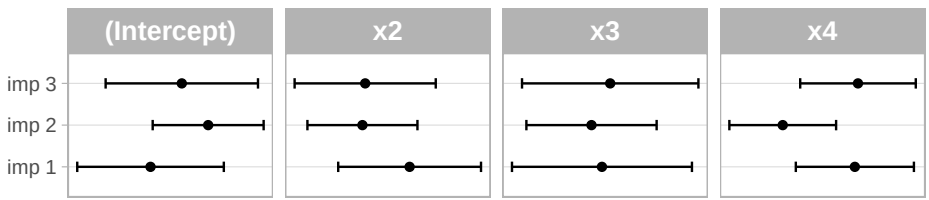
	est.	se
β_0	0.19	0.16
β_1	0.14	0.01
β_2	-0.59	0.03
β_3	0.2	0.03

	est.	se
β_0	0.04	0.22
β_1	0.14	0.01
β_2	-0.58	0.03
β_3	0.28	0.03

4. Pooling

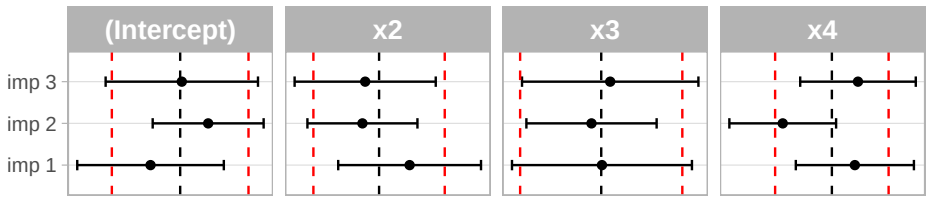
In the results from multiply imputed data there are **two types of variation/uncertainty**:

- **within** imputation (represented by the confidence intervals)
- **between** imputation (horizontal shift between imputations)



4. Pooling

To summarize the results, we can take the mean of the results from the separate analyses. This is the **pooled point estimate**.



But does the same work for the standard error (or bounds of the CIs)?

The averaged CI's (marked in red) underestimate the total variation (within + between).

4. Pooling

The most commonly used method to pool results from analyses of multiply imputed data was introduced by Rubin [9], hence **Rubin's Rules**.

Notation:
m: number of imputed datasets
Q_ℓ: quantity of interest (e.g., regr. parameter β) from ℓ-th imputation
U_ℓ: variance of Q_ℓ (e.g., $var(\beta) = se(\beta)^2$)

Pooled parameter estimate:

$$\bar{Q} = \frac{1}{m} \sum_{\ell=1}^{m} \hat{Q}_\ell$$

The **variance** of the pooled parameter estimate is calculated from the **within and between imputation variance**.

Average within imputation variance:

$$\bar{U} = \frac{1}{m} \sum_{\ell=1}^{m} \hat{U}_\ell$$

Between imputation variance:

$$B = \frac{1}{m-1} \sum_{\ell=1}^{m} \left(\hat{Q}_\ell - \bar{Q} \right)^T \left(\hat{Q}_\ell - \bar{Q} \right)$$

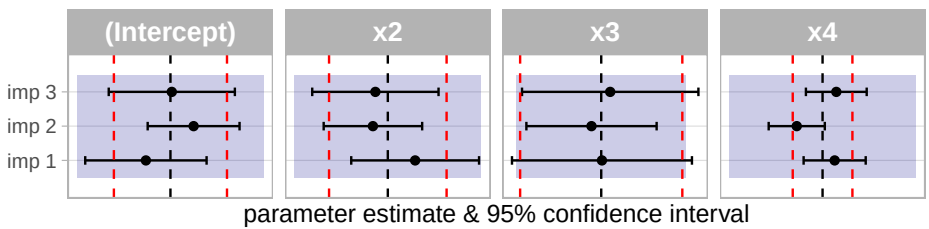
Total variance:

$$T = \bar{U} + B + B/m$$

The $(1-\alpha)$ **100% confidence interval** is then

$$\bar{Q} \pm t_\nu(\alpha/2)\sqrt{T},$$

where t_ν is the $\alpha/2$ quantile of the t distribution with $\nu = (m-1)\left(1+r_m^{-1}\right)^2$ degrees of freedom [1], where $r_m = \frac{(B+B/m)}{\bar{U}}$ is the relative increase in variance that is due to the missing values.



[1] Barnard et al. [1] proposed an improvement to calculate the degrees of freedom. This improved version is implemented in the **mice** package.

Part II
Multiple Imputation Workflow

To demonstrate the work flow when performing multiple imputation with the **mice** package, we use data from the National Health and Nutrition Examination Survey (NHANES).

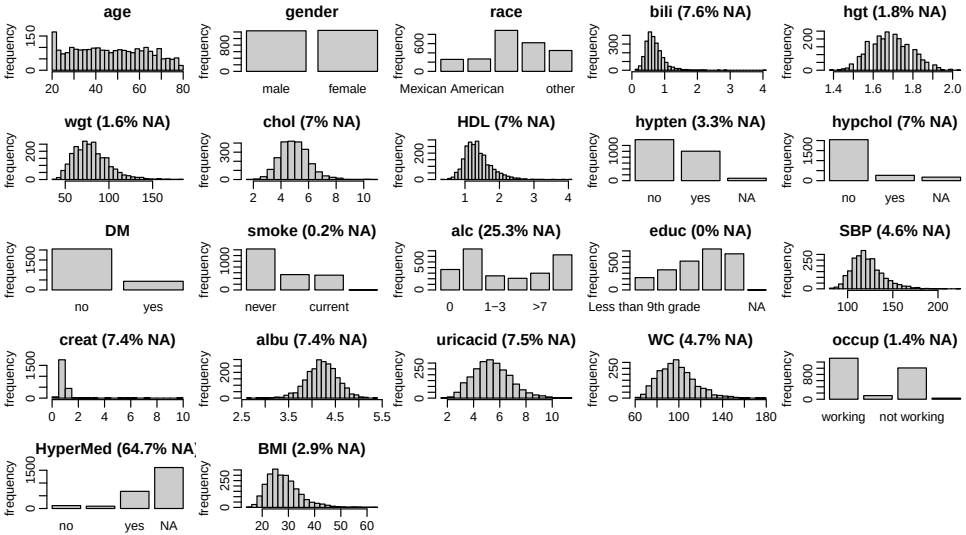
There are several packages in R that provide functions to visualize incomplete data.

Examples are:
naniar, **VIM**, **visdat**, **mice**, **JointAI**, **Amelia**, ...

5. Know your data

5.2. Data distributions

```
JointAI::plot_all(NHANES, nclass = 30)
```



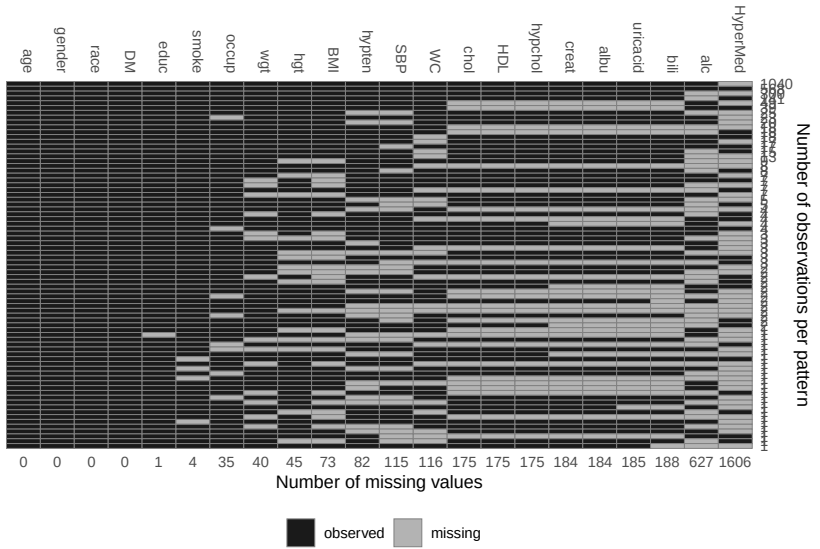
```
mdp <- mice::md.pattern(NHANES, plot = FALSE)
head(mdp[, -c(6:14)]) # omit some columns to fit it on the slide
```

```
##      age gender race DM educ HDL hypchol creat albu uricacid bili alc HyperMed
## 568    1      1    1  1    1   1       1     1    1         1    1   1         1 0
## 1040   1      1    1  1    1   1       1     1    1         1    1   1         0 1
## 141    1      1    1  1    1   1       1     1    1         1    1   0         1 1
## 300    1      1    1  1    1   1       1     1    1         1    1   0         0 2
## 2      1      1    1  1    1   1       1     1    1         1    0   1         0 2
## 1      1      1    1  1    1   1       1     1    1         1    0   0         0 3
```

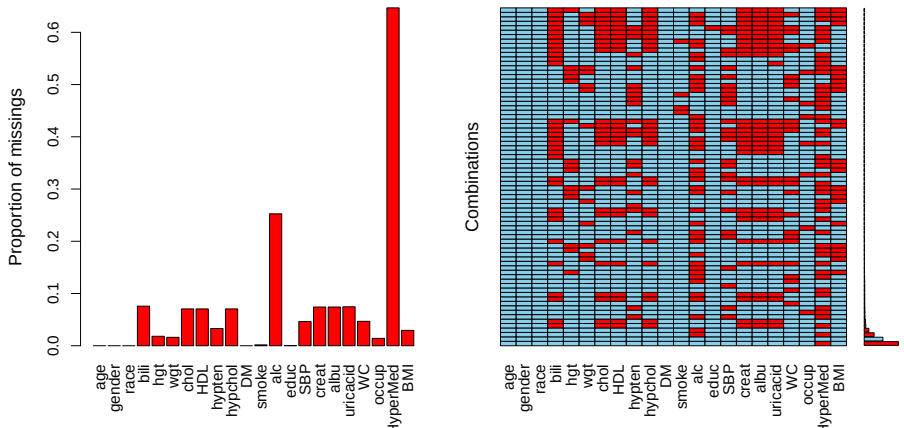
```
tail(mdp[, -c(6:14)])
```

```
##   age gender race DM educ HDL hypchol creat albu uricacid bili alc HyperMed
## 1   1      1    1  1    1   1       1     1    1         1    1   1         1    1
## 1   1      1    1  1    1   1       1     1    1         1    1   1         0    2
## 1   1      1    1  1    1   0       0     0    0         0    0   0         0   10
## 1   1      1    1  1    1   1       1     1    1         1    1   1         0    4
## 1   1      1    1  1    0   0       0     0    0         0    0   1         0   12
##     0      0    0  0    1 175     175   184  184       185  188 627      1606 4010
```

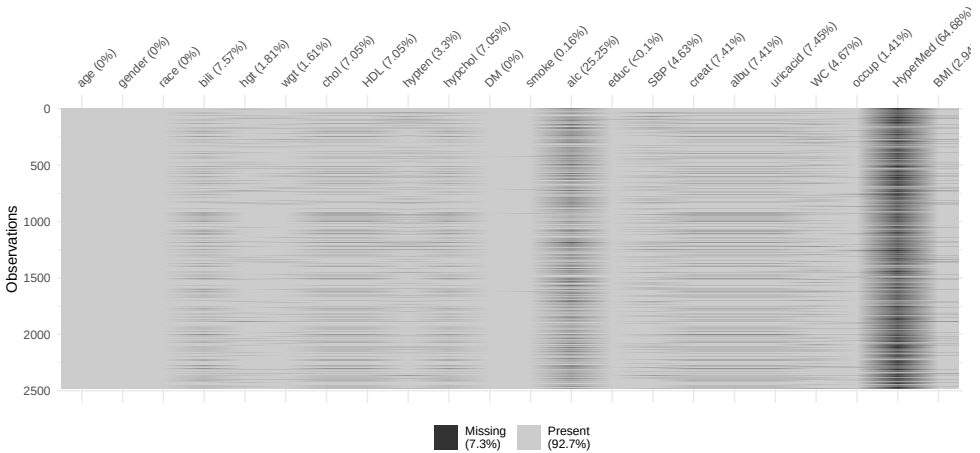
```
JointAI::md_pattern(NHANES)
```



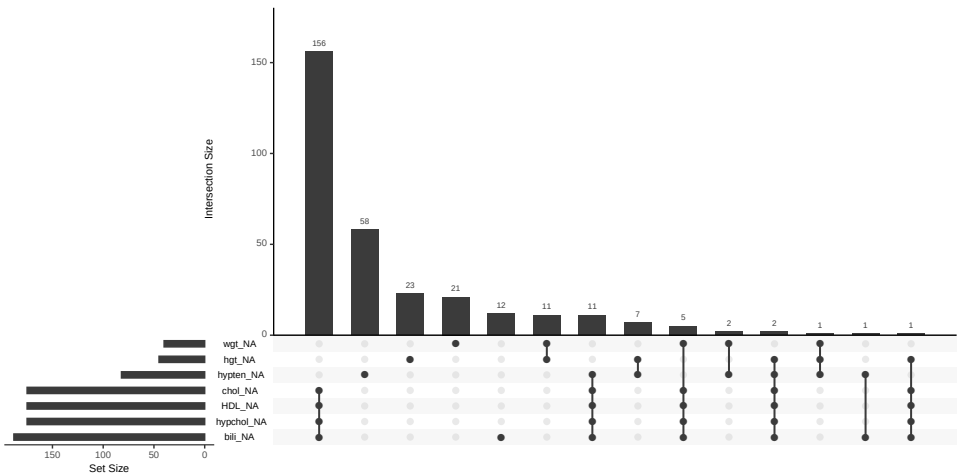
```
VIM::aggr(NHANES, prop = TRUE, numbers = FALSE)
```



```
naniar::vis_miss(NHANES)
```



```
naniar::gg_miss_upset(NHANES[, 1:10], nsets = 10)
```



We are also interested in the number and proportion of (in)complete cases ...

```
cbind(
  "#" = table(ifelse(complete.cases(NHANES), 'incompl.', 'complete')),
  "%" = round(100 * table(complete.cases(NHANES))/nrow(NHANES), 2)
)
```

```
##              #     %
## complete  1915 77.12
## incompl.   568 22.88
```

... and the proportion of missing values per variable:

```
cbind("# NA" = sort(colSums(is.na(NHANES))),
      "% NA" = round(sort(colMeans(is.na(NHANES))) * 100, 2))
# see also: naniar::miss_var_summary()
```

```
##         # NA % NA
## age        0 0.00
## gender     0 0.00
## race       0 0.00
## DM         0 0.00
## educ       1 0.04
## smoke      4 0.16
## occup     35 1.41
## wgt       40 1.61
## hgt       45 1.81
## BMI       73 2.94
## hypten    82 3.30
```

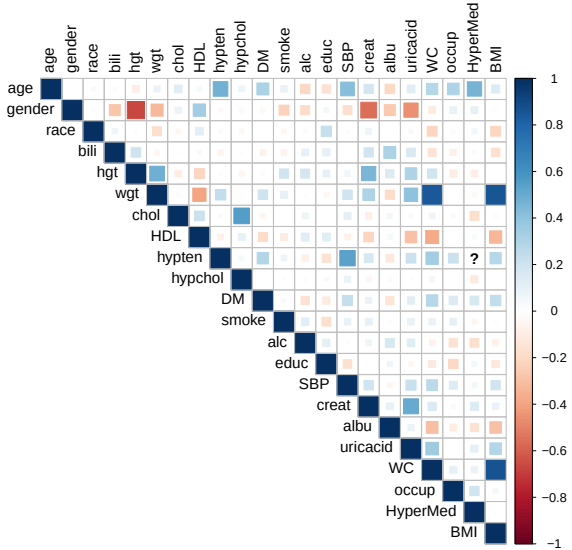
```
##           # NA  % NA
## SBP        115  4.63
## WC         116  4.67
## chol       175  7.05
## HDL        175  7.05
## hypchol    175  7.05
## creat      184  7.41
## albu       184  7.41
## uricacid   185  7.45
## bili       188  7.57
## alc        627 25.25
## HyperMed  1606 64.68
```

A quick (and dirty) way to check for strong correlations between variables is:

```
# re-code all variables as numeric and calculate spearman correlation
Corr <- cor(sapply(NHANES, as.numeric),
            use = "pairwise.complete.obs", method = "spearman")

## Warning in cor(sapply(NHANES, as.numeric), use =
"pairwise.complete.obs", :  the standard deviation is zero
```

```
corrplot::corrplot(Corr, method = "square", type = "upper",
                   tl.col = "black")
```

Note: We only use the correlation coefficient for categorical variables for visualization, not as a statistical result!


age
gender
race
bili
hgt
wgt
chol
HDL
hypten
hypchol
DM
smoke
alc
educ
SBP
creat
albu
uricacid
WC
occup
HyperMed
BMI
?
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

Check out what the problem is with `hypertension` and `HyperMed`:

```
table(hypertension = NHANES$hypten,
      HyperMed = NHANES$HyperMed, exclude = NULL)

##              HyperMed
## hypertension   no previous  yes <NA>
##         no      0        0    0 1397
##         yes   114       90  673  127
##         <NA>    0        0    0   82
```

Knowing your data also means being able to answer these questions:

- Do missing values in multiple variables always **occur together**? (e.g. blood measurements)
- Are there **structural missing values**? (e.g. pregnancy status in men)
- Are there **patterns** in the missing values? (e.g. only patients with hypertension have observations of `HyperMed`)
- Are values **missing by design**?
- Is the **assumption of ignorable missingness** (MAR or MCAR) justifiable?

Auxiliary variables are variables that are not part of the analysis but **can help during imputation**.

Good auxiliary variables [15]

- are **related to the probability of missingness** in a variable, or
- are **related to the incomplete variable** itself,
- do **not have many missing values** themselves and
- are (mostly) **observed** when the incomplete variable of interest is missing.

The main arguments needed to impute data with `mice()` are:

- `data`: the dataset
- `m`: number of imputed datasets (default is 5)
- `maxit`: number of iterations (default is 5)
- `method`: vector of imputation methods
- `defaultMethod`: vector of default imputation methods for numerical, binary, unordered and ordered factors with > 2 levels (default is `c("pmm", "logreg", "polyreg", "polr")`)
- `predictorMatrix`: matrix specifying roles of variables

mice has implemented many **imputation methods**, the most commonly used ones are:

- `pmm`: predictive mean matching (any)
- `norm`: Bayesian linear regression (numeric)
- `logreg`: binary logistic regression (binary)
- `polr`: proportional odds model (ordered factors)
- `polyreg`: polytomous logistic regression (unordered factors)

Change the default imputation method:
Example: To use `norm` instead of `pmm` for all continuous incomplete variables, use:

```
mice(NHANES, defaultMethod = c("norm", "logreg", "polyreg", "polr"))
```

Change imputation method for a single variable:
To change the imputation method for single variables (but also for changes in other arguments) it is convenient to **do a setup run** of `mice()` without iterations (`maxit = 0`) and to extract and modify the parameters from there.

Exclude variable from imputation:
When a variable that has missing values should not be imputed, the method needs to be set to `""`.

```
library("mice")
imp0 <- mice(NHANES, maxit = 0)
meth <- imp0$method
meth
```

```
##       age    gender      race      bili       hgt       wgt
##        ""        ""        ""     "pmm"     "pmm"     "pmm"
##      chol       HDL    hypten   hypchol        DM     smoke
##     "pmm"     "pmm"  "logreg"  "logreg"        ""    "polr"
##       alc      educ       SBP     creat     albu  uricacid
##    "polr" "polyreg"     "pmm"     "pmm"     "pmm"     "pmm"
##        WC     occup  HyperMed       BMI
##     "pmm" "polyreg"    "polr"     "pmm"
```

```
meth["albu"] <- "norm"
meth["HyperMed"] <- ""
# imp <- mice(NHANES, method = meth)
```

The predictorMatrix is a matrix that specifies **which variables are used as predictors** in which imputation model.
Each row represents the model for the variable given in the rowname.

```
head(imp0$predictorMatrix)[, 1:11]

##         age gender race bili hgt wgt chol HDL hypten hypchol DM
## age       0      1    1    1   1   1    1   1      1       1  1
## gender    1      0    1    1   1   1    1   1      1       1  1
## race      1      1    0    1   1   1    1   1      1       1  1
## bili      1      1    1    0   1   1    1   1      1       1  1
## hgt       1      1    1    1   0   1    1   1      1       1  1
## wgt       1      1    1    1   1   0    1   1      1       1  1
```

Variables **not used as predictor** are (or have to be set to) **zero**.

By **default, all variables** (except the variable itself) **are used** as predictors.

Important:
A variable that has **missing values needs to be imputed** in order to be used as a predictor for other imputation models!!!

Note:
By default, **ALL** variables with missing values are imputed and **ALL** variables are used as predictor variables.

➡ Make sure to adjust the `predictorMatrix` and `method` to avoid using ID variables or other columns of the data that should not be part of the imputation.

➡ Make sure all **variables are coded correctly**, so that the automatically chosen imputation models are appropriate.

```r
library(mice)
# setup-run
imp0 <- mice(NHANES, maxit = 0,
             defaultMethod = c("norm", "logreg", "polyreg", "polr"))

# adjust imputation methods
meth <- imp0$method
meth["educ"] <- "polr"
meth["HyperMed"] <- ""

# adjust predictor matrix
pred <- imp0$predictorMatrix
pred[, "HyperMed"] <- 0

# run imputation with adjusted settings
imp <- mice(NHANES, method = meth, predictorMatrix = pred,
            printFlag = FALSE)
```

In some cases, variables are **functions of other variables**, e.g., $BMI = \frac{wgt}{hgt^2}$.

If we impute BMI directly, its values may be **inconsistent** with the (imputed) values of hgt and wgt.

```
DF1 <- complete(imp, 1) # select the first imputed dataset
round(cbind("wgt/hgt^2" = DF1$wgt/DF1$hgt^2,
            BMI = DF1$BMI)[is.na(NHANES$BMI), ], 2)[1:5, ]

##      wgt/hgt^2   BMI
## [1,]     23.87 25.91
## [2,]     28.75 27.95
## [3,]     23.73 21.67
## [4,]     25.25 24.95
## [5,]     27.43 26.58
```

The imputed values of BMI are impossible given the corresponding values of hgt and wgt.

Moreover, if some components of a variable are observed we want to use that **information to reduce uncertainty**.

```
table(wgt_missing = is.na(NHANES$wgt),
      hgt_missing = is.na(NHANES$hgt))

##            hgt_missing
## wgt_missing FALSE TRUE
##       FALSE  2410   33
##       TRUE     28   12
```

Here we have $33 + 28 = 61$ cases in which either `hgt` or `wgt` is observed.

We would like to impute `hgt` and `wgt` separately and calculate `BMI` from the (imputed) values of the two variables.

If BMI is not a relevant predictor in any of the other imputation models, we could just exclude BMI from the imputation and **re-calculate it afterwards**.

To use BMI as predictor in the imputation, it has to be **calculated in each iteration** of the algorithm. In **mice** this is possible with **passive imputation**.

Instead of using a standard imputation method, we can specify a formula to calculate BMI:

```
meth["BMI"] <- "~I(wgt/hgt^2)"      # formula to impute BMI
pred[c("wgt", "hgt"), "BMI"] <- 0   # prevent feedback
```

To **prevent feedback** from BMI in the imputation of hgt and wgt the predictorMatrix needs to be modified.

Since BMI depends on wgt, and the two variables are highly correlated (ρ =0.87) it may be beneficial **not to use them simultaneously** as predictors in the other imputation models.
Which one to use may differ between imputation models.

Passive imputation can also be useful in settings where

- imputation models include **interaction terms** between incomplete variables (see [15, p. 133] for an example), or when
- a number of covariates is used to form a **sum score**. The sum score, instead of all single elements, can then be used as predictor in other imputation models.

`mice()` has an argument `post` that can be used to specify functions that modify imputed values.

Helpful functions are

- `squeeze()` to censor variables at given boundaries
- ~~`ifdo()` for conditional manipulation~~ (not yet implemented)

Example:
When inspecting the imputed values from `imp`, we find that some imputed values in `creat` are negative.

```
# DF1 is the first imputed dataset we extracted earlier
summary(DF1$creat)

##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.2829  0.7000  0.8400  0.8882  0.9900  9.5100
```

With the following syntax all imputed values of creat that are outside the interval c(0, 100) will be **set to those limiting values**.

```
post <- imp$post
post["creat"] <- "imp[[j]][,i] <- squeeze(imp[[j]][,i], c(0, 100))"
imp2 <- update(imp, post = post, maxit = 20, seed = 123)
```

Note:
When many observations are outside the limits it may be better to **change the imputation model** since the implied **assumption of the imputation model** apparently **does not fit the** (assumption about the) **complete data distribution**.

This **post-processing** of imputed values allows for many **more data manipulations** and is not restricted to `squeeze()` (and `ifdo()`).

Any strings of R commands provided will be evaluated after the corresponding variable is imputed, within each iteration.

For example, if subjects with SBP > 140 should be classified as hypertensive:

```
post["hypten"] <- "imp[[j]][p$data[where[, j], 'SBP'] > 140, i] <- 'yes'"
```

This also allows for (some) **MNAR scenarios**, for example, by multiplying or adding a constant to the imputed values, or to re-impute values depending on their current value.

When the **post-processed or passively imputed values** of a variable depend on other variables, the **sequence in which the variables are imputed** may be important to obtain **consistent values**.

Example:
If BMI is passively imputed (calculated) before the new imputations for hgt and wgt are drawn, the resulting values of BMI, will match hgt and wgt from the **previous iteration**, but not the iteration given in the imputed dataset.

In mice() the argument visitSequence specifies in which order the columns of the data are imputed. By default mice() imputes in the order of the columns in data.

```
visitSeq <- imp2$visitSequence
visitSeq
```

```
##  [1] "age"       "gender"    "race"      "bili"      "hgt"
##  [6] "wgt"       "chol"      "HDL"       "hypten"    "hypchol"
## [11] "DM"        "smoke"     "alc"       "educ"      "SBP"
## [16] "creat"     "albu"      "uricacid" "WC"        "occup"
## [21] "HyperMed" "BMI"
```

Currently, `hypten` is imputed before `SBP`, but the imputed values of `hypten` are post-processed depending on the current value of `SBP`. To get consistent values of these two variables, we need to change the `visitSequence`.

```
visitSeq <- c(visitSeq[-which(visitSeq == "hypten")],
              "hypten")
visitSeq
```

```
##  [1] "age"      "gender"   "race"     "bili"     "hgt"      "wgt"
##  [7] "chol"     "HDL"      "hypchol"  "DM"       "smoke"    "alc"
## [13] "educ"     "SBP"      "creat"    "albu"     "uricacid" "WC"
## [19] "occup"    "HyperMed" "BMI"      "hypten"
```

The `visitSequence` may specify that a column is visited multiple times during one iteration. All incomplete variables must be visited at least once.

`mice()` performs some **pre-processing** and **removes**

- incomplete variables that are not imputed but are specified as predictors,
- constant variables, and
- collinear variables.

In each iteration

- linearly dependent variables are removed and
- `polr` imputation models that do not converge are replaced by `polyreg`.

Why?
To avoid problems in the imputation models.

As a **consequence**

- imputation models may differ from what the user has specified or assumes is happening, or
- variables that should be imputed are not.

➡ Know your data
➡ Make sure `method` and `predictorMatrix` are specified appropriately
➡ Check the output and log of these automatic actions carefully

Practical

To practice the content of the previous section find the instructions for the practical here:

https://nerler.com/teaching/fgme2019/mimice

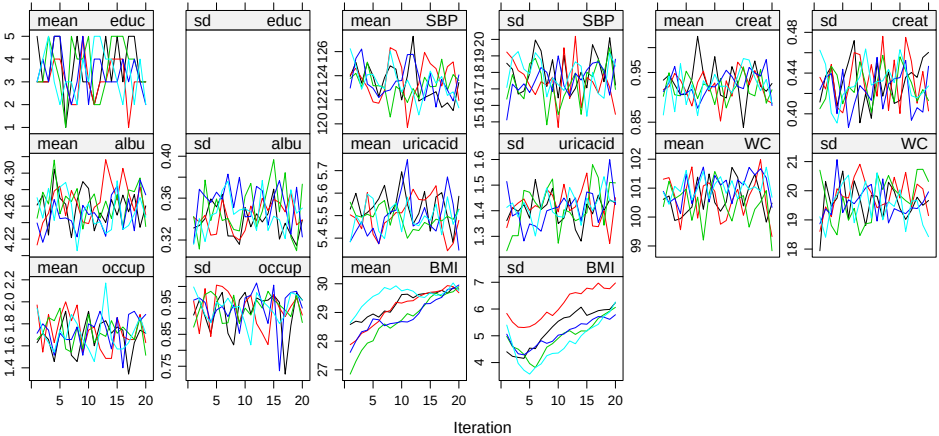
The log of the automatic changes is returned as part of the `mids` object:

```
demo <- NHANES[, 1:5]
demo$dupl <- demo[, 4]
demo$const <- 1
demo$age[demo$gender == 'male'] <- NA

demoimp <- mice(demo)
head(demoimp$loggedEvents)
```

```
## Warning:  Number of logged events:  8

##   it im dep      meth          out
## 1  0  0      constant        const
## 2  0  0     collinear         dupl
## 3  1  1 age       pmm genderfemale
## 4  1  2 age       pmm genderfemale
## 5  1  3 age       pmm genderfemale
## 6  2  1 age       pmm genderfemale
```

With columns

`it`	iteration number
`im`	imputation number
`dep`	dependent variable
`meth`	imputation method used
`out`	names of altered or removed predictors


mean educ
sd educ
mean SBP
sd SBP
mean creat
sd creat
mean albu
sd albu
mean uricacid
sd uricacid
mean WC
sd WC
mean occup
sd occup
mean BMI
sd BMI
Iteration

Strong trends and traces that show **correlation** between variables indicate **problems of feedback**. This needs to be investigated and resolved in the specification of the `predictorMatrix`.

Weak trends may be artefacts that often disappear when the imputation is performed with more iterations.

When MCMC chains have converged, the **distributions of the imputed and observed values** can be compared to investigate differences between observed and imputed data.

Note:
Plots usually show the **marginal** distributions of observed and imputed values, which do not have do be identical under MAR.

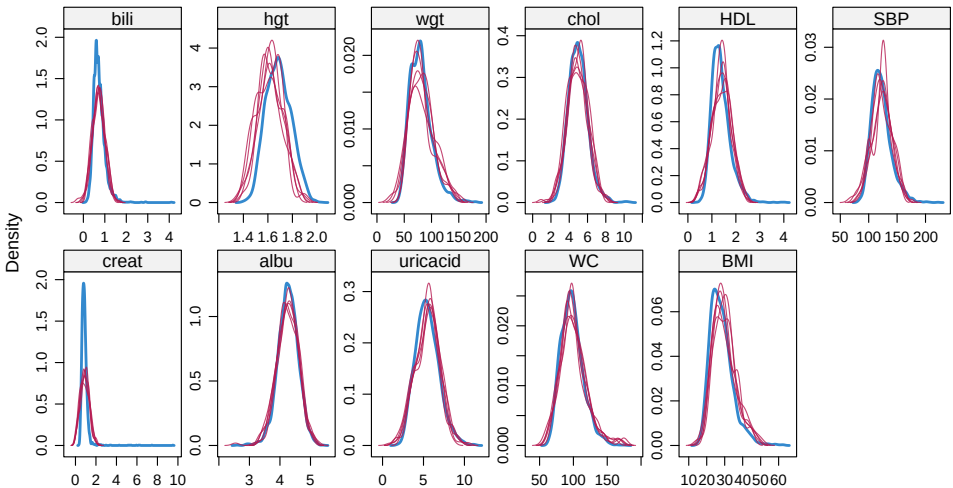
Recall:
The **conditional** distributions (given all the other variables in the imputation model) of the imputed values are assumed to be the same as the conditional distributions of the observed data.

mice provides several functions for visual diagnosis of imputed values:

- `densityplot()` (for large datasets and variables with many NAs)
- `stripplot()` (for smaller datasets and/or variables with few NAs)
- `bwplot()`
- `xyplot()`

These functions create lattice graphics, which can be modified analogously to their parent functions from the **lattice** package.

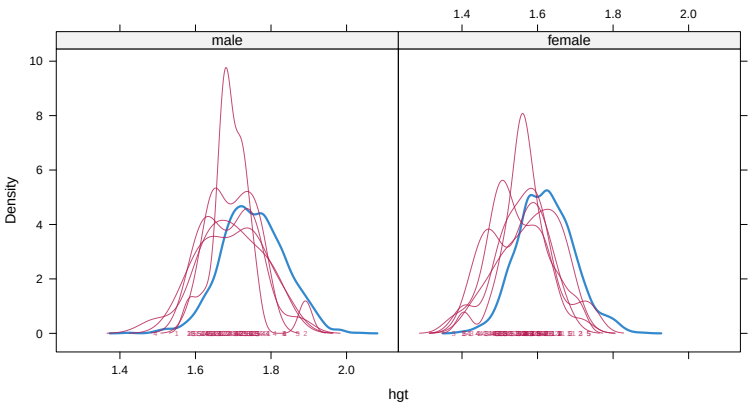
```
densityplot(imp2)
```



The `densityplot()` shows that the distribution of imputed values of `creat` is wider than the distribution of the observed values and that imputed values of `hgt` are smaller than the observed values.

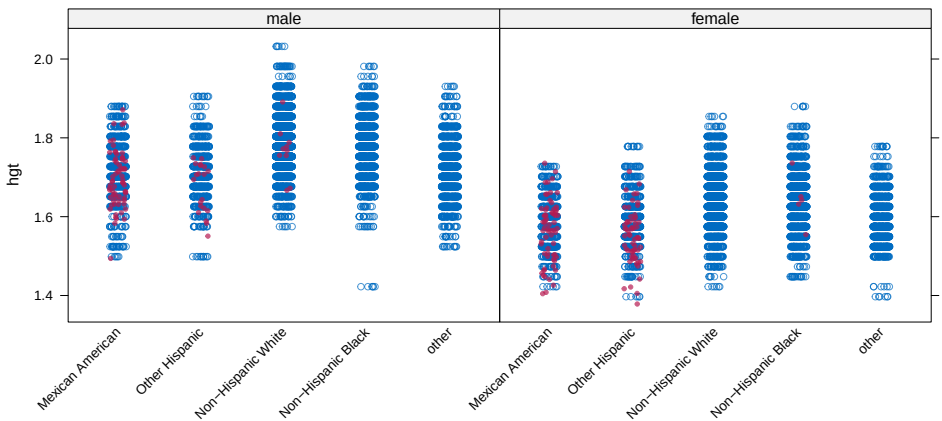
In some cases, differences in distributions can be explained by strata in the data, however, here, `gender` does not explain the difference in observed and imputed values.

```
densityplot(imp2, ~hgt|gender, plot.points = TRUE)
```



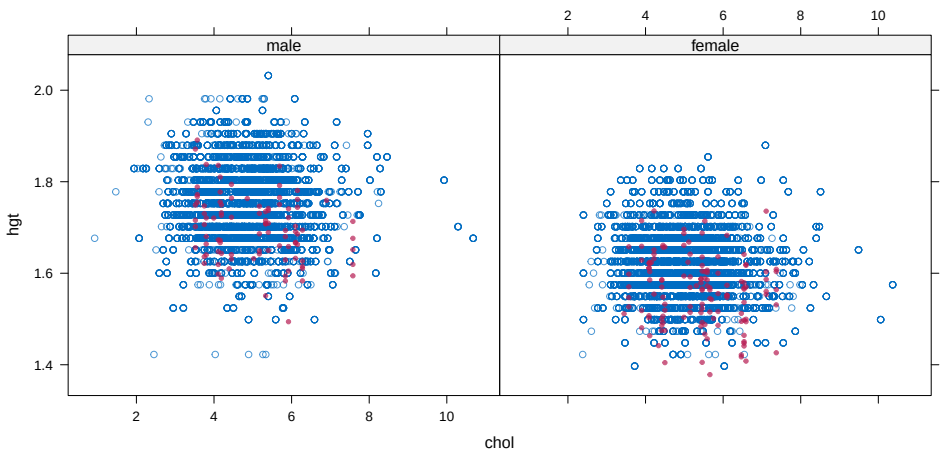
For (combinations of) variables with very few missing values a `stripplot()` may be better suited. Here we can also split the data for `gender` and `race`.

```
stripplot(imp2, hgt ~ race|gender, pch = c(1, 20),
          scales = list(x = list(rot = 45)))
```



The function `xyplot()` allows multivariate investigation of the imputed versus observed values.

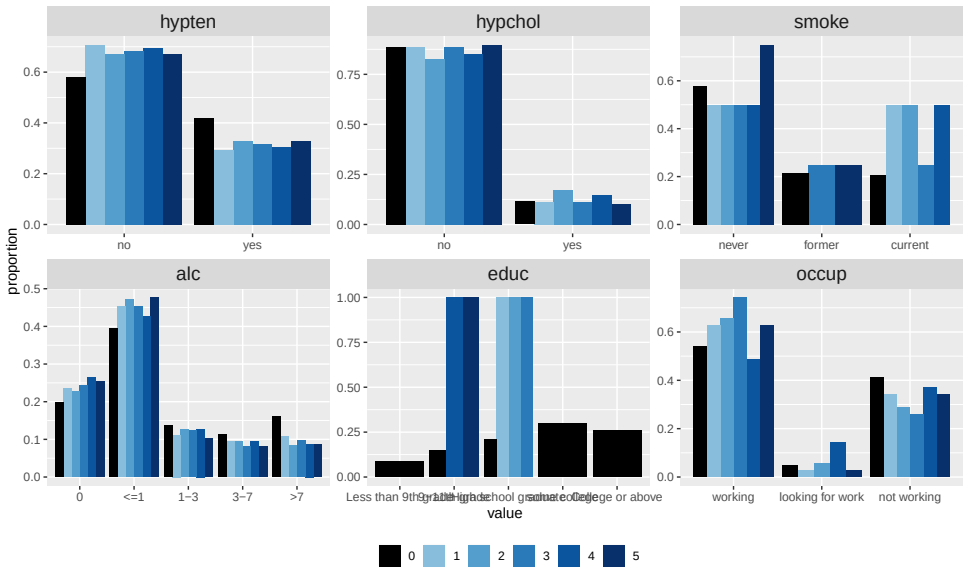
```
xyplot(imp2, hgt ~ chol|gender, pch = c(1,20))
```



All of the above graphs displayed only continuous imputed variables.
For categorical variables we can compare the proportion of values in each category.

mice does not provide a function to do this, but we can write one ourselves, as for instance the function `propplot()`, for which the syntax can be found here: https://gist.github.com/NErler/0d00375da460dd33839b98faeee2fdab

```
propplot(imp2, strip.text = element_text(size = 14))
```

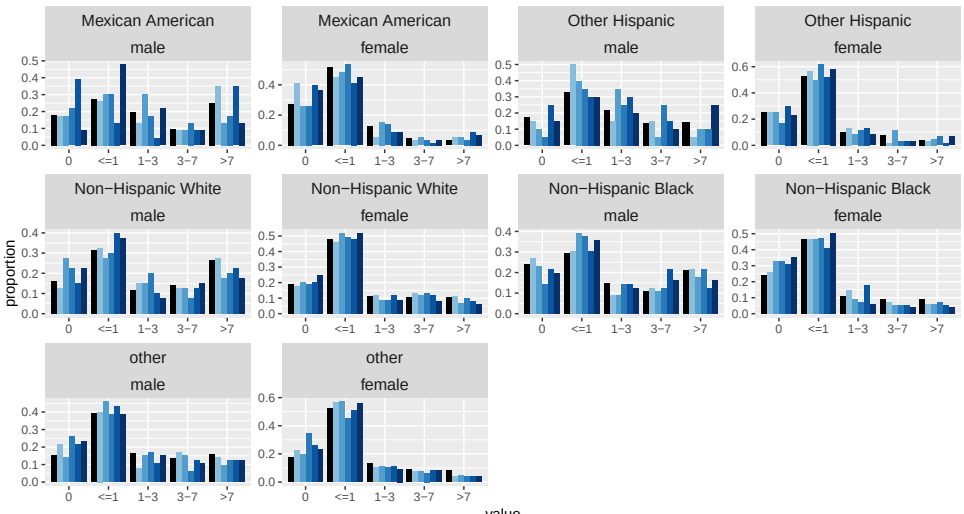


`smoke` and `educ` have very few missing values (4 and 1, respectively), so we do not need to worry about differences between observed and imputed data for those variables.

- `alc`: missing values are imputed in the lower consumption categories more often than we would expect from the observed data
- `hypten` is less frequent and
- `hypchol` a bit more frequent, in the imputed data compared to the observed.

If we expect that `gender` and `race` might explain the differences for `alc`, we can include those factors into the plot.

```
propplot(imp2, formula = alc ~ race + gender)
```



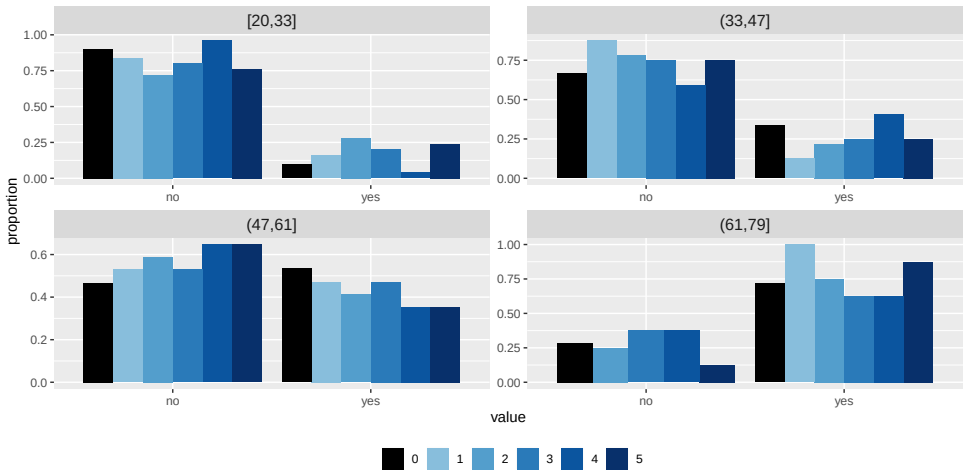
Since hypertension is more common in older individuals, we may want to investigate if age can explain the differences in imputed values of hypten.

```
round(sapply(split(NHANES[, "age"], addNA(NHANES$hypten)), summary), 1)

##           no  yes <NA>
## Min.    20.0 20.0 20.0
## 1st Qu. 28.0 47.0 30.0
## Median  38.0 59.0 38.5
## Mean    40.7 56.9 41.5
## 3rd Qu. 51.0 68.0 50.8
## Max.    79.0 79.0 78.0
```

The table shows that the distribution of age in participants with missing hypten is very similar to the distribution of age in participants without hypten.

Plotting the proportions of observed and imputed `hypten` separately per quartile of `age`:

```
propplot(imp2, formula = hypten ~ cut(age, quantile(age), include.lowest = T))
```



Practical

To practice the content of the previous section find the instructions for the practical here:

https://nerler.com/teaching/fgme2019/micheck

Once we have confirmed that our imputation was successful, we can move on to the **analysis of the imputed data**.

For example, we might be interested in the following logistic regression model:

```
glm(DM ~ age + gender + hypchol + BMI + smoke + alc,
    family = "binomial")
```

To fit the model on each of the imputed datasets, we do not need to extract the data from the `mids` object, but can use `with()`.

```
mod1 <- with(imp2, glm(DM ~ age + gender + hypchol + BMI + smoke + alc,
                       family = "binomial"))
```

`mod1` is an object of class `mira`.

Pooled results can be obtained using `pool()` and its summary.

```
res1 <- summary(pool(mod1), conf.int = TRUE)
round(res1, 3)

##               estimate std.error statistic       df p.value  2.5 % 97.5 %
## (Intercept)     -7.484     0.404   -18.520 2146.569   0.000 -8.277 -6.692
## age              0.056     0.004    12.698 1363.291   0.000  0.047  0.065
## genderfemale    -0.424     0.127    -3.349 2333.764   0.001 -0.673 -0.176
## hypcholyes       0.009     0.201     0.043   87.509   0.966 -0.392  0.409
## BMI              0.105     0.009    11.509 2440.963   0.000  0.087  0.123
## smoke.L          0.063     0.116     0.545 2401.105   0.586 -0.165  0.291
## smoke.Q         -0.075     0.115    -0.650 2409.227   0.515 -0.300  0.151
## alc.L           -0.562     0.165    -3.402  197.275   0.001 -0.887 -0.236
## alc.Q            0.182     0.189     0.963   50.461   0.340 -0.197  0.560
## alc.C            0.017     0.188     0.089   49.989   0.930 -0.361  0.395
## alc^4           -0.045     0.206    -0.217   45.455   0.829 -0.460  0.371
```

Pooling with `mice::pool()` is available for most types of models.

It extracts the model coefficients and variance-covariance matrices using `tidy()` from the package **broom**. Hence, pooling using the `pool()` function from **mice** only works for models of classes for which a method `tidy()` exists.

An alternative is offered by the package **mitools** and the function `MIcombine()`.

mice currently has two functions available for evaluating model fit / model comparison

For **linear** regression models the pooled R^2 can be calculated using `pool.r.squared()`.

```
mod2 <- with(imp2, lm(SBP ~ DM + age + hypten))
pool.r.squared(mod2, adjusted = TRUE)

##               est     lo 95     hi 95 fmi
## adj R^2 0.3252735 0.2943749 0.3562265 NaN
```

The argument `adjusted` specifies whether the adjusted R^2 or the standard R^2 is returned.

The function `pool.compare()` allows comparison of **nested models** (i.e., models where one is a special case of the other, with some parameters fixed to zero) using a **Wald test**.

Example: To test if `smoke` has a relevant contribution to the model for `DM` from above we re-fit the model without `smoke` and compare the two models:

```
mod3 <- with(imp2, glm(DM ~ age + gender + hypchol + BMI + alc,
                       family = "binomial"))
# Wald test
pool.compare(mod1, mod3)$pvalue

##           [,1]
## [1,] 0.6978098
```

`anova()` allows comparison of multiple nested models

The package **miceadds** extends **mice**, for example with the following functionality:

Combine χ^2 or F statistics from multiply imputed data:

```
miceadds::micombine.chisquare(dk, df, ...)
miceadds::micombine.F(values, df1, ...)
```

These functions take vectors of statistics computed on each imputed dataset and pool them.

Calculate correlation or covariance of imputed data:

```
miceadds::micombine.cor(mi.res, ...)
miceadds::micombine.cov(mi.res, ...)
```

These functions take `mids` objects as input.

The function `complete()` allows **extraction of the imputed data** from a `mids` object:

```
mice::complete(data, action = 1, include = FALSE, ...)
```

- `data`: the `mids` object
- `action`:
  - `1`, ..., `m` (single imputed dataset)
  - `"long"`: long format (imputed data stacked vertically)
  - `"broad"`: wide format (imputed data combined horizontally; ordered by imputation)
  - `"repeated"`: (like `"broad"`, but ordered by variable)
- `include`: include the original data?
  (if `action` is `"long"`, `"broad"` or `"repeated"`)

The function mids2spss() allows the **export of imputed data** (mids objects) to SPSS.

```
mids2spss(imp2,
          filedat = "datafile.txt", # the file containing the data
          filesps = "importsyntax.sps", # syntax to get .sav from .txt
          silent = TRUE, ...
)
```

Data from mids objects can also be exported to MPLUS using mids2mplus().

To **increase the number of imputed datasets** without re-doing the initial *m* imputations, a second set of imputations can be done and the two mids objects combined using ibind().

```
# same syntax as before, but different seed
imp2b <- update(imp2, post = post, maxit = 20, seed = 456)
imp2combi <- ibind(imp2, imp2b)
```

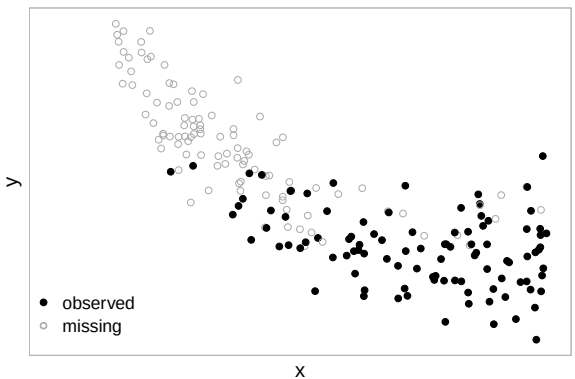
```
# check the new number of impute datasets:
imp2combi$m

## [1] 10
```

Part III
When MICE might fail

Consider the case where the **analysis model** (which we assume to be true) is

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots,$$

i.e., y has a **quadratic relationship** with x, and x is incomplete.

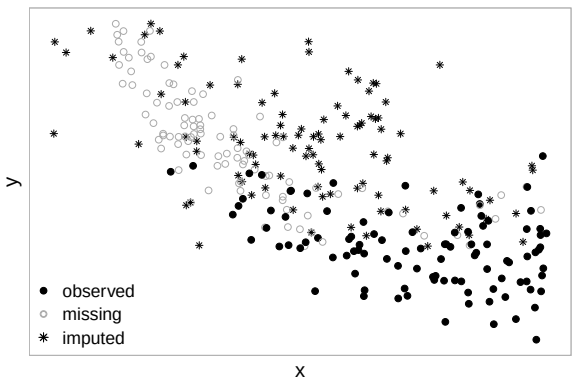


The original data show a curved pattern.

The model used to **impute** x when using MICE (naively) is

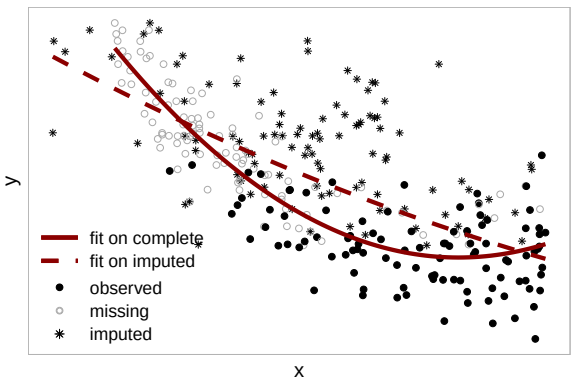
$$x = \theta_{10} + \theta_{11} y + \ldots,$$

i.e., a **linear relation** between x and y is assumed.



The imputed values **distort the curved pattern** of the original data.

The model fitted on the imputed data gives **severely biased results**; the non-linear shape of the curve has almost completely disappeared.

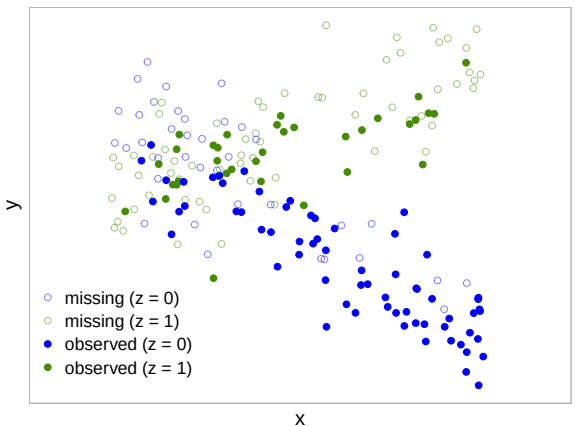


	β	95% CI
Original		
Intercept	-0.99	[-1.04, -0.95]
x	-0.61	[-0.66, -0.56]
x^2	0.52	[0.43, 0.62]
Imputed		
Intercept	-0.73	[-0.79, -0.66]
x	-0.53	[-0.62, -0.44]
x^2	0.07	[-0.07, 0.22]

Another example occurs when the analysis model (again, assumed to be true) is

$$y = \beta_0 + \beta_x x + \beta_z z + \boldsymbol{\beta_{xz} xz} + \ldots,$$

i.e., y has a **non-linear relationship** with x due to the **interaction term**.

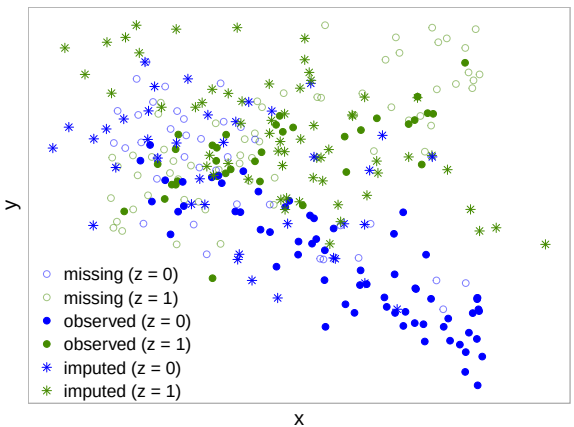


The original data shows a "<" shaped pattern.

The model used to impute x when using MICE (naively) is

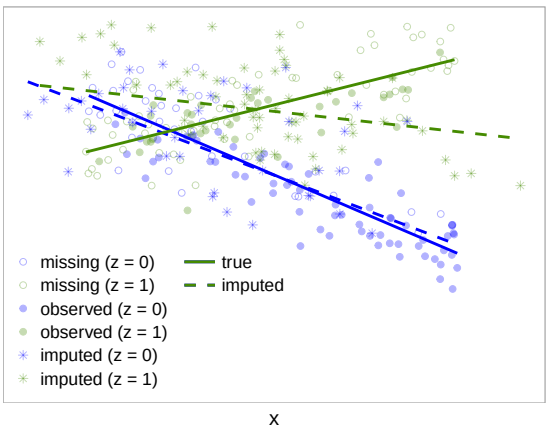
$$x = \theta_{10} + \theta_{11} y + \theta_{12} z + \ldots,$$

i.e., a linear relation between x and y is assumed.



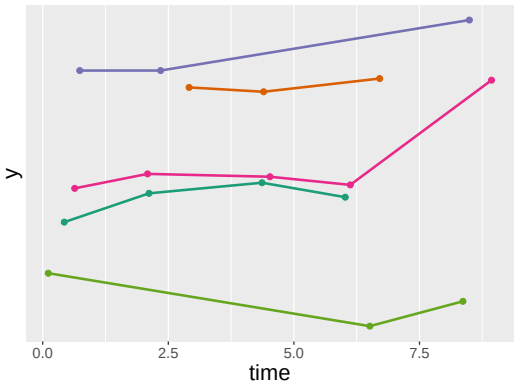
The "<" shaped pattern of the true data is **distorted by the imputed values**.

And the analysis on these naively imputed values leads to **severely biased estimates**.



	β	95% CI
Original		
Intercept	-0.96	[-1.00, -0.92]
x	-0.59	[-0.65, -0.53]
z	0.5	[0.45, 0.56]
x:z	0.94	[0.85, 1.03]
Imputed		
Intercept	-0.96	[-1.01, -0.91]
x	-0.52	[-0.61, -0.44]
z	0.46	[0.39, 0.54]
x:z	0.37	[0.24, 0.51]

Another setting where imputation with MICE is not straightforward is when the **outcome variable is longitudinal**.



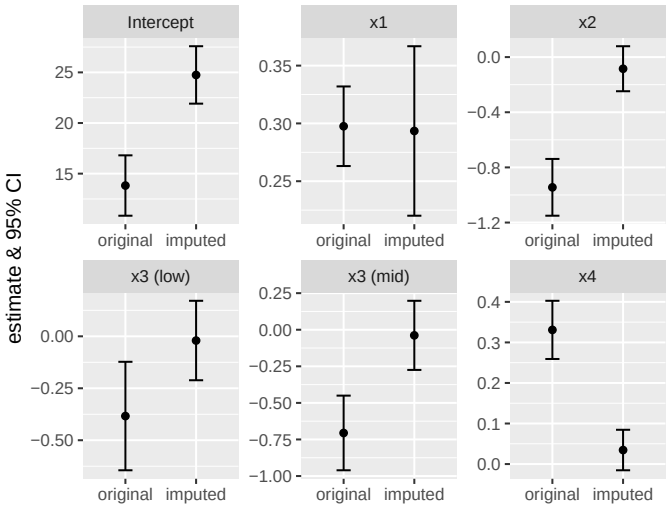
ID	y	x_1	x_2	x_3	x_4	time
5	✓	✓	✓		✓	0.43
5	✓	✓	✓		✓	2.12
5	✓	✓	✓		✓	4.37
5	✓	✓	✓		✓	6.02
6	✓	✓			✓	2.91
6	✓	✓			✓	4.40
6	✓	✓			✓	6.71
8	✓	✓	✓	✓		0.64
8	✓	✓	✓	✓		2.09
8	✓	✓	✓	✓		4.52
8	✓	✓	✓	✓		6.12
8	✓	✓	✓	✓		8.93
18	✓	✓		✓	✓	0.11
18	✓	✓		✓	✓	6.51
18	✓	✓		✓	✓	8.37
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Here, $x_1, \ldots, x_4$ are baseline covariates, i.e., not measured repeatedly (e.g. age at baseline, gender, education level, ...

If we use MICE in the data in this (long) format, each row would be regarded as independent, which may cause bias and **inconsistent imputations**.

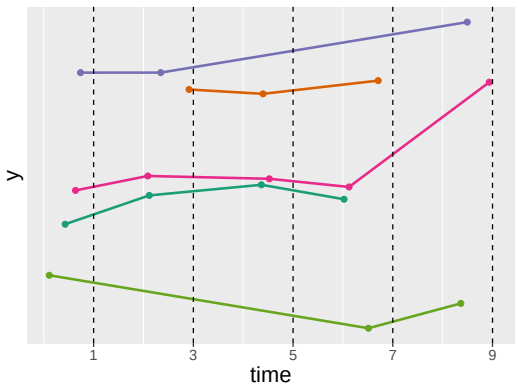
Imputed values of baseline covariates are imputed with different values, creating data that could not have been observed.

ID	y	x_1	x_2	x_3	x_4	time
5	✓	✓	✓	low	✓	0.43
5	✓	✓	✓	mid	✓	2.12
5	✓	✓	✓	high	✓	4.37
5	✓	✓	✓	low	✓	6.02
6	✓	✓	boy	low	✓	2.91
6	✓	✓	boy	high	✓	4.40
6	✓	✓	boy	low	✓	6.71
8	✓	✓	✓	✓	39.75	0.64
8	✓	✓	✓	✓	39.64	2.09
8	✓	✓	✓	✓	41.41	4.52
8	✓	✓	✓	✓	39.58	6.12
8	✓	✓	✓	✓	40.37	8.93
18	✓	✓	boy	✓	✓	0.11
18	✓	✓	girl	✓	✓	6.51
18	✓	✓	girl	✓	✓	8.37
⋮	⋮	⋮	⋮	⋮	⋮	⋮



Estimates can be severely biased.

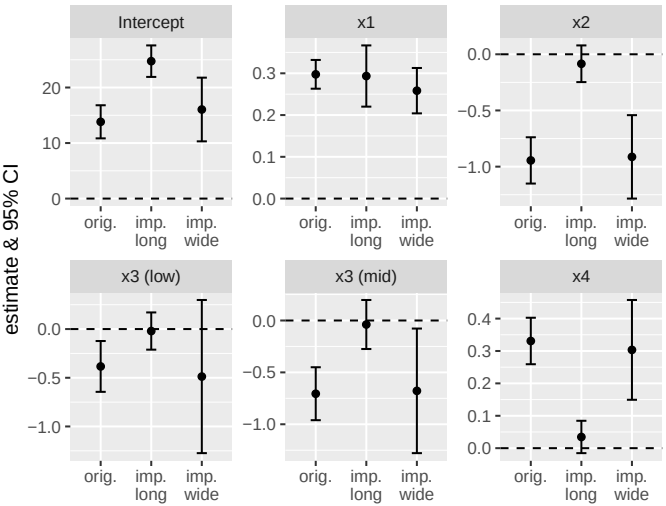
In some settings **imputation in wide format** may be possible.



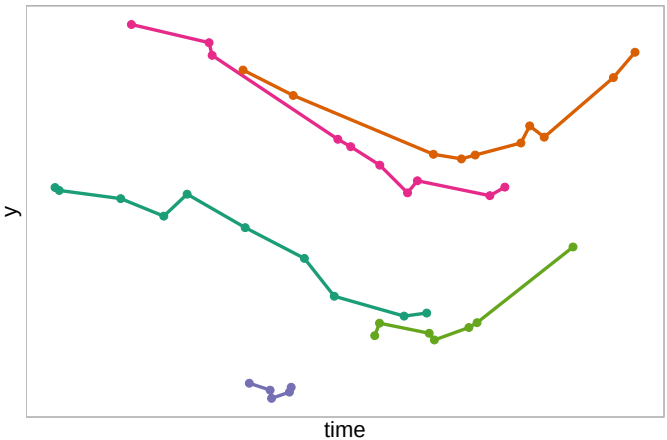
ID	y	x_1	x_2	x_3	x_4	time
5	✓	✓	✓		✓	0.43
5	✓	✓	✓		✓	2.12
5	✓	✓	✓		✓	4.37
5	✓	✓	✓		✓	6.02
6	✓	✓			✓	2.91
6	✓	✓			✓	4.40
6	✓	✓			✓	6.71
8	✓	✓	✓	✓		0.64
8	✓	✓	✓	✓		2.09
8	✓	✓	✓	✓		4.52
8	✓	✓	✓	✓		6.12
8	✓	✓	✓	✓		8.93
18	✓	✓		✓	✓	0.11
18	✓	✓		✓	✓	6.51
18	✓	✓		✓	✓	8.37
⋮	⋮	⋮	⋮	⋮	⋮	⋮

id	y.1	y.3	y.5	y.7	y.9	time.1	time.3	time.5	time.7	time.9	...
5	33.57	33.87	33.98	33.83		0.43	2.12	4.37	6.02		...
6		34.96	34.92	35.05			2.91	4.4	6.71		...
8	33.92	34.07	34.04	33.96	35.04	0.64	2.09	4.52	6.12	8.93	...
18	33.05			32.5	32.76	0.11			6.51	8.37	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

In this **wide format data** frame, missing values in the outcome and measurement times need to be imputed (to be able to use them as predictors to impute covariates), even though we would not need to impute them for the analysis (mixed model is valid when outcome measurements are M(C)AR).

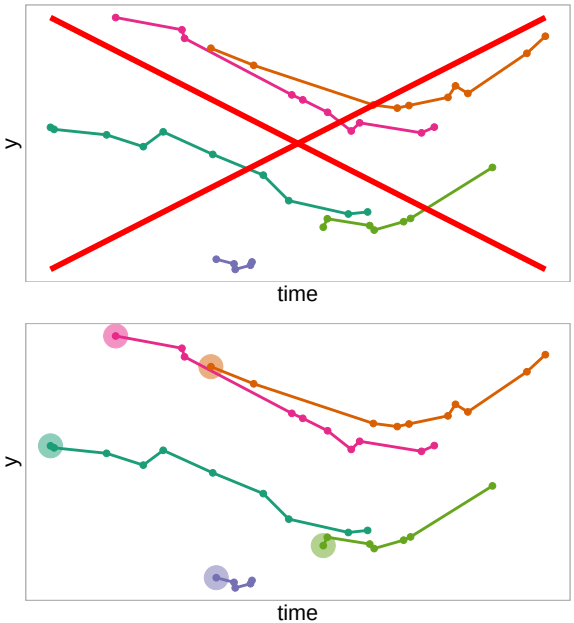


Better, but large confidence intervals.



When the data is very **unbalanced**, transformation to wide format is not possible.

(Or at least transformation to wide format leads to variables with high proportions of missing values.)



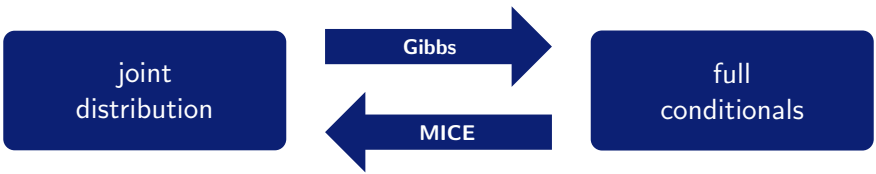
Naive approaches that are sometimes used are to

- **ignore the outcome** in the imputation, or to
- use only the **first/baseline outcome**

However, **important information may be lost**, resulting in invalid imputations and biased results.

The MICE algorithm is based on the idea of Gibbs sampling.

Gibbs sampling exploits the fact that a joint distribution is fully determined by its full conditional distributions.



In MICE, the full conditionals are not derived from the joint distribution: we directly specify the full conditionals and hope a joint distribution exists.

Two important definitions:

Compatibility:

A joint distribution exists, that has the full conditionals (imputation models) as its conditional distributions.

Congeniality:

The imputation model is compatible with the analysis model.

Important requirements for MICE to work well include:

- Compatibility
- Congeniality
- MAR or MCAR (in the standard implementations)
- **All relevant variables** need to be included. (Omission might result in MNAR.)
- **The outcome needs to be included** as predictor variable (but we usually do not impute missing outcome values).
- The imputation models (and analysis model) need to be **correctly specified** (which is a requirement in any standard analysis).

To **avoid incompatible** and **uncongenial** imputation models, we need to

- specify the joint distribution
- and derive full conditionals / imputation models from this joint distribution

instead of specifying them directly.

Problem:
The joint distribution may not be of any known form:

$$\begin{array}{l} x_1 \sim N(\mu_1, \sigma_1^2) \\ x_2 \sim N(\mu_2, \sigma_2^2) \end{array} \quad \Rightarrow \quad \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \sim N\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix} \right)$$

but $$\begin{array}{l} x_1 \sim N(\mu_1, \sigma_1^2) \\ x_2 \sim Bin(\mu_2) \end{array} \quad \Rightarrow \quad \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \sim ???$$

Possible approaches:

Approach 1: **Multivariate Normal Model**
Approximate the joint distribution by a known multivariate distribution (usually the normal distribution; this is the joint model MI mentioned before).

Approach 2: **Sequential Factorization**
Factorize the joint distribution into a (sequence of) conditional and a marginal distributions.

The **joint distribution** of two variables y and x can be written as the product of conditional distributions:

$$p(y, x) = p(y \mid x)\, p(x)$$

(or alternatively $p(y, x) = p(x \mid y)\, p(y)$)

This can easily be **extended for more variables**:

$$p(y, x_1, \ldots, x_p, X_c) = \underbrace{p(y \mid x_1, \ldots, x_p, X_c)}_{\text{analysis model}}\, p(x_1 \mid x_2, \ldots, x_p, X_c)\ \ldots\ p(x_p \mid X_c)$$

where $x_1, \ldots, x_p$ denote incomplete covariates and X_c contains all completely observed covariates.

The analysis model is part of the specification of the joint distribution.

➡ Advantages:

- The outcome is **automatically included in the imputation** procedure.
- The outcome does not appear in any of the predictors of the imputation models:
  - **no need to approximate** complex outcomes,
  - **no need to summarize** complex outcomes.
- The parameters of interest are obtained directly
  ➡ imputation and analysis in one step
- **Non-linear associations** or interactions involving incomplete covariates are specified in the analysis model and thereby **automatically taken into account**

Since the joint distribution usually does not have a known form, Gibbs sampling is used to estimate parameters and sample imputed values.

Joint Analysis and Imputation,
uses the **sequential factorization approach** to perform simultaneous analysis and imputation in the Bayesian framework [4, 5, 3].

JointAI (version 0.6.0) can analyse incomplete data using

- linear regression
- generalized linear regression
- linear mixed models
- generalized linear mixed models
- (ordinal) cumulative logit regression
- (ordinal) cumulative logit mixed models
- parametric (Weibull) survival models
- Cox proportional hazards models

while assuring compatibility between analysis model and imputation models when non-linear functions or interactions are included.

The necessary **Gibbs sampling** is performed using **JAGS** (an external program), which is free, but needs to be installed from https://sourceforge.net/projects/mcmc-jags/files/.

JointAI can be installed from CRAN or GitHub:

```
install.packages("devtools")
devtools::install_github("NErler/JointAI")
```

JointAI has its own web page (https://nerler.github.io/JointAI/) with several vignettes on Visualization of Incomplete Data, a Minimal Example, details on Model Specification, etc.

There is no strategy for MICE that can guarantee valid imputations when non-linear functional forms and/or interactions are involved, but some settings in **mice** may help to reduce bias in the resulting estimates.

For imputation of variables that have non-linear associations

- PMM often works better than imputation with a normal model,
- the **J**ust **A**nother **V**ariable approach can reduce bias in interactions,
- `quadratic` can help to impute variables with quadratic association.

Just Another Variable (JAV) approach:

- pre-calculate the non-linear form (or interaction term) in the incomplete data,
- add it as a column to the dataset, and
- impute it as if it was just another variable.

`quadratic` uses the "polynomial combination" method to impute covariates that have a quadratic association with the outcome [15, pp. 139–141], [16].

This is to ensure the imputed values for x and x^2 are consistent, and to reduce bias in the subsequent analysis that uses x and x^2.

In my experience, using `quadratic` can lead to numerical problems.

To demonstrate the approaches, we use a simulated example dataset `DFnonlin`, with

- continuous outcome y
- continuous (normal) covariate x (50% missing values MCAR)
- quadratic effect of x on y
- binary covariate z (complete)
- interaction between x and z

In the naive approach, we leave all settings to the defaults.

```
# naive imputation, using only y, x, z
impnaive <- mice(DF_nonlin, printFlag = F)
```

We use two different JAV approaches:

JAV: calculating the **quadratic and interaction term** before imputation

```
# add quadratic term and interaction to data
DF2 <- DF_nonlin
DF2$xx <- DF2$x^2
DF2$xz <- DF2$x * DF2$z

# JAV imputation
impJAV <- mice(DF2, printFlag = F, maxit = 20)
```

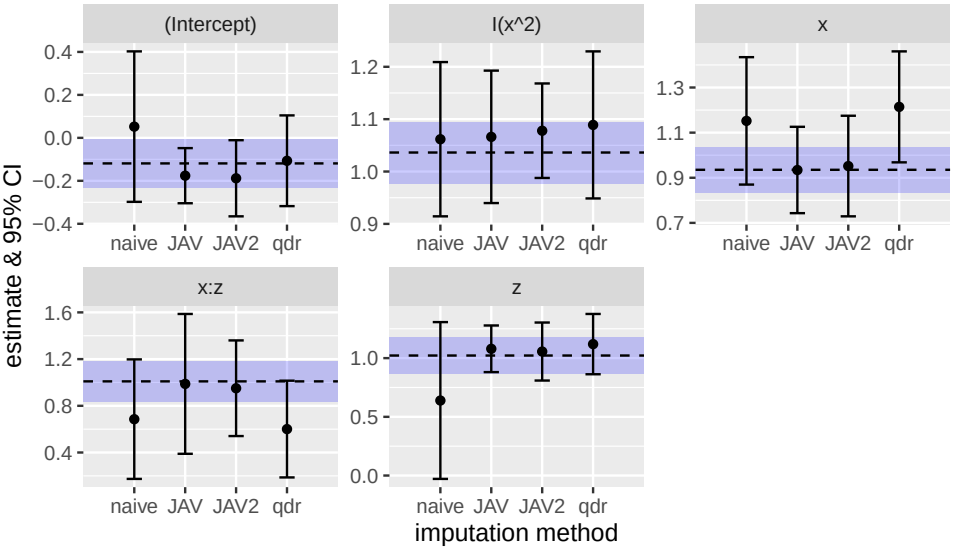
JAV2: additionally using an interaction between z and y

```
# add interaction between y and z to data
DF3 <- DF2
DF3$yz <- DF3$y * DF3$z

# JAV imputation with additional interaction
impJAV2 <- mice(DF3, printFlag = F, maxit = 20)
```

We also try using imputation method `quadratic`.

```
# adapt the imputation method for quadratic imputation
methqdr <- impJAV$meth
methqdr[c("x", "xx", "xz")] <- c("quadratic", "~I(x^2)", "~I(x*z)")

# adapt the predictor matrix
predqdr <- impJAV$pred
predqdr[, "xx"] <- 0

impqdr <- mice(DF2, meth = methqdr, pred = predqdr,
               printFlag = F, maxit = 10)
```

Note: there were warning messages about numerical issues for this approach (`glm.fit: fitted probabilities numerically 0 or 1 occurred`).



For this example, **none of the approaches provided satisfying results**.

The syntax we use to analyse and impute the current example using **JointAI** is similar to the specification of a standard linear model using `lm()`.

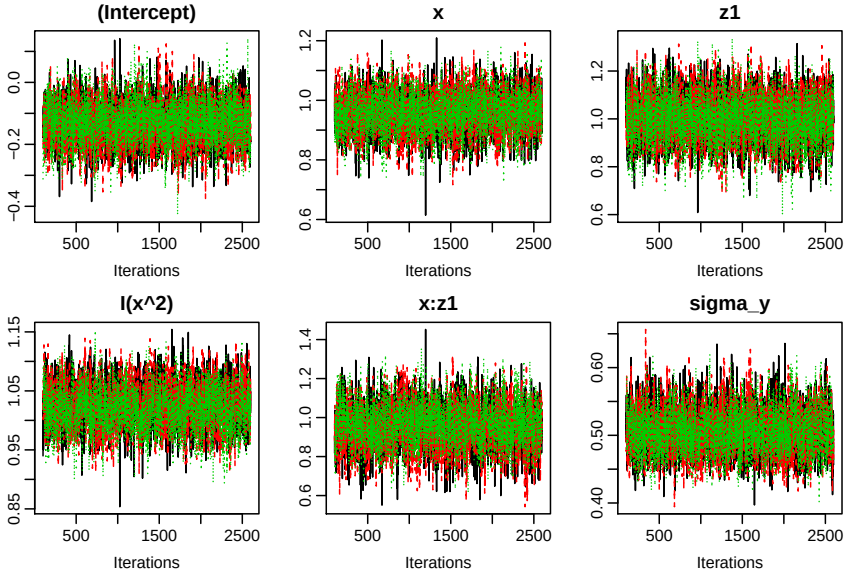
```
library(JointAI)
JointAI_nonlin <- lm_imp(y ~ x*z + I(x^2), data = DF_nonlin,
                         n.iter = 2500)
```

Convergence of the Gibbs sampler can be checked using a traceplot.

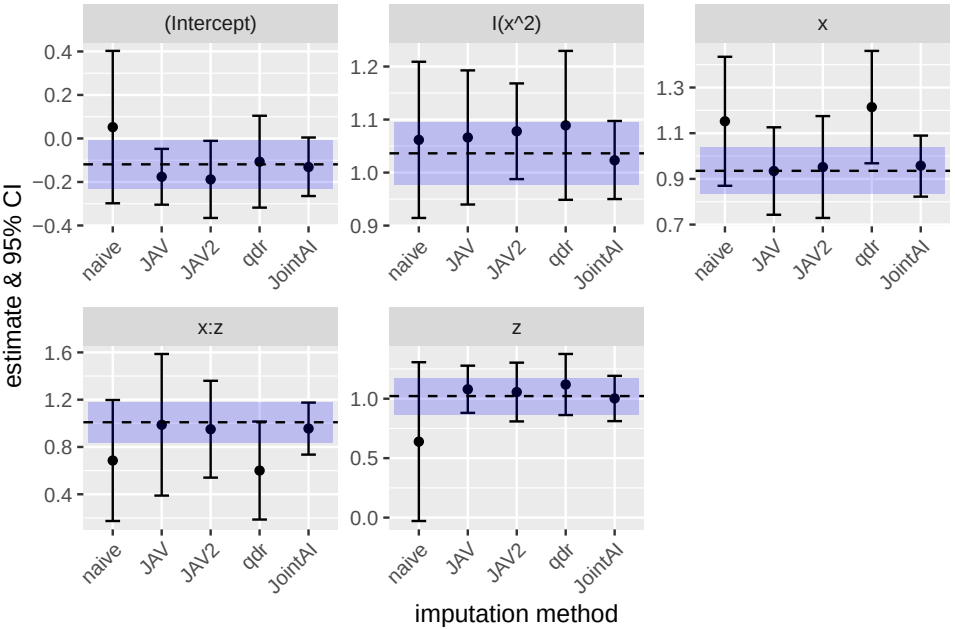
```
traceplot(JointAI_nonlin, ncol = 3)
```

Results (no separate analysis & pooling is necessary) can be obtained with the `summary()` function:

```
summary(JointAI_nonlin)
```


(Intercept)
x
z1
I(x^2)
x:z1
sigma_y
Iterations
500
1500
2500
−0.4
−0.2
0.0
0.6
0.8
1.0
1.2
0.85
0.95
1.05
1.15
0.40
0.50
0.60
1.4

```
##
##  Linear model fitted with JointAI
##
## Call:
## lm_imp(formula = y ~ x * z + I(x^2), data = DF_nonlin, n.iter = 2500)
##
## Posterior summary:
##               Mean     SD   2.5%   97.5% tail-prob. GR-crit
## (Intercept) -0.131 0.0691 -0.265 0.00445     0.0589    1.02
## x            0.958 0.0677  0.822 1.08954     0.0000    1.00
## z1           1.003 0.0977  0.812 1.19210     0.0000    1.02
## I(x^2)       1.023 0.0378  0.950 1.09735     0.0000    1.05
## x:z1         0.956 0.1132  0.735 1.17537     0.0000    1.04
##
## Posterior summary of residual std. deviation:
##          Mean     SD  2.5% 97.5% GR-crit
## sigma_y 0.506 0.0337 0.445 0.577       1
##
##
## MCMC settings:
## Iterations = 101:2600
## Sample size per chain = 2500
## Thinning interval = 1
## Number of chains = 3
##
## Number of observations: 200
```


(Intercept)
I(x^2)
x
x:z
z
estimate & 95% CI
imputation method
naive
JAV
JAV2
qdr
JointAI

Practical

To practice imputation with non-linear forms or interaction terms find the instructions for the practical here:

https://nerler.com/teaching/fgme2019/minonlin

mice has functions to allow imputation of longitudinal (2-level) data:

- **Level 1:**
  repeated measurements within subjects or subjects within classes
- **Level 2:**
  time-constant/baseline covariates, between subjects effects, variables on the group level

Imputation methods for **level-1** variables:

- `2l.pan`
- `2l.norm`
- `2l.lmer`
- `2l.bin`

Imputation methods for **level-2** variables:

- `2lonly.norm`
- `2lonly.pmm`
- `2lonly.mean`

`2l.pan` uses a linear two-level model with **homogeneous within group variances** using Gibbs sampling [13]. It needs the package **pan** to be installed.

`2l.pan` allows for different roles of predictor variables, that can be specified as different values in the `predictorMatrix`:

- grouping/ID variable: -2
- random effects (also included as fixed effects): 2
- fixed effects of group means: 3
- fixed effects of group means & random effects: 4

```
# random effects of x in model for y
pred["y","x"] <- 2
# fixed effects of x and group mean of x
pred["y","x"] <- 3
# random effects of x and group mean of x
pred["y","x"] <- 4
```

`2l.norm` implements a (Bayesian) linear two-level model with **heterogenous** group variances.
In the current implementation all predictors should be specified as random effects (set to 2 in the `predictorMatrix`, because the algorithm does not handle predictors that are specified as fixed effects).

`2l.lmer`/`2l.bin` imputes univariate systematically and sporadically missing data using a two-level normal/logistic model using `lmer()`/`glmer()` from package **lme4**.

`2lonly.norm` and `2lonly.pmm` can be used to impute level-2 variables (in combination with `2l.pan` for level-1 variables).

In all cases, the group identifier ("id" variable) needs to be set to -2 in the `predictorMatrix`.

`2lonly.mean` imputes values with the mean of the observed values per class. This method should only be used to fill in values that are known to be constant per class and have some values observed in each class.

Example: In a multi-center trial the type of some medical equipment is known to be the same for all patients treated in the same hospital, but not filled in for some patients.

As an example, we will impute the second (unbalanced) longitudinal data example from above. The data contain

- $x1$ (complete)
- $x2$ (binary, 30% missing values)
- $x3$ (3 categories, 30% missing values)
- $x4$ (continuous/normal, 30% missing values)
- y (longitudinal outcome)
- *time* (time variable with quadratic effect)
- *id* (id variable)

Since there is no 2-level method for categorical data, we use `2lonly.pmm` to impute $x2$ and $x3$.

As usual, we start with the setup run of `mice()`

```
imp0 <- mice(DFexlong2, maxit = 0)
meth <- imp0$method
pred <- imp0$predictorMatrix
```

and adjust the imputation `method` and `predictorMatrix`

```
meth[c("x2", "x3")] <- "2lonly.pmm"
meth[c("x4")] <- "2lonly.norm"

pred[, "id"] <- -2  # identify id variable
pred[, "ti"] <- 0 # don't use time-point indicator
```

We can then perform the imputation.

```
imp <- mice(DFexlong2, maxit = 10, method = meth,
            predictorMatrix = pred, printFlag = FALSE)
```

The imputed data can be analysed using either `lmer()` from the package **lme4**, or `lme()` from **nlme**. Here we use the former.

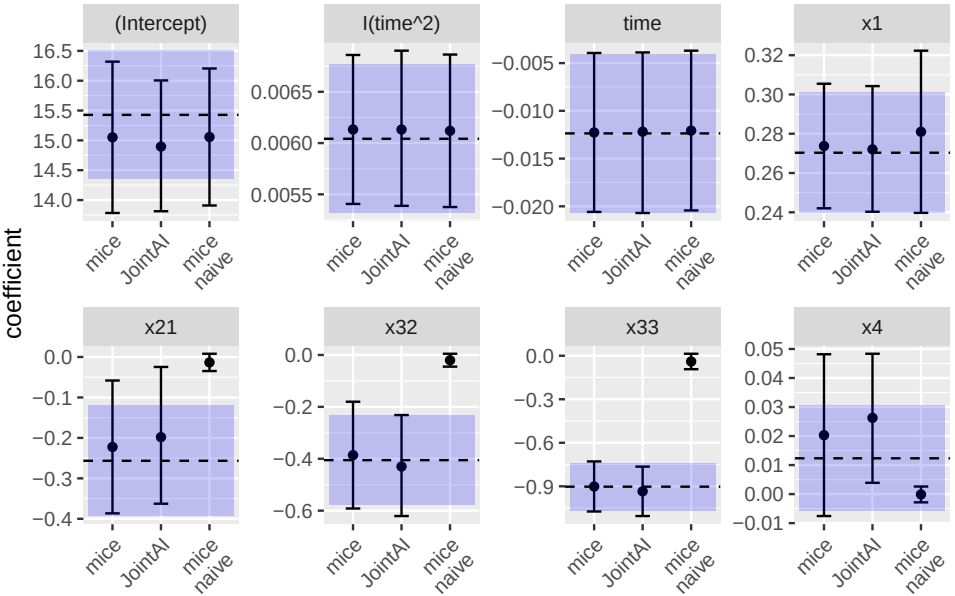
```
library(lme4)
models <- with(imp, lmer(y ~ x1 + x2 + x3 + x4 + time + I(time^2) +
                           (time|id),
                         control = lmerControl(optimizer = "Nelder_Mead")
))
mice_longimp <- summary(pool(models), conf.int = TRUE)
```

Linear mixed models with incomplete covariates can also be analysed using the package **JointAI**.

The syntax is analogous the syntax used in `lme()` of the package **nlme**.

```
library(JointAI)
JointAI_long <- lme_imp(y ~ x1 + x2 + x3 + x4 + time + I(time^2),
                        random = ~time|id, data = DFexlong2,
                        n.iter = 5000)
```

Again, convergence of the Gibbs sampler should be checked, e.g., using `traceplot()` before obtaining the results.
Contrary to the two-level imputation of **mice**, non-linear associations are appropriately handled.


coefficient
(Intercept)
I(time^2)
time
x1
x21
x32
x33
x4
mice
JointAI
mice naive

Practical

To practice imputation with longitudinal data find the instructions for the practical here:

https://nerler.com/teaching/fgme2019/milong

Part IV
Multiple Imputation Strategies

In complex settings, variables may need to be **re-calculated** or **re-coded** after imputation:

- Use `complete()` to convert the imputed data from a `mids` object to a `data.frame`.
- Perform the necessary calculations.
- Convert the changed `data.frame` back to a `mids` object using the functions such as `as.mids()`, `miceadds::datalist2mids()`, `mitools::imputationList()`, ...

Not just in imputation: Set a **seed value** to create reproducible results.

- in R: `set.seed()`
- in `mice()`: argument `seed`

Early publications on multiple imputation suggested that 3 – 5 imputations are sufficient and this is still a common assumption in practice.[11]

The reasoning behind using a small number of imputed datasets was that **storage of imputed data was "expensive"** (which is no longer the case) and a larger number of imputations would only have little advantage.[12]

More **recent work** from various authors [17, 15, 6] considers the efficiency of the pooled estimates, reproducibility of the results, statistical power of tests or the width of the resulting confidence intervals compared to the width of the true confidence intervals.

A **suggested rule of thumb** is that the **number of imputed datasets** should be similar to the **percentage of incomplete cases**.[17] Since this percentage depends on the size of the dataset, the **average percentage of missing values** per variable could be used as an alternative.[15]

Generally, using **more imputed datasets should be preferred**, especially in settings where the computational burden allows for it. Even though results are unlikely to change with a larger number of imputations, it can increase the efficiency and reproducibility of the results.

In imputation, generally the **advice is to include as much information as possible** in the imputation models.

Using a large number of predictor variables

- makes the **MAR assumption more plausible** (and, hence, reduces bias due to MNAR missingness)
- can **reduce uncertainty** about the missing values

This can **work well in small or medium sized datasets** (20 – 30 separate variables, i.e. without interactions, variables derived from others, . . .)

However, **in large datasets** (contain hundreds or thousands of variables) this is **not feasible**.[15]

For large datasets a possible strategy is to

- Include all **variables used in the analysis model(s)** (including the outcome!).
- Include auxiliary variables if they are **strong predictors of missingness**.
- Include auxiliary variables if they have **strong associations with the incomplete variables**.
- Use **auxiliary variables only if they do not have too many missing values** themselves (and are observed for most of the incomplete cases of the variable of interest).
- Use **auxiliary variables** only in those imputation models for which they are **relevant** (and exclude them for others using the predictor matrix).
- Calculate **summary scores** from multiple items referring to the same concept and use the summary score as predictor variable.

There is **no clear cut-off** for the proportion of missing values that can be handled adequately using MICE (or any other imputation method).

The amount of missingness that can be handeled **depends on the information that is available** to impute it.

- Are there **strong predictor variabels** available & observed?
- Are there **sufficient observed cases** to get reliable estimates for the predictive distribution?

Example:

- In a set of $N = 50$ cases, 50% missing values leaves 25 cases to estimate the parameters of the predictive distribution.
- In a large set of $N = 5000$ subjects, 50% missing cases leaves 2500 observed cases to estimate parameters.

Usually, **missing outcome values are not imputed**.

Why?

When there are no auxiliary variables, imputation and analysis model are equal.

- Parameters of the imputation model are estimated on observed cases of the outcome.
- Imputed values will fit the assumed model perfectly.
- Including imputed cases in the analysis does not add any information.

Exception:

- When very **strong auxiliary variables** are available.
- Outcomes may be imputed when one imputation is performed for **several analysis models**, because not imputing the outcome(s) would mean
  - excluding cases with missing outcome(s) from the imputation, or
  - excluding the outcome variable(s) as predictor(s).

Multiple imputation is **not a quick and easy solution for missing data**. It requires **care and knowledge** about

- the **data** to be imputed (and the context of the data),
- the statistical **method** used for imputation, and
- the **software** implementation used.

Moreover

- **Never accept default settings of software blindly.**
- **Question the plausibility of the MAR assumption.** If it is doubtful, use sensitivity analysis.

Remember:

- **Use as much information as possible**
  - include all covariates and the outcome
  - use auxiliary information
  - use the most detailed version of variables if possible
- **Avoid feedback** from derived variables to their originals.
- Think carefully how to handle variables that are derived from other variables.
- Consider the impact the **visit sequence** may have.
- **Imputation models must fit the data**
  (correct assumption of error distribution and functional forms and possible interactions of predictor variables).
- Choose an appropriate **number of imputations**.
- Make sure the imputation algorithm has **converged**.
- Use **common sense** when evaluating if the imputed values are plausible.

Currently, there are **289 packages** available on CRAN that use the word **"missing"** in either the title or description of the package, **163** that use either **"impute" or "imputation"** and **65** that use the word **"incomplete"**.

➡ **The mice package is often a good option, but certainly not the only option to perform imputation!**

CRAN Task View on Missing Data:
https://cran.r-project.org/web/views/MissingData.html

- overview on the available R packages for missing data
- good starting point when searching for a package with a particular functionality

We have focussed on a few imputation methods that cover the most common types of data but there are many more methods implemented.

Imputation methods implemented in the **mice** package:

mice.impute.2l.bin	mice.impute.lda	mice.impute.panImpute
mice.impute.2l.lmer	mice.impute.logreg	mice.impute.passive
mice.impute.2l.norm	mice.impute.logreg.boot	mice.impute.pmm
mice.impute.2l.pan	mice.impute.mean	mice.impute.polr
mice.impute.2lonly.mean	mice.impute.midastouch	mice.impute.polyreg
mice.impute.2lonly.norm	mice.impute.norm	mice.impute.quadratic
mice.impute.2lonly.pmm	mice.impute.norm.boot	mice.impute.rf
mice.impute.cart	mice.impute.norm.nob	mice.impute.ri
mice.impute.jomoImpute	mice.impute.norm.predict	mice.impute.sample

Note: Just because a method is implemented does not mean you need to / should use it.

Imputation methods implemented in the **miceadds** package:

mice.impute.2l.binary	mice.impute.hotDeck
mice.impute.2l.contextual.norm	mice.impute.lm
mice.impute.2l.contextual.pmm	mice.impute.lm_fun
mice.impute.2l.continuous	mice.impute.lqs
mice.impute.2l.groupmean	mice.impute.ml.lmer
mice.impute.2l.groupmean.elim	mice.impute.plausible.values
mice.impute.2l.latentgroupmean.mcmc	mice.impute.pls
mice.impute.2l.latentgroupmean.ml	mice.impute.pmm3
mice.impute.2l.plausible.values	mice.impute.pmm4
mice.impute.2l.pls	mice.impute.pmm5
mice.impute.2l.pls2	mice.impute.pmm6
mice.impute.2l.pmm	mice.impute.rlm
mice.impute.2lonly.function	mice.impute.smcfcs
mice.impute.2lonly.norm2	mice.impute.tricube.pmm
mice.impute.2lonly.pmm2	mice.impute.tricube.pmm2
mice.impute.bygroup	mice.impute.weighted.norm
mice.impute.grouped	mice.impute.weighted.pmm

Imputation methods implemented in the **micemd** package:

mice.impute.2l.2stage.bin	mice.impute.2l.glm.bin
mice.impute.2l.2stage.norm	mice.impute.2l.glm.norm
mice.impute.2l.2stage.pmm	mice.impute.2l.glm.pois
mice.impute.2l.2stage.pois	mice.impute.2l.jomo

Besides **JointAI**, there are more alternatives for imputation in complex settings:

- **smcfcs:** substantive model compatible fully conditional specification (in GLMs & survival models)
- **jomo:** Joint model MI (GLMs, GLMMs, Cox, ordinal mixed model)
- **mdmb:** model based missing data models (linear, logistic, multi-level)

References

References

[1] John Barnard and Donald B Rubin.
Miscellanea. small-sample degrees of freedom with multiple imputation.
Biometrika, 86(4):948–955, 1999.

[2] James Carpenter and Michael Kenward.
Multiple imputation and its application.
John Wiley & Sons, 2012.

[3] Nicole S Erler, Dimitris Rizopoulos, Vincent WV Jaddoe, Oscar H Franco, and Emmanuel MEH Lesaffre.
Bayesian imputation of time-varying covariates in linear mixed models.
Statistical Methods in Medical Research, 28(2):555 – 568, 2019.

[4] Nicole S. Erler, Dimitris Rizopoulos, and Emmanuel M.E.H. Lesaffre.
JointAI: Joint analysis and imputation of incomplete data in r.
arXiv e-prints, page arXiv:1907.10867, Jul 2019.

References (cont.)

[5] Nicole S Erler, Dimitris Rizopoulos, Joost van Rosmalen, Vincent WV Jaddoe, Oscar H Franco, and Emmanuel MEH Lesaffre.

Dealing with missing covariates in epidemiologic studies: a comparison between multiple imputation and a full Bayesian approach.

Statistics in Medicine, 35(17):2955–2974, 2016.

[6] John W Graham, Allison E Olchowski, and Tamika D Gilreath.

How many imputations are really needed? some practical clarifications of multiple imputation theory.

Prevention science, 8(3):206–213, 2007.

[7] Roderick JA Little.

Missing-data adjustments in large surveys.

Journal of Business & Economic Statistics, 6(3):287–296, 1988.

References (cont.)

[8] Donald B Rubin.
Statistical matching using file concatenation with adjusted weights and multiple imputations.
Journal of Business & Economic Statistics, 4(1):87–94, 1986.

[9] Donald B. Rubin.
Multiple Imputation for Nonresponse in Surveys.
Wiley Series in Probability and Statistics. Wiley, 1987.

[10] Donald B Rubin.
Multiple imputation after 18+ years.
Journal of the American statistical Association, 91(434):473–489, 1996.

[11] Donald B Rubin.
The design of a general and flexible system for handling nonresponse in sample surveys.
The American Statistician, 58(4):298–302, 2004.

References (cont.)

[12] Joseph L Schafer.
Analysis of incomplete multivariate data.
CRC press, 1997.

[13] Joseph L Schafer and Recai M Yucel.
Computational strategies for multivariate linear mixed-effects models with missing values.
Journal of computational and Graphical Statistics, 11(2):437–457, 2002.

[14] Juned Siddique and Thomas R Belin.
Multiple imputation using an iterative hot-deck with distance-based donor selection.
Statistics in medicine, 27(1):83–102, 2008.

[15] Stef van Buuren.
Flexible Imputation of Missing Data.
Chapman & Hall/CRC Interdisciplinary Statistics. Taylor & Francis, 2012.

References (cont.)

[16] Gerko Vink and Stef van Buuren.
Multiple imputation of squared terms.
Sociological Methods & Research, 42(4):598–607, 2013.

[17] Ian R White, Patrick Royston, and Angela M Wood.
Multiple imputation using chained equations: issues and guidance for practice.
Statistics in medicine, 30(4):377–399, 2011.

Erasmus MC
University Medical Center Rotterdam

n.erler@erasmusmc.nl
N_Erler
NErler
www.nerler.com