



Missing Values in Clinical Research (EP16)

Multiple Imputation

Nicole Erler

Department of Biostatistics, Erasmus MC
n.erler@erasmusmc.nl

14 - 18 May, 2018

Erasmus MC
University Medical Center Rotterdam



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.

Practical: Imputation with **mice**

Outline (cont.)

Part III: When MICE might fail

Introduction to

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

Practical: Imputation in complex settings

Part IV: Multiple Imputation Strategies

Some tips & tricks

Part I

Multiple Imputation

Outline of Part I

1. What is Multiple Imputation?

- 1.1 History & Ideas
- 1.2 Three steps

2. Imputation step

- 2.1 Univariate missing data
- 2.2 Multivariate missing data
- 2.3 FCS/MICE
- 2.4 Checking convergence

3. Analysis step

4. Pooling

- 4.1 Why pooling?
- 4.2 Rubin's Rules

5. A closer look at the imputation step

- 5.1 Bayesian multiple imputation
- 5.2 Bootstrap multiple imputation
- 5.3 Semi-parametric imputation
- 5.4 What is implemented in software?

1. What is Multiple Imputation?

1.1. History & Ideas

- 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**.

1. What is Multiple Imputation?

1.1. History & Ideas

- 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**.

Such data should be usable by [11]

- 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**.

1. What is Multiple Imputation?

1.1. History & Ideas

Rubin's thoughts: [12]

One imputed value can not be correct in general.

➡ We need to represent missing values by a **number of imputations**.

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

1. What is Multiple Imputation?

1.1. History & Ideas

Rubin's thoughts: [12]

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.

1. What is Multiple Imputation?

1.1. History & Ideas

Rubin's thoughts: [12]

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 to impute is the **'predictive distribution'** of the missing values given the observed values.

1. What is Multiple Imputation?

1.1. History & Ideas

How to obtain that predictive distribution?

Rubin suggests to

- fit a model to the observed data (“respondents”), and to
 - obtain for each “nonrespondent” the conditional distribution of the missing data (given the observed data) as if he/she was a respondent.
- ➡ We assume nonrespondents are just like respondents, and obtain the predictive distribution from the model of the respondents data.

1. What is Multiple Imputation?

1.1. History & Ideas

How to obtain that predictive distribution?

Rubin suggests to

- fit a model to the observed data (“respondents”), and to
 - obtain for each “nonrespondent” the conditional distribution of the missing data (given the observed data) as if he/she was a respondent.
- ➡ We assume nonrespondents are just like respondents, and obtain the predictive distribution from the model of the respondents data.

Example: survey about age, gender and height

Boys aged 10 – 12 years old answered (on average) that they are 1.45m tall.

- ➡ We assume that boys aged 10 to 12 who did not report their height are also around 1.45m tall.

1. What is Multiple Imputation?

1.1. History & Ideas

How to represent the multiple imputed values?

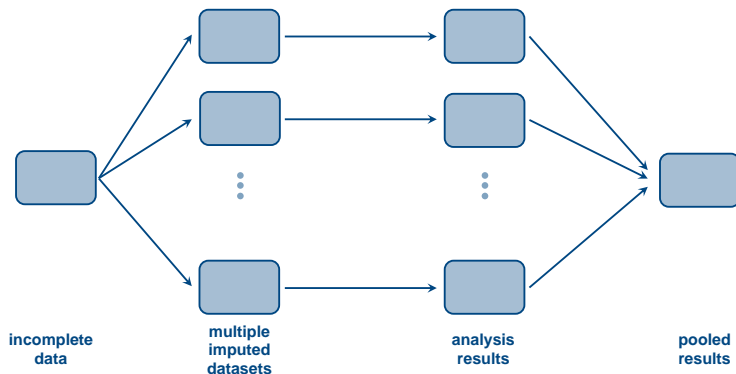
For each missing value, we now have multiple imputed values.

- For each set of imputed values, create a dataset (those datasets agree in the observed values but imputed values differ).
- Analyse each dataset, and
- take the results from each analysis.

➡ We can describe how (much) the **results vary between the imputed datasets**, and calculate summary measures.

1. What is Multiple Imputation?

1.2. Three steps



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

2. Imputation step

2.1. Univariate missing data

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	✓	✓
⋮	⋮	⋮	⋮

2. Imputation step

2.1. Univariate missing data

How can we actually get imputed values?

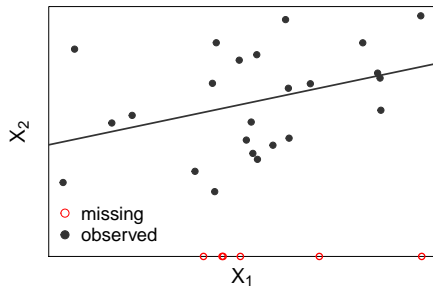
For now: assume only one continuous variable has missing values (**univariate missing data**)

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

Idea: Predict values

Model:

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



2. Imputation step

2.1. Univariate missing data

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	✓	✓
⋮	⋮	⋮	⋮

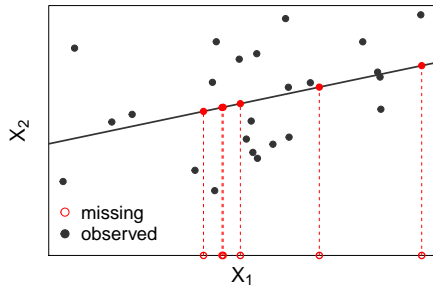
Idea: Predict values

Model:

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

Imputed/predicted value:

$$\hat{x}_{i2} = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i3} + \hat{\beta}_3 x_{i4}$$



2. Imputation step

2.1. Univariate missing data

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.

2. Imputation step

2.1. Univariate missing data

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).

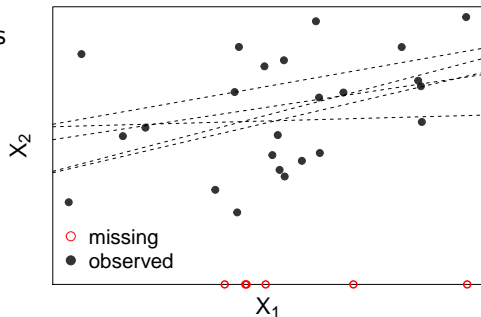
2. Imputation step

2.1. Univariate missing data

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**.



2. Imputation step

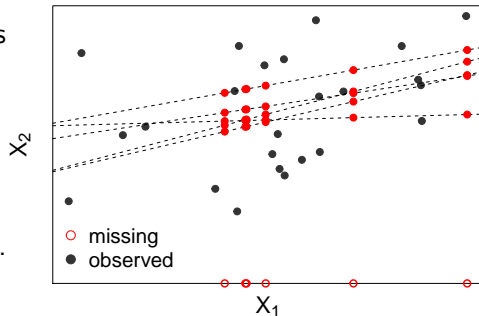
2.1. Univariate missing data

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**.



2. Imputation step

2.1. Univariate missing data

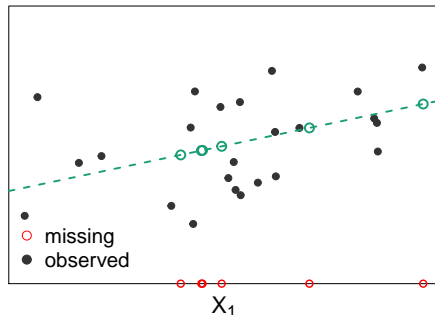
Taking into account the prediction error:

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

X_2



2. Imputation step

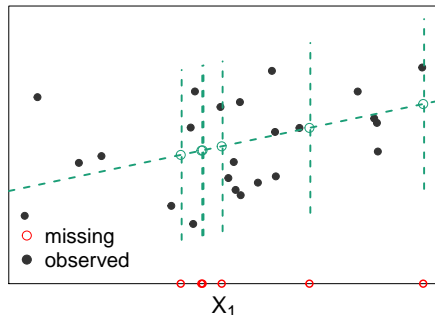
2.1. Univariate missing data

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 .



2. Imputation step

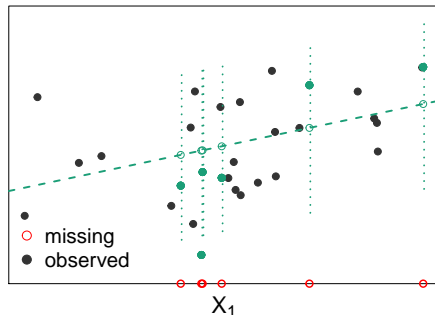
2.1. Univariate missing data

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 .



2. Imputation step

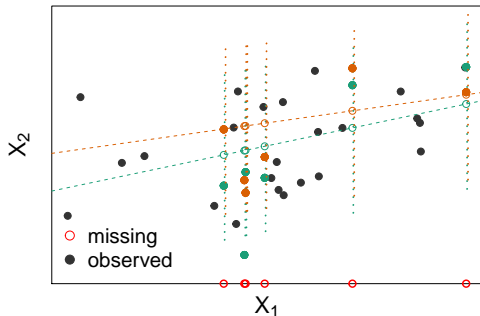
2.1. Univariate missing data

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 .



2. Imputation step

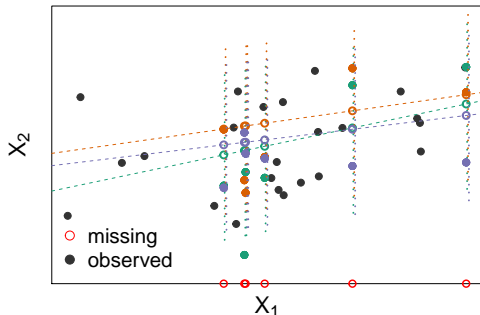
2.1. Univariate missing data

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 .



2. Imputation step

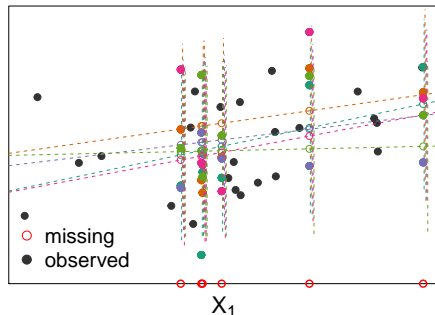
2.1. Univariate missing data

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.

2. Imputation step

2.2. Multivariate missing data

Multivariate missing data:

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

2. Imputation step

2.2. Multivariate missing data

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_2	X_3	X_4
✓	NA	✓	✓
✓	NA	NA	✓
✓	NA	NA	NA
⋮	⋮	⋮	⋮

2. Imputation step

2.2. Multivariate missing data

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_2	X_3	X_4
✓	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
⋮	⋮	⋮	⋮

2. Imputation step

2.2. Multivariate missing data

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

2. Imputation step

2.2. Multivariate missing data

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

(more details later)

2. Imputation step

2.2. Multivariate missing data

Markov Chain Monte Carlo

is a technique to **draw samples from a complex probability distribution** by creating a chain of random variables (a Markov Chain). The distribution each element in the chain is sampled from depends on the value of the previous element. When certain conditions are met, the chain eventually stabilizes and by continuing to sample elements of the chain a sample from the complex distribution of interest can be obtained.

2. Imputation step

2.2. Multivariate missing data

Markov Chain Monte Carlo

is a technique to **draw samples from a complex probability distribution** by creating a chain of random variables (a Markov Chain). The distribution each element in the chain is sampled from depends on the value of the previous element. When certain conditions are met, the chain eventually stabilizes and by continuing to sample elements of the chain a sample from the complex distribution of interest can be obtained.

Gibbs sampling

is an MCMC method where a **sample from a multivariate distribution** is obtained by repeatedly drawing from each of the univariate full conditional distributions instead.

2. Imputation step

2.3. FCS/MICE

MICE (**M**ultiple **I**mputation using **C**hained **E**quations) or
FCS (multiple imputation using **F**ully **C**onditional **S**pecification)

extends univariable imputation to the setting with multivariate non-monotone missingness:

MICE/FCS

- imputes multivariate missing data on a variable-by-variable basis,
- using the technique for univariate missing data.

2. Imputation step

2.3. FCS/MICE

MICE (**M**ultiple **I**mputation using **C**hained **E**quations) or
FCS (multiple imputation using **F**ully **C**onditional **S**pecification)

extends univariable imputation to the setting with multivariate non-monotone missingness:

MICE/FCS

- imputes multivariate missing data on a variable-by-variable basis,
- using the technique for univariate missing data.

Moreover, MICE/FCS is

- an iterative procedure, specifically
- a Markov Chain Monte Carlo (MCMC) method,
- uses the idea of the Gibbs sampler, and
- is a Gibbs sampler if the conditional distributions are compatible (we will come back to this)

2. Imputation step

2.3. FCS/MICE

Notation

- X : $n \times p$ data matrix with n rows and p variables x_1, \dots, x_p
- R : $n \times p$ missing indicator matrix containing 0 (missing) or 1 (observed)

$$\mathbf{X} = \begin{array}{c|ccc} & X_{-2} & X_2 & X_{-2} \\ \hline x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ x_{2,1} & x_{2,2} & \dots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{array}$$

$$\mathbf{R} = \begin{array}{cccc} R_{1,1} & R_{1,2} & \dots & R_{1,p} \\ R_{2,1} & R_{2,2} & \dots & R_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ R_{n,1} & R_{n,2} & \dots & R_{n,p} \end{array}$$

2. Imputation step

2.3. FCS/MICE

Notation

- X : $n \times p$ data matrix with n rows and p variables x_1, \dots, x_p
- R : $n \times p$ missing indicator matrix containing 0 (missing) or 1 (observed)

$$X = \begin{array}{c|ccc} & X_{-2} & X_2 & X_{-2} \\ \hline & x_{1,1} & x_{1,2} & \dots & x_{1,p} \\ & x_{2,1} & x_{2,2} & \dots & x_{2,p} \\ & \vdots & \vdots & \ddots & \vdots \\ & x_{n,1} & x_{n,2} & \dots & x_{n,p} \end{array}$$

$$R = \begin{array}{cccc} R_{1,1} & R_{1,2} & \dots & R_{1,p} \\ R_{2,1} & R_{2,2} & \dots & R_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ R_{n,1} & R_{n,2} & \dots & R_{n,p} \end{array}$$

For example:

$$X = \begin{array}{c|cccc} & X_1 & X_2 & X_3 & X_4 \\ \hline & \checkmark & \text{NA} & \checkmark & \checkmark \\ & \checkmark & \checkmark & \text{NA} & \text{NA} \\ & \checkmark & \text{NA} & \checkmark & \text{NA} \end{array}$$

$$\Rightarrow R = \begin{array}{cccc} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{array}$$

2. Imputation step

2.3. FCS/MICE

Algorithm 1 MICE algorithm [17] for **one** imputed dataset

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

2. Imputation step

2.3. FCS/MICE

Algorithm 1 MICE algorithm [17] for **one** imputed dataset

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

5: for  $t$  in  $1, \dots, T$ : ▷ loop through iterations
6:   for  $j$  in  $1, \dots, p$ : ▷ loop through variables

10:   end for
11: end for
```

2. Imputation step

2.3. FCS/MICE

Algorithm 1 MICE algorithm [17] for **one** imputed dataset

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

5: for  $t$  in  $1, \dots, T$ : ▷ loop through iterations
6:   for  $j$  in  $1, \dots, p$ : ▷ loop through variables
7:     Define currently complete data except  $X_j$ 
      $\dot{X}_{-j}^t = (\dot{X}_1^t, \dots, \dot{X}_{j-1}^t, \dot{X}_{j+1}^{t-1}, \dots, \dot{X}_p^{t-1})$ .

10:   end for
11: end for
```

2. Imputation step

2.3. FCS/MICE

Algorithm 1 MICE algorithm [17] for **one** imputed dataset

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

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

10:   end for
11: end for
```

2. Imputation step

2.3. FCS/MICE

Algorithm 1 MICE algorithm [17] for **one** imputed dataset

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

5: for  $t$  in  $1, \dots, T$ : ▷ loop through iterations
6:   for  $j$  in  $1, \dots, p$ : ▷ loop through variables
7:     Define currently complete data except  $X_j$ 
      $\dot{X}_{-j}^t = (\dot{X}_1^t, \dots, \dot{X}_{j-1}^t, \dot{X}_{j+1}^{t-1}, \dots, \dot{X}_p^{t-1})$ .
8:     Draw parameters  $\dot{\theta}_j^t \sim p(\theta_j^t \mid X_j^{obs}, \dot{X}_{-j}^t, R)$ .
9:     Draw imputations  $\dot{X}_j^t \sim P(X_j^{mis} \mid \dot{X}_{-j}^t, R, \dot{\theta}_j^t)$ .
10:   end for
11: end for
```

2. Imputation step

2.3. FCS/MICE

Algorithm 1 MICE algorithm [17] for **one** imputed dataset

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

5: for  $t = 1$ : ▷ loop through iterations
6:   for  $j = 1$ : ▷ loop through variables
7:     Define currently complete data except  $X_1$ 
      $\dot{X}_{-1}^1 = (\dot{X}_2^0, \dot{X}_3^0, \dot{X}_4^0)$ .
8:     Draw parameters  $\dot{\theta}_1^1 \sim p(\theta_1^1 \mid X_1^{obs}, \dot{X}_{-1}^1, R)$ .
9:     Draw imputations  $\dot{X}_1^1 \sim P(X_1^{mis} \mid \dot{X}_{-1}^1, R, \dot{\theta}_1^1)$ .
10:   end for
11: end for
```

2. Imputation step

2.3. FCS/MICE

Algorithm 1 MICE algorithm [17] for **one** imputed dataset

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

5: for  $t = 1$ : ▷ loop through iterations
6:   for  $j = 2$ : ▷ loop through variables
7:     Define currently complete data except  $X_2$ 
      $\dot{X}_{-2}^1 = (\dot{X}_1^1, \dot{X}_3^0, \dot{X}_4^0)$ .
8:     Draw parameters  $\dot{\theta}_2^1 \sim p(\theta_2^1 \mid X_2^{obs}, \dot{X}_{-2}^1, R)$ .
9:     Draw imputations  $\dot{X}_2^1 \sim P(X_2^{mis} \mid \dot{X}_{-2}^1, R, \dot{\theta}_2^1)$ .
10:   end for
11: end for
```

2. Imputation step

2.3. FCS/MICE

Algorithm 1 MICE algorithm [17] for **one** imputed dataset

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

5: for  $t = 1$ : ▷ loop through iterations
6:   for  $j = 3$ : ▷ loop through variables
7:     Define currently complete data except  $X_3$ 
      $\dot{X}_{-3}^1 = (\dot{X}_1^1, \dot{X}_2^1, \dot{X}_4^0)$ .
8:     Draw parameters  $\dot{\theta}_3^1 \sim p(\theta_3^1 \mid X_3^{obs}, \dot{X}_{-3}^1, R)$ .
9:     Draw imputations  $\dot{X}_3^1 \sim P(X_3^{mis} \mid \dot{X}_{-3}^1, R, \dot{\theta}_3^1)$ .
10:   end for
11: end for
```

2. Imputation step

2.3. FCS/MICE

Algorithm 1 MICE algorithm [17] for **one** imputed dataset

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

5: for  $t = 1$ : ▷ loop through iterations
6:   for  $j = 4$ : ▷ loop through variables
7:     Define currently complete data except  $X_4$ 
      $\dot{X}_{-4}^1 = (\dot{X}_1^1, \dot{X}_2^1, \dot{X}_3^1)$ .
8:     Draw parameters  $\dot{\theta}_4^1 \sim p(\theta_4^1 \mid X_4^{obs}, \dot{X}_{-4}^1, R)$ .
9:     Draw imputations  $\dot{X}_4^1 \sim P(X_4^{mis} \mid \dot{X}_{-4}^1, R, \dot{\theta}_4^1)$ .
10:   end for
11: end for
```

2. Imputation step

2.3. FCS/MICE

Algorithm 1 MICE algorithm [17] for **one** imputed dataset

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

5: for  $t = 2$ : ▷ loop through iterations
6:   for  $j = 1$ : ▷ loop through variables
7:     Define currently complete data except  $X_1$ 
      $\dot{X}_{-1}^2 = (\dot{X}_2^1, \dot{X}_3^1, \dot{X}_4^1)$ .
8:     Draw parameters  $\dot{\theta}_1^2 \sim p(\theta_1^2 \mid X_1^{obs}, \dot{X}_{-1}^2, R)$ .
9:     Draw imputations  $\dot{X}_1^2 \sim P(X_1^{mis} \mid \dot{X}_{-1}^2, R, \dot{\theta}_1^2)$ .
10:   end for
11: end for
```

2. Imputation step

2.3. FCS/MICE

Algorithm 1 MICE algorithm [17] for **one** imputed dataset

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

5: for  $t = 2$ : ▷ loop through iterations
6:   for  $j = 2$ : ▷ loop through variables
7:     Define currently complete data except  $X_2$ 
      $\dot{X}_{-2}^2 = (\dot{X}_1^2, \dot{X}_3^1, \dot{X}_4^1)$ .
8:     Draw parameters  $\dot{\theta}_2^2 \sim p(\theta_2^2 \mid X_2^{obs}, \dot{X}_{-2}^2, R)$ .
9:     Draw imputations  $\dot{X}_2^2 \sim P(X_2^{mis} \mid \dot{X}_{-2}^2, R, \dot{\theta}_2^2)$ .
10:   end for
11: end for
```

2. Imputation step

2.3. FCS/MICE

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

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

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

One run through the algorithm ➡ one imputed dataset.

2. Imputation step

2.3. FCS/MICE

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

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

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**

2. Imputation step

2.3. FCS/MICE

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

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

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**

We refer to the **sequence of imputations** for one missing value, from starting value to final iteration, as a **chain**. Each run through the MICE algorithm produces one chain per missing value.

2. Imputation step

2.3. FCS/MICE

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.

2. Imputation step

2.3. FCS/MICE

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

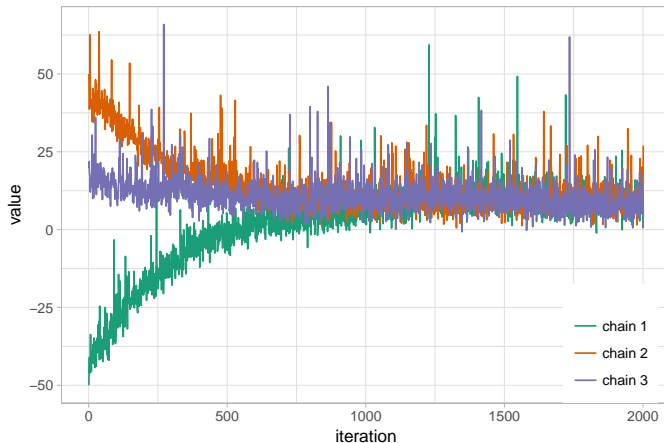
(Note: the imputed value will still vary between iterations.)

How to evaluate convergence?

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

2. Imputation step

2.4. Checking convergence



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

2. Imputation step

2.4. Checking convergence

In imputation we have

- several **variables** with missing values (e.g., p)
 - several missing **values** in each of these variables
 - m **chains** for each missing value
- ➡ possibly a large number of MCMC chain

To check all chains separately could be very time consuming in large datasets (and storing all iterations from all imputed values is inefficient).

2. Imputation step

2.4. Checking convergence

In imputation we have

- several **variables** with missing values (e.g., p)
 - several missing **values** in each of these variables
 - m **chains** for each missing value
- ➔ possibly a large number of MCMC chain

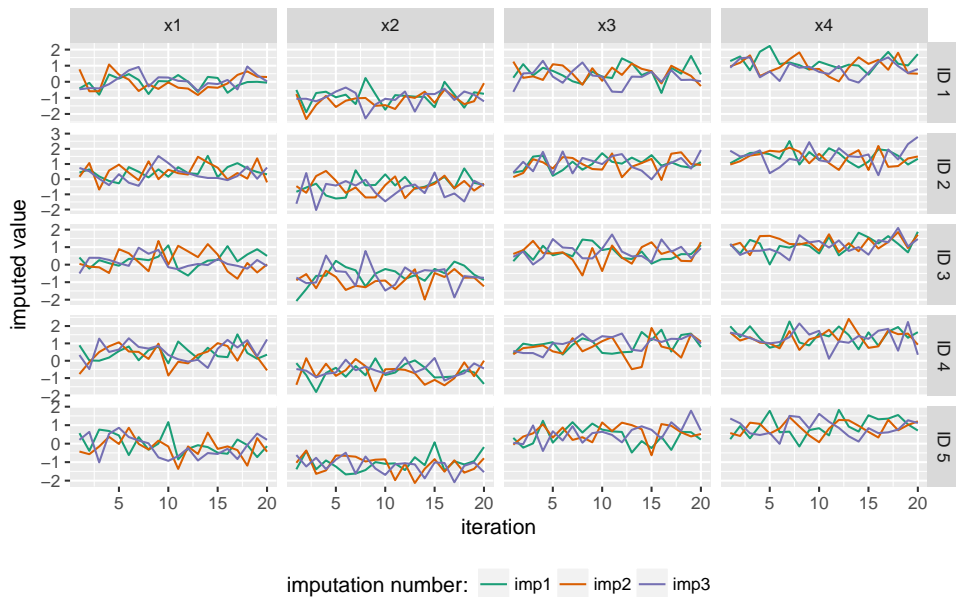
To check all chains separately could be very time consuming in large datasets (and storing all iterations from all imputed values is inefficient).

Alternative: Calculate and plot a summary (e.g., the mean) of the imputed values over all subjects, separately per chain and variable

➔ only $m \times p$ chains to check

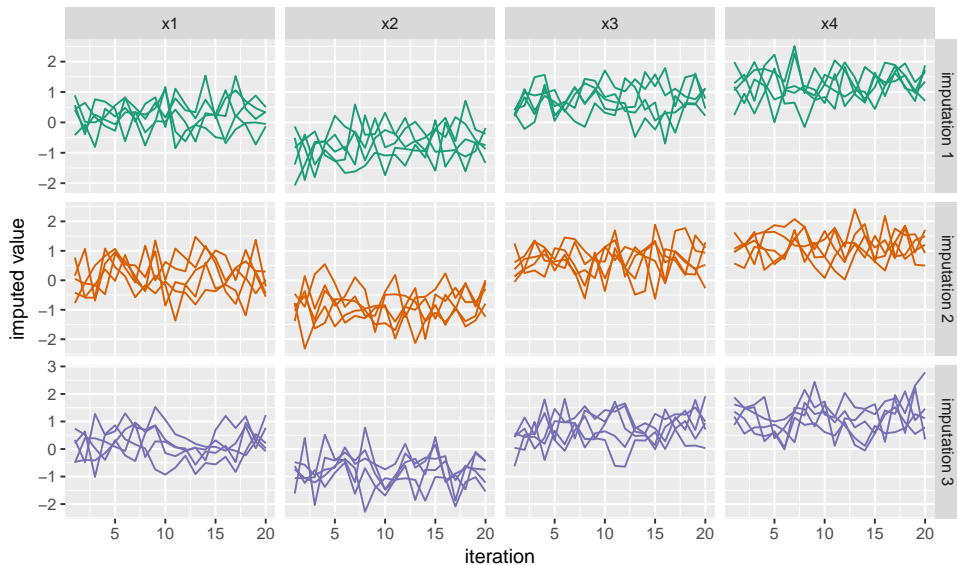
2. Imputation step

2.4. Checking convergence



2. Imputation step

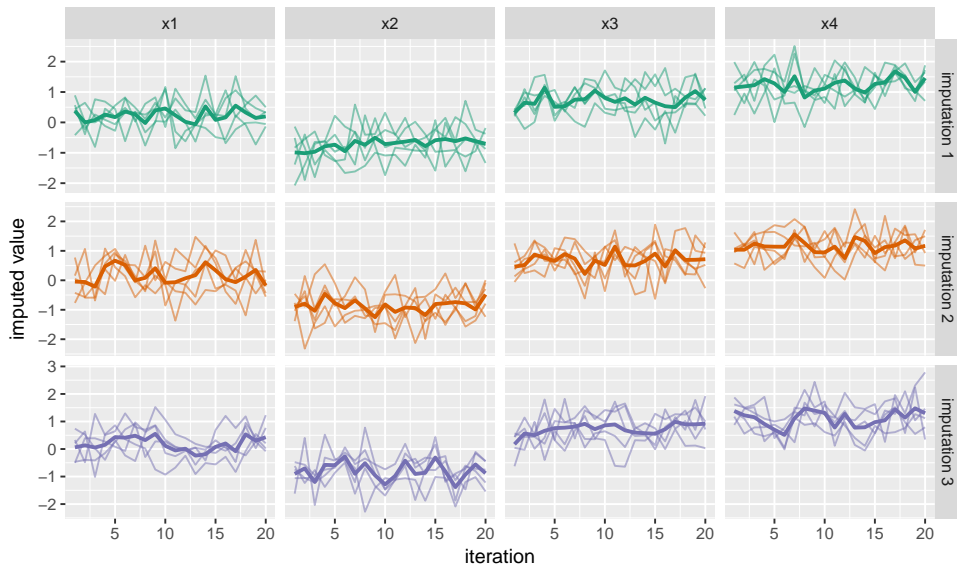
2.4. Checking convergence



imputation number: 1 2 3

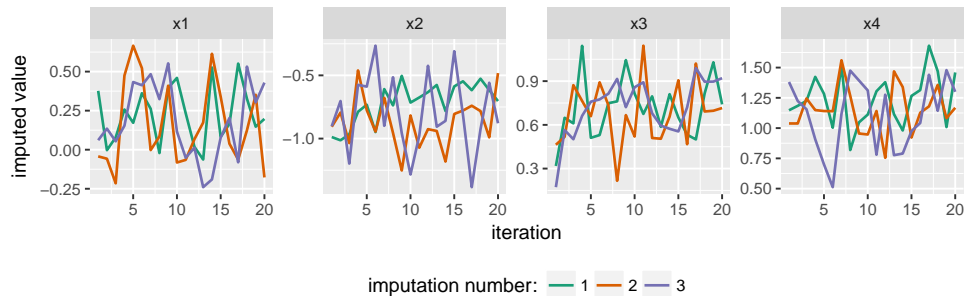
2. Imputation step

2.4. Checking convergence



2. Imputation step

2.4. Checking convergence



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
\vdots	\vdots	\vdots	\vdots

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
\vdots	\vdots	\vdots	\vdots

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
\vdots	\vdots	\vdots	\vdots

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
\vdots	\vdots	\vdots	\vdots

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
\vdots	\vdots	\vdots	\vdots

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
\vdots	\vdots	\vdots	\vdots

Analysis model of interest, e.g.,

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

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
\vdots	\vdots	\vdots	\vdots

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
\vdots	\vdots	\vdots	\vdots

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
\vdots	\vdots	\vdots	\vdots

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	0.35	0.21
β_1	0.14	0.02
β_2	-0.64	0.03
β_3	0.18	0.03

	est.	se
β_0	0.44	0.23
β_1	0.12	0.01
β_2	-0.61	0.03
β_3	0.22	0.03

	est.	se
β_0	0.19	0.21
β_1	0.14	0.01
β_2	-0.59	0.03
β_3	0.2	0.03

4. Pooling

4.1. Why pooling?

Recall from slide 6:

We need to represent missing values by a **number of imputations**.

➡ m imputed datasets

4. Pooling

4.1. Why pooling?

Recall from slide 6:

We need to represent missing values by a **number of imputations**.

➡ m imputed datasets

From the different imputed datasets we get **different sets of parameter estimates**, each of them with a standard error, representing the uncertainty about the estimate.

4. Pooling

4.1. Why pooling?

Recall from slide 6:

We need to represent missing values by a **number of imputations**.

➡ m imputed datasets

From the different imputed datasets we get **different sets of parameter estimates**, each of them with a standard error, representing the uncertainty about the estimate.

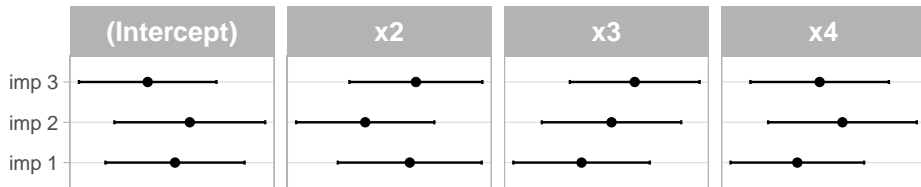
We want to **summarize** the results and describe **how (much) the results vary** between the imputed datasets.

4. Pooling

4.1. Why 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)

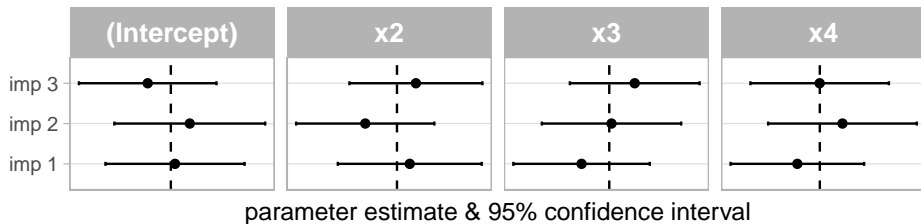


parameter estimate & 95% confidence interval

4. Pooling

4.1. Why pooling?

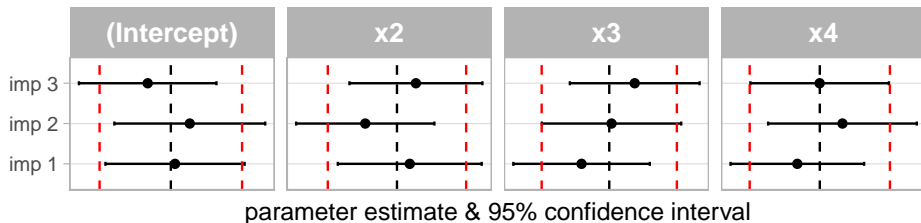
To summarize the results, we can take the mean of the results from the separate analyses. This is the **pooled point estimate**.



4. Pooling

4.1. Why pooling?

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

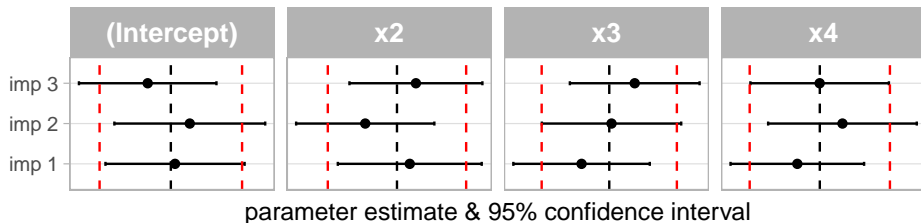


But does the same work for the std. error (or bounds of the CIs)?

4. Pooling

4.1. Why 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 std. error (or bounds of the CIs)?

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

4. Pooling

4.2. Rubin's Rules

The most commonly used method to pool results from analyses of multiply imputed data was introduced by Rubin [10], 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., $\text{var}(\beta) = \text{se}(\beta)^2$)

Pooled parameter estimate:

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

4. Pooling

4.2. Rubin's Rules

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

Average within imputation variance:

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

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:

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

4. Pooling

4.2. Rubin's Rules

Confidence intervals for pooled estimates can be obtained using the **pooled standard error** \sqrt{T} and a **reference t distribution** with degrees of freedom

$$\nu = (m - 1) (1 + r_m^{-1})^2,$$

where $r_m = \frac{(B+B/m)}{\bar{U}}$ is the relative increase in variance that is due to the missing values.

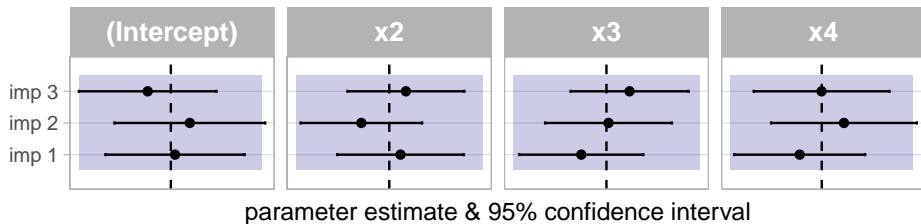
The $(1 - \alpha)$ **100% confidence interval** is then

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

where t_ν is the $\alpha/2$ quantile of the t distribution with ν degrees of freedom.

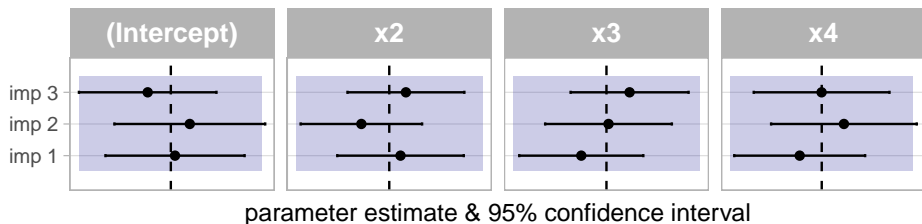
4. Pooling

4.2. Rubin's Rules



4. Pooling

4.2. Rubin's Rules



The corresponding **p-value** is the probability

$$Pr \left\{ F_{1,\nu} > (Q_0 - \bar{Q}_m)^2 / T \right\},$$

where $F_{1,\nu}$ is a random variable that has an F distribution with 1 and ν degrees of freedom, and Q_0 is the null hypothesis value (typically zero).

Quiz

To reiterate the content of the above sections, you can take the corresponding quiz. An interactive version can be found at

https://emcbiostatistics.shinyapps.io/MICourse_Quiz_PartI

or you can download an html version from Canvas
(Files > Principal documents > Multiple Imputation > Quiz_PartI_static.html).

5. A closer look at the imputation step

5.1. Bayesian multiple imputation

The imputation step consists itself of two (or three) steps:

0. Specification of the imputation model,
1. **estimation** or sampling **of the parameters**, and
2. **drawing imputed values** from the predictive distribution.

5. A closer look at the imputation step

5.1. Bayesian multiple imputation

The imputation step consists itself of two (or three) steps:

0. Specification of the imputation model,
1. **estimation** or sampling **of the parameters**, and
2. **drawing imputed values** from the predictive distribution.

Notation:

Let \mathbf{y} be the incomplete covariate to be imputed, and \mathbf{X} the design matrix of other (complete or imputed) variables.

$$\mathbf{y} = \left\{ \begin{array}{l} \mathbf{y}_{obs} \\ \mathbf{y}_{mis} \end{array} \right\} \left[\begin{array}{c} y_1 \\ \vdots \\ y_q \\ NA \\ \vdots \\ NA \end{array} \right]$$
$$\mathbf{X} = \left\{ \begin{array}{l} \mathbf{X}_{obs} \\ \mathbf{X}_{mis} \end{array} \right\} \left[\begin{array}{cccc} 1 & x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_{q1} & \dots & x_{qp} \\ 1 & x_{q+1,1} & \dots & x_{q+1,p} \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{array} \right]$$

5. A closer look at the imputation step

5.1. Bayesian multiple imputation

In the **Bayesian framework**, **everything unknown** or unobserved is considered as a **random variable**. Here, this includes for example regression coefficients β , residual variance σ^2 and missing values \mathbf{y}_{mis} and \mathbf{X}_{mis} .

5. A closer look at the imputation step

5.1. Bayesian multiple imputation

In the **Bayesian framework**, **everything unknown** or unobserved is considered as a **random variable**. Here, this includes for example regression coefficients β , residual variance σ^2 and missing values y_{mis} and X_{mis} .

Random variables have a **probability distribution**. The **expectation** of that distribution quantifies where which **values** of the random variable are **most likely**, the **variance** is a measure of the **uncertainty** about the values.

5. A closer look at the imputation step

5.1. Bayesian multiple imputation

In the **Bayesian framework**, **everything unknown** or unobserved is considered as a **random variable**. Here, this includes for example regression coefficients β , residual variance σ^2 and missing values y_{mis} and X_{mis} .

Random variables have a **probability distribution**. The **expectation** of that distribution quantifies where which **values** of the random variable are **most likely**, the **variance** is a measure of the **uncertainty** about the values.

In **Bayesian imputation**, the **information obtained from the observed data** is used to **estimate the probability distributions** for the missing values and unknown parameters, and values are **imputed by draws** from that posterior (= after having seen the data) distribution.

5. A closer look at the imputation step

5.1. Bayesian multiple imputation

To determine the **expectation** of the posterior distribution of the missing values, usually a **regression model** is used, that depends on the unknown coefficients β .

$$\mathbb{E}(\mathbf{y}_{mis} \mid \mathbf{X}, \beta) = f(\mathbf{X}_{mis}\beta)$$

The posterior distribution of β and σ , $p(\beta, \sigma \mid \mathbf{y}_{obs}, \mathbf{X}_{obs})$, is estimated from the corresponding regression model on the observed data.

5. A closer look at the imputation step

5.1. Bayesian multiple imputation

To determine the **expectation** of the posterior distribution of the missing values, usually a **regression model** is used, that depends on the unknown coefficients β .

$$\mathbb{E}(\mathbf{y}_{mis} \mid \mathbf{X}, \beta) = f(\mathbf{X}_{mis}\beta)$$

The posterior distribution of β and σ , $p(\beta, \sigma \mid \mathbf{y}_{obs}, \mathbf{X}_{obs})$, is estimated from the corresponding regression model on the observed data.

To **impute** missing values, while **taking into account the uncertainty about β and σ** , the estimated posterior distributions of the missing values and parameters are multiplied

$$p(\mathbf{y}_{mis} \mid \mathbf{X}_{mis}, \beta, \sigma) p(\beta, \sigma \mid \mathbf{y}_{obs}, \mathbf{X}_{obs})$$

In practice, this can be implemented by first making a draw from the posterior distributions of β and σ , and plugging the values into the distribution of \mathbf{y}_{obs} .

5. A closer look at the imputation step

5.1. Bayesian multiple imputation

Example: We assume that \mathbf{y} given \mathbf{X} is approximately normal.

Then $p(\mathbf{y}_{mis} \mid \mathbf{X}_{mis}, \boldsymbol{\beta}, \sigma)$ is a normal distribution and we can

- draw $\tilde{\boldsymbol{\beta}}$ from $p(\boldsymbol{\beta} \mid \mathbf{y}_{obs}, \mathbf{X}_{obs})$,
- draw $\tilde{\sigma}$ from $p(\sigma \mid \mathbf{y}_{obs}, \mathbf{X}_{obs})$,
- draw $\tilde{\mathbf{y}}_{mis}$ from a normal distribution with mean (= expectation) $\mathbf{X}_{mis}\tilde{\boldsymbol{\beta}}$ and variance $\tilde{\sigma}^2$.

This is actually the approach we have seen previously on Slides 12/13 and 19.

5. A closer look at the imputation step

5.2. Bootstrap multiple imputation

An alternative approach is to capture the uncertainty with **bootstrap** sampling.

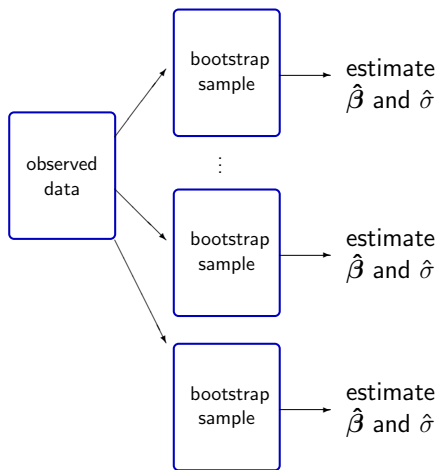
In empirical **Bootstrap**, (many) replications of the data are created by repeatedly drawing values from the original data.

5. A closer look at the imputation step

5.2. Bootstrap multiple imputation

An alternative approach is to capture the uncertainty with **bootstrap** sampling.

In empirical **Bootstrap**, (many) replications of the data are created by repeatedly drawing values from the original data.



Bootstrap samples can contain some **observations multiple times** and some **observations not at all**.

The statistic of interest is then calculated on each of the bootstrap samples.

5. A closer look at the imputation step

5.2. Bootstrap multiple imputation

In **bootstrap multiple imputation**,

- **one bootstrap sample** of the **observed data** is created per imputation,
- the (least squares or maximum likelihood) estimates of the parameters are calculated from

$$\mathbf{y}_{obs} = \mathbf{X}_{obs} \underset{\downarrow}{\hat{\boldsymbol{\beta}}} + \varepsilon_{obs} \underset{\downarrow}{\hat{\sigma}} \quad (\text{step 1}).$$

- Imputed values are sampled from $p(\mathbf{y}_{mis} \mid \mathbf{X}_{mis}, \hat{\boldsymbol{\beta}}, \hat{\sigma})$ (step 2).

5. A closer look at the imputation step

5.2. Bootstrap multiple imputation

In **bootstrap multiple imputation**,

- **one bootstrap sample** of the **observed data** is created per imputation,
- the (least squares or maximum likelihood) estimates of the parameters are calculated from

$$\mathbf{y}_{obs} = \mathbf{X}_{obs} \underset{\downarrow \hat{\beta}}{\beta} + \underset{\downarrow \hat{\sigma}}{\varepsilon_{obs}} \quad (\text{step 1}).$$

- Imputed values are sampled from $p(\mathbf{y}_{mis} \mid \mathbf{X}_{mis}, \hat{\beta}, \hat{\sigma})$ (step 2).

Analogous to Bayesian multiple imputation, for a normal imputation model, $p()$ is the normal distribution and

$$\tilde{\mathbf{y}}_{mis} = \mathbf{X}_{mis} \hat{\beta} + \tilde{\varepsilon}$$

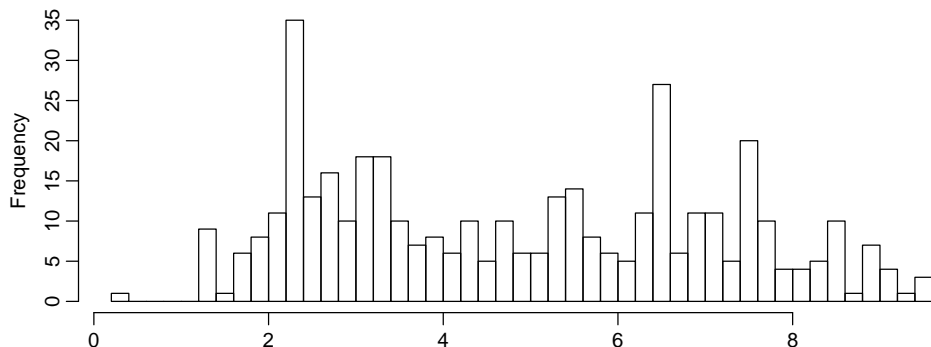
where $\tilde{\varepsilon}$ is drawn independently from $N(0, \hat{\sigma}^2)$.

5. A closer look at the imputation step

5.3. Semi-parametric imputation

Both Bayesian and bootstrap multiple imputation sample imputed values from a distribution $p()$ in step 2.

Sometimes, the empirical distribution can not be adequately approximated by a known probability distribution.



5. A closer look at the imputation step

5.3. Semi-parametric imputation

Predictive Mean Matching (PMM) was developed to provide a semi-parametric approach to imputation for settings where the normal distribution is not a good choice for the predictive distribution.[8, 9]

The idea is to **find cases in the observed data that are similar to the cases with missing values** and to 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.

5. A closer look at the imputation step

5.3. Semi-parametric imputation

The steps in PMM:

1. Obtain parameter estimates for $\hat{\beta}$ and $\hat{\sigma}$ (see later)
2. Calculate the predicted values for the observed data

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

3. Calculate the predicted value for the incomplete data

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

4. For each missing value, find d donor candidates that fulfill a given criterium (details on the next slide).
5. Randomly select one of the donors.

5. A closer look at the imputation step

5.3. Semi-parametric imputation

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

1. The donor is the **(one) case with the smallest absolute difference**
 $|\hat{y}_{mis,i} - \hat{y}_{obs,j}|, j = 1, \dots, q.$

5. A closer look at the imputation step

5.3. Semi-parametric imputation

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

1. The donor is the **(one) case with the smallest absolute difference**
 $|\hat{y}_{mis,i} - \hat{y}_{obs,j}|, j = 1, \dots, q.$
2. Donor candidates are the **d cases with the smallest absolute difference**
 $|\hat{y}_{mis,i} - \hat{y}_{obs,j}|, j = 1, \dots, q.$ The donor is selected randomly from the candidates.

5. A closer look at the imputation step

5.3. Semi-parametric imputation

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

1. The donor is the **(one) case with the smallest absolute difference** $|\hat{y}_{mis,i} - \hat{y}_{obs,j}|$, $j = 1, \dots, q$.
2. Donor candidates are the **d cases with the smallest absolute difference** $|\hat{y}_{mis,i} - \hat{y}_{obs,j}|$, $j = 1, \dots, q$. 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** : $|\hat{y}_{mis,i} - \hat{y}_{obs,j}| < \eta$, $j = 1, \dots, q$. The donor is selected randomly from the candidates.

5. A closer look at the imputation step

5.3. Semi-parametric imputation

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

1. The donor is the **(one) case with the smallest absolute difference** $|\hat{y}_{mis,i} - \hat{y}_{obs,j}|$, $j = 1, \dots, q$.
2. Donor candidates are the **d cases with the smallest absolute difference** $|\hat{y}_{mis,i} - \hat{y}_{obs,j}|$, $j = 1, \dots, q$. 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** : $|\hat{y}_{mis,i} - \hat{y}_{obs,j}| < \eta$, $j = 1, \dots, q$. 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 $|\hat{y}_{mis,i} - \hat{y}_{obs,j}|$. [16]

5. A closer look at the imputation step

5.3. Semi-parametric imputation

Potential issues with donor selection

- Selection criteria 2. - 4., **require the number of candidates d** (or maximal difference η) to be specified. Common choices for d are 3, 5 or 10.
- 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**.

5. A closer look at the imputation step

5.3. Semi-parametric imputation

Potential issues with donor selection

- Selection criteria 2. - 4., **require the number of candidates d** (or maximal difference η) to be specified. Common choices for d are 3, 5 or 10.
 - 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**.
- ➔ 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**.

5. A closer look at the imputation step

5.3. Semi-parametric imputation

Potential issues with donor selection

- Selection criteria 2. - 4., **require the number of candidates d** (or maximal difference η) to be specified. Common choices for d are 3, 5 or 10.
- 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**.
 - ➔ 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**.
- Therefore, using $d = 1$ (selection criterion 1.) is not a good idea. On the other hand, using too many candidates can lead to bad matches.

5. A closer look at the imputation step

5.3. Semi-parametric imputation

Potential issues with donor selection

- Selection criteria 2. - 4., **require the number of candidates d** (or maximal difference η) to be specified. Common choices for d are 3, 5 or 10.
- 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**.
 - ➔ 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**.
- Therefore, using $d = 1$ (selection criterion 1.) is not a good idea. On the other hand, using too many candidates can lead to bad matches.
- Schenker and Taylor [15] proposed an adaptive procedure to select d , but it is not used much in practice.

5. A closer look at the imputation step

5.3. Semi-parametric imputation

For the **sampling of the parameters** (step 1 on slide 43), different approaches have been introduced in the literature:

- Type-0 point estimates $\hat{\beta}$ are used in both prediction models (least squares or maximum likelihood)
- Type-I $\hat{\beta}$ to predict \hat{y}_{obs} ; $\tilde{\beta}$ to predict \hat{y}_{mis} is sampled from the posterior distribution of β (Bayesian) or bootstrapped
- Type-II $\tilde{\beta}$ to predict \hat{y}_{obs} as well as \hat{y}_{mis}
- Type-III different draws $\tilde{\beta}^{(1)}$ and $\tilde{\beta}^{(2)}$ to predict \hat{y}_{obs} and \hat{y}_{mis} , respectively

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

5. A closer look at the imputation step

5.3. Semi-parametric imputation

Another point of consideration is the **choice of the set of data used to train the prediction models**.

In the version presented on slide 43, 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**.

As an alternative, the **model could be trained on the whole data** (using previously imputed values) or to use a **leave-one-out approach** on the observed data.

5. A closer look at the imputation step

5.4. What is implemented in software?

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()`
- ...

1. What is Multiple Imputation?

- Rubin's two **ideas**:
 - Missing values need to be represented by **multiple imputed values**.
 - A **model is necessary** to obtain good imputations.
- Imputed values are obtained from the **predictive distribution** of the missing data, given the observed data.
- Multiple completed datasets are created from the multiple imputed values.
- Multiple imputation has **three steps: Imputation, analysis, pooling**

Summary of Part I (cont.)

2. Imputation step

- Two **sources of variation** need to be taken into account
 - **parameter uncertainty**
 - **random variation**
- **Two approaches** to MI for imputation of non-monotone multivariate missing data
 - **MICE/FCS**
 - **Joint model imputation**
- The MICE algorithm re-uses univariate imputation models by iterating through all incomplete variables, multiple times (**iterations**)
- **Multiple runs** through the algorithm are necessary to create multiple imputed dataset
- The **convergence of the chains** needs to be checked.

Summary of Part I (cont.)

3. Analysis step

- Analyse each imputed dataset the way you would analyse a complete dataset

4. Pooling

- Results from analyses of multiple imputed datasets can be summarized by taking the **average of the regression coefficients**
- For the total variance, **two sources of variation** need to be considered:
 - **within imputation variance**
 - **between imputation variance**

Summary of Part I (cont.)

5. A closer look at the imputation step

- Two **parametric approaches** for imputation:
 - **Bayesian** (sample from posterior distribution of parameters)
 - **Bootstrap** (uses bootstrap samples of the data to estimate parameters)
- **Predictive mean matching** is a semi-parametric alternative (it matches observed and missing cases based on their predicted values).
- In PMM we need to consider
 - **donor selection**
 - **matching type** (how parameters are sampled/estimated),
 - the **set of data** used to calculate/estimate the parameters.
- Bayesian and bootstrap imputation take into account the variation, while many **choices in PMM lead to underestimation of the variation**.

Part II

Multiple Imputation Workflow

Outline of Part II

6. Know your data

- 6.1 Missing data patterns
- 6.2 Data distributions
- 6.3 Correlations & patterns
- 6.4 Why are values missing?
- 6.5 Auxiliary variables

7. Imputation with `mice()`

- 7.1 Main function arguments
- 7.2 Imputation methods
- 7.3 Predictor matrix
- 7.4 Passive imputation
- 7.5 Post processing
- 7.6 Visit sequence
- 7.7 Good to know

8. Convergence & Diagnostics

- 8.1 Logged events
- 8.2 Convergence
- 8.3 Diagnostics

9. Analyse & pool the imputed data

- 9.1 Analysing imputed data
- 9.2 Pooling results
- 9.3 Functions for pooled results

10. Additional functions in `mice()`

- 10.1 Extract & export imputed data
- 10.2 Combining `mids` objects
- 10.3 Adding variables to `mids` objects

11. Multiple Imputation in SPSS

- 11.1 Where to get help
- 11.2 Multiple Imputation Features

6. Know your data

6.1. Missing data patterns

There are several packages in R that provide functions to create and **plot the missing data pattern**.

Examples are:

mice, **VIM**, **Amelia**, **visdat**, **nanian**, ...

To create the plots on the following slides, we need to load the example data

```
datadir <-      # fill in path to folder containing the data  
load(file.path(datadir, "NHANES.RData"))
```

6. Know your data

6.1. Missing data patterns

```
mdp <- mice::md.pattern(NHANES)
head(mdp[, -c(7:14)]) # omit some columns to fit it on the slide
```

```
##      age gender race DM educ smoke hypchol creat albu uricacid bili alc HyperMed
## 572    1      1    1  1    1    1      1    1    1      1    1    1      1  0
## 1      1      1    1  1    1    0      1    1    1      1    1    1      1  1
## 141    1      1    1  1    1    1      1    1    1      1    1    0      1  1
## 17     1      1    1  1    1    1      1    1    1      1    1    1      1  1
## 1063   1      1    1  1    1    1      1    1    1      1    1    1      0  1
## 18     1      1    1  1    1    1      1    1    1      1    1    1      1  1
```

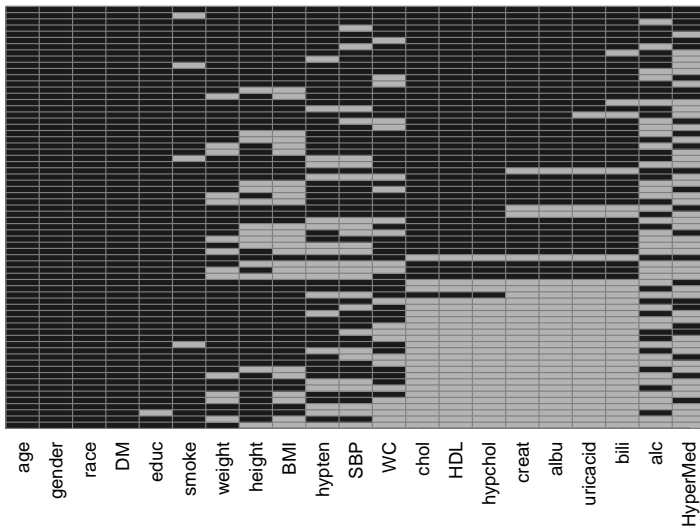
```
tail(mdp[, -c(7:14)])
```

```
##      age gender race DM educ smoke hypchol creat albu uricacid bili alc HyperMed
## 2      1      1    1  1    1    1      0    0    0      0    0    0      1  11
## 2      1      1    1  1    1    1      0    0    0      0    0    0      0  12
## 1      1      1    1  1    0    1      0    0    0      0    0    1      0  12
## 1      1      1    1  1    1    1      0    0    0      0    0    0      0  12
## 2      1      1    1  1    1    1      0    0    0      0    0    0      0  14
##      0      0    0  0    1    4     175   184   184     185   188 627     1606 3975
```

6. Know your data

6.1. Missing data patterns

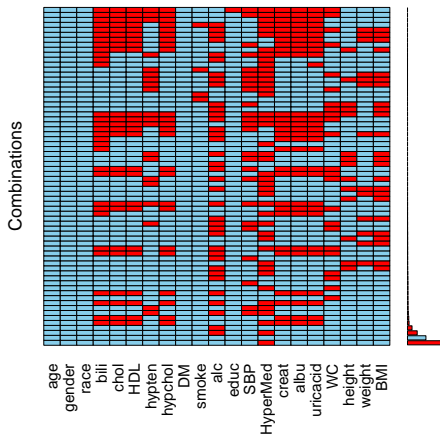
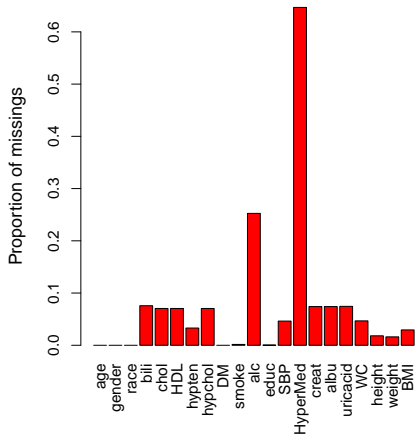
```
par(mar = c(5, 0.5, 1, 3), mgp = c(2, 0.6, 0))  
JointAI::md_pattern(NHANES, print = F, printN = F, yaxis_pars = list(yaxt = 'n'))
```



6. Know your data

6.1. Missing data patterns

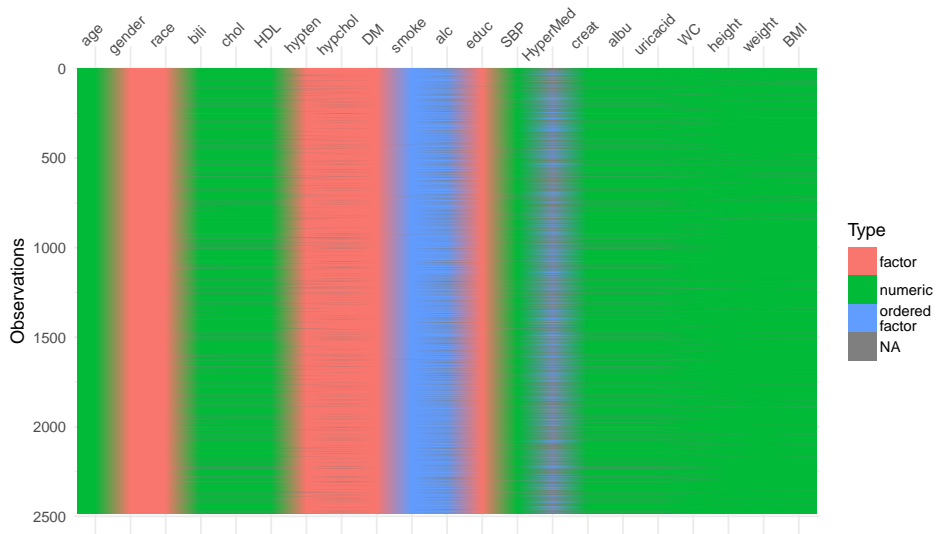
```
par(mar = c(6, 3, 2, 1))  
VIM::aggr(NHANES, prop = T, numbers = FALSE)
```



6. Know your data

6.1. Missing data patterns

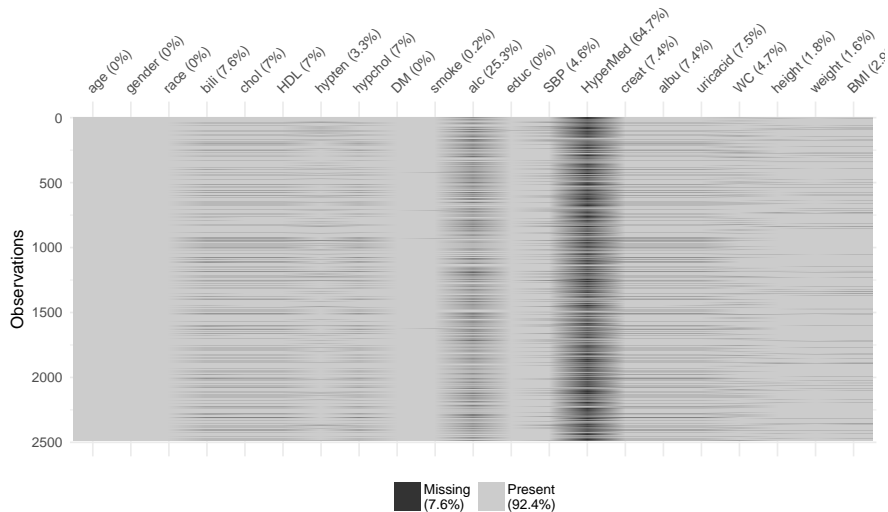
```
visdat::vis_dat(NHANES, sort_type = FALSE)
```



6. Know your data

6.1. Missing data patterns

```
visdat::vis_miss(NHANES)
```



6. Know your data

6.1. Missing data patterns

```
# number and proportion of complete cases
Ncc <- cbind(
  "#" = table(complete.cases(NHANES)),
  "%" = round(100 * table(complete.cases(NHANES))/nrow(NHANES), 2)
)
rownames(Ncc) <- c("incompl.", "complete")
```

```
# number and proportion of missing values per variable
nmis <- cbind("# NA" = sort(colSums(is.na(NHANES))),
              "% NA" = round(sort(colMeans(is.na(NHANES))) * 100, 2))
```

6. Know your data

6.1. Missing data patterns

Number and proportion of (in)complete cases

```
##           #      %  
## incompl. 1911 76.96  
## complete  572 23.04
```

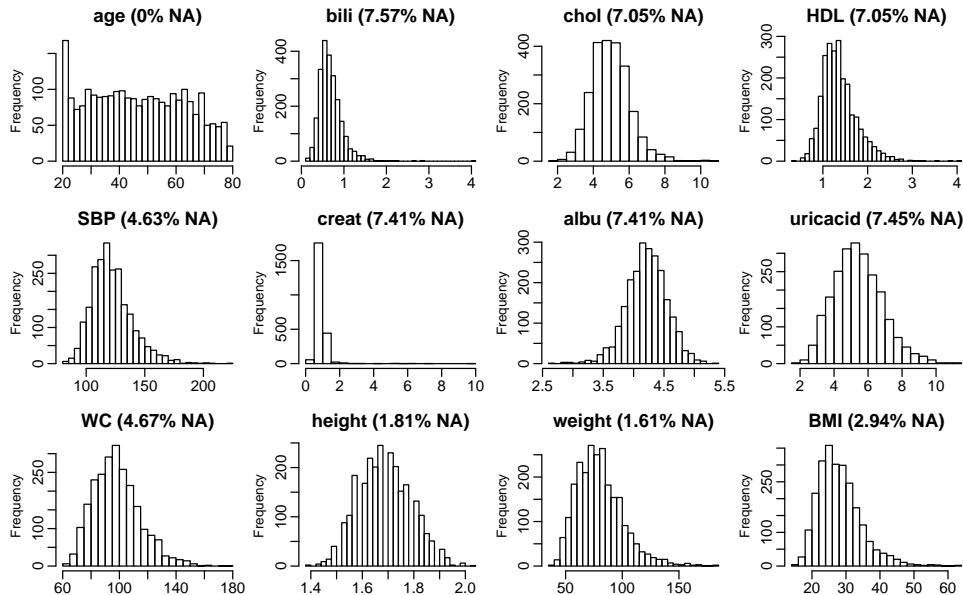
Number and proportion of missing values per variable

```
##      # 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  
## weight    40 1.61  
## height    45 1.81  
## BMI       73 2.94  
## hypten    82 3.30  
## SBP      115 4.63
```

```
##      # NA % NA  
## 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
```

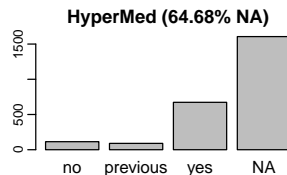
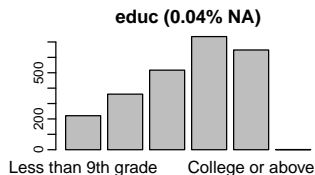
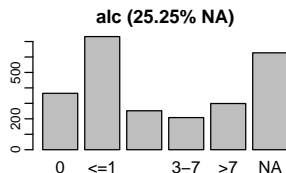
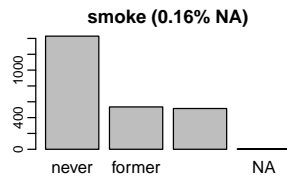
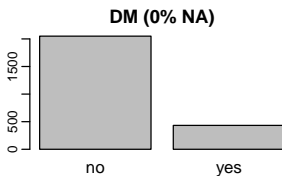
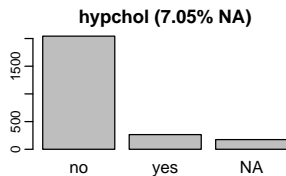
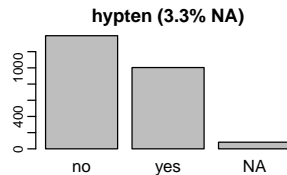
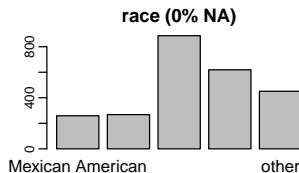
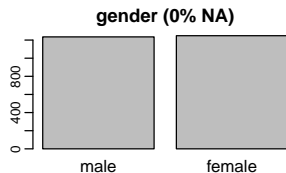
6. Know your data

6.2. Data distributions



6. Know your data

6.2. Data distributions



6. Know your data

6.2. Data distributions

```
# syntax for continuous variables
NHANESnum <- NHANES[, sapply(NHANES, is.numeric)]
par(mfrow = c(3, 4), mar = c(3, 3.2, 0.5, 0.5), mgp = c(2, 0.6, 0))
for (i in 1:ncol(NHANESnum)) {
  hist(NHANESnum[, i], nclass = 30, xlab = names(NHANESnum)[i], main = "")
  legend("topright", bty = "n",
        legend = paste0(round(mean(is.na(NHANESnum[, i]))*100, 2), "% NA"))
}

# syntax for factors
NHANESfac <- NHANES[, sapply(NHANES, is.factor)]
par(mfrow = c(3, 5), mar = c(3, 3.2, 2.5, 0.5), mgp = c(2, 0.6, 0))
for (i in 1:ncol(NHANESfac)) {
  tab <- table(NHANESfac[, i], exclude = NULL)
  names(tab)[is.na(names(tab))] <- "NA"
  barplot(tab, main = paste0(names(NHANESfac)[i], " (",
                             round(mean(is.na(NHANESfac[, i]))*100,
                                     2), "% NA)"))
}
```


6. Know your data

6.3. Correlations & patterns

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")

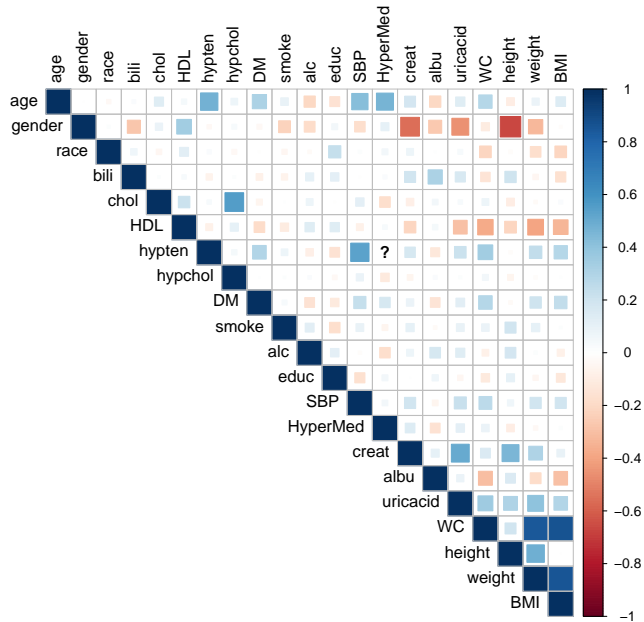
## Warning in cor(sapply(NHANES, as.numeric), use =
"pairwise.complete.obs", : the standard deviation is zero
```

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

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

6. Know your data

6.3. Correlations & patterns



6. Know your data

6.3. Correlations & patterns

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
```

6. Know your data

6.4. Why are values missing?

Knowing your data also means to be 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?

6. Know your data

6.5. Auxiliary variables

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

Good auxiliary variables

- 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.

7. Imputation with `mice()`

7.1. Main function arguments

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

7. Imputation with `mice()`

7.2. Imputation methods

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)

7. Imputation with mice()

7.2. Imputation methods

Change the default imputation method:

E.g.: to use `norm` instead of `pmm` for all continuous incomplete variables, use:

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


7. Imputation with mice()

7.2. Imputation methods

Change the default imputation method:

E.g.: 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.

7. Imputation with mice()

7.2. Imputation methods

Change the default imputation method:

E.g.: 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 `"`.

7. Imputation with mice()

7.2. Imputation methods

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

##	age	gender	race	bili	chol	HDL
##	"	"	"	"pmm"	"pmm"	"pmm"
##	hypoten	hypchol	DM	smoke	alc	educ
##	"logreg"	"logreg"	"	"polr"	"polr"	"polyreg"
##	SBP	HyperMed	creat	albu	uricacid	WC
##	"pmm"	"polr"	"pmm"	"pmm"	"pmm"	"pmm"
##	height	weight	BMI			
##	"pmm"	"pmm"	"pmm"			

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

7. Imputation with mice()

7.3. Predictor matrix

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	chol	HDL	hypten	hypchol	DM	smoke	alc
## age	0	0	0	0	0	0	0	0	0	0	0
## gender	0	0	0	0	0	0	0	0	0	0	0
## race	0	0	0	0	0	0	0	0	0	0	0
## bili	1	1	1	0	1	1	1	1	1	1	1
## chol	1	1	1	1	0	1	1	1	1	1	1
## HDL	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 predictor.
For complete variables all entries are 0.

7. Imputation with `mice()`

7.3. Predictor matrix

Important:

A variable that has **missing values needs to be imputed** in order to be used as 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 chosen imputation models are appropriate (e.g., ordered factors).

7. Imputation with mice()

7.3. Predictor matrix

```
library(mice)
imp0 <- mice(NHANES, maxit = 0,
             defaultMethod = c("norm", "logreg", "polyreg", "polr"))
meth <- imp0$method
meth["educ"] <- "polr"
meth["HyperMed"] <- ""

pred <- imp0$predictorMatrix
pred[, "HyperMed"] <- 0
imp <- mice(NHANES, method = meth, predictorMatrix = pred,
            printFlag = F)
```

7. Imputation with mice()

7.4. Passive imputation

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

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

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

```
##      wgt/hgt^2    BMI
## [1,]      27.25 28.77
## [2,]      23.80 22.94
## [3,]      25.77 24.06
## [4,]      27.56 27.50
## [5,]      23.75 24.07
```

The imputed values of **BMI** are impossible given the corresponding values of **height** and **weight**.

7. Imputation with mice()

7.4. Passive imputation

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

```
table(weight_missing = is.na(NHANES$weight),  
      height_missing = is.na(NHANES$height))
```

```
##           height_missing  
## weight_missing FALSE TRUE  
##           FALSE  2410   33  
##           TRUE    28   12
```

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

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

7. Imputation with `mice()`

7.4. Passive imputation

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**.

7. Imputation with `mice()`

7.4. Passive imputation

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(weight/height^2)"      # formula to impute BMI
pred[c("weight", "height"), "BMI"] <- 0  # prevent feedback
```

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

7. Imputation with mice()

7.4. Passive imputation

Since BMI depends on weight, and the two variables are highly correlated ($\rho=0.87$) 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 an **interaction terms** between incomplete variables (see [17, 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.

7. Imputation with mice()

7.5. Post processing

`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:

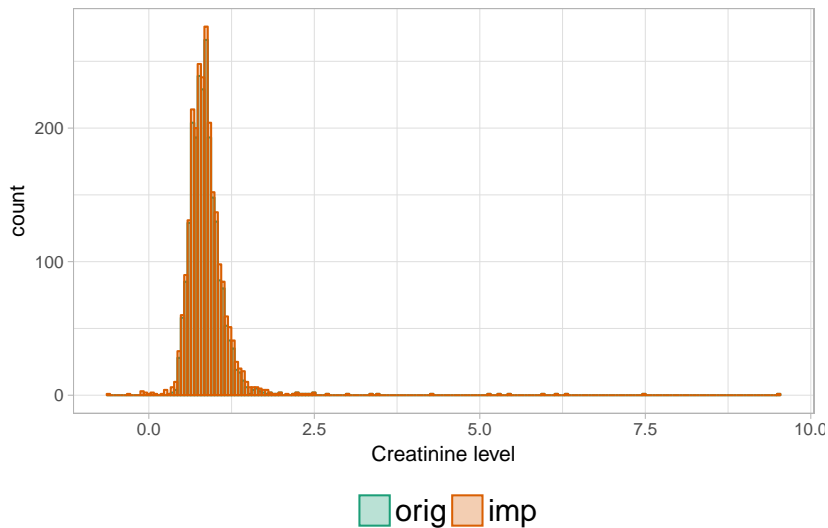
When inspecting the imputed values from `imp`, we find that some imputed values in `creat` are negative.

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

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.  
## -0.6155  0.7000  0.8300  0.8853  0.9900  9.5100
```

7. Imputation with mice()

7.5. Post processing



7. Imputation with mice()

7.5. Post processing

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**.

7. Imputation with mice()

7.5. Post processing

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.

7. Imputation with `mice()`

7.6. Visit sequence

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 `height` and `weight` are drawn, the resulting values of `BMI`, will match `height` and `weight` 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.

7. Imputation with mice()

7.6. Visit sequence

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

```
##      bili      chol      HDL      hypten      hypchol      smoke      alc  
##         4         5         6         7         8        10        11  
##      educ      SBP HyperMed      creat      albu uricacid      WC  
##        12        13        14        15        16        17        18  
##    height    weight      BMI  
##        19        20        21
```

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`.

7. Imputation with mice()

7.6. Visit sequence

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

```
visitSeq
```

```
##      bili      chol      HDL  hypchol      smoke      alc      educ  
##        4        5        6        8        10        11        12  
##      SBP HyperMed      creat      albu uricacid      WC      height  
##       13       14       15       16       17       18       19  
##  weight      BMI  hypten  
##       20       21       7
```

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

7. Imputation with mice()

7.6. Visit sequence

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

```
visitSeq
```

```
##      bili      chol      HDL  hypchol      smoke      alc      educ  
##        4        5        6        8        10        11        12  
##      SBP HyperMed      creat      albu uricacid      WC      height  
##       13       14       15       16       17       18       19  
##  weight      BMI  hypten  
##       20       21       7
```

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

`visitSequence` can also be specified using one of the keywords `"roman"` (left to right), `"arabic"` (right to left), `"monotone"` (sorted in increasing amount of missingness), `"revmonotone"` (reverse of monotone)

7. Imputation with `mice()`

7.7. Good to know

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

- incomplete variables that are not imputed but are specified as predictor,
- 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.

7. Imputation with `mice()`

7.7. Good to know

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

7. Imputation with mice()

A note

*“Please realize that these choices are always needed. Imputation software needs to make default choices. These choices are intended to be useful across a wide range of applications. However, the **default choices are not necessarily the best for the data at hand. There is simply no magical setting that always works**, so often some tailoring is needed.” [17, p. 124]*

8. Convergence & Diagnostics

8.1. Logged events

The log of the automatic changes (slide 89) is returned as part of the `mids` object:

```
head(imp2$loggedEvents)
```

```
##   it im co   dep    meth out
## 1  1  1 10 smoke multinom
## 2  1  1 11  alc multinom
## 3  1  1 12 educ multinom
## 4  1  2 10 smoke multinom
## 5  1  2 11  alc multinom
## 6  1  2 12 educ multinom
```

With columns

<code>it</code>	iteration number
<code>im</code>	imputation number
<code>co</code>	column number in the data
<code>dep</code>	dependent variable
<code>meth</code>	imputation method used
<code>out</code>	names of altered or removed predictors

8. Convergence & Diagnostics

8.2. Convergence

Recall from slides 19 and 23:

mice uses an **iterative algorithm** and imputations from the first few iterations may not be samples from the “correct” distributions.

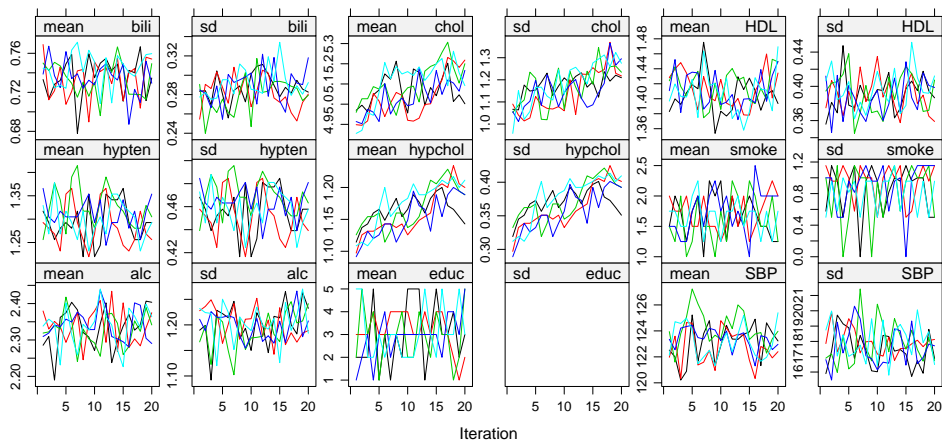
Traceplots can be used to visually assess **convergence**.

In **mice** the function `plot()` produces traceplots of the mean and standard deviation (across subjects) per incomplete variable (see slide 25).

8. Convergence & Diagnostics

8.2. Convergence

```
plot(imp2, layout = c(6, 3))
```



8. Convergence & Diagnostics

8.2. Convergence

The traceplots show that the imputations for `chol` and `hypchol` have an upward trend.

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.

8. Convergence & Diagnostics

8.3. Diagnostics

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 to 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.

8. Convergence & Diagnostics

8.3. Diagnostics

mice provides several functions for visual diagnosis of imputed values:

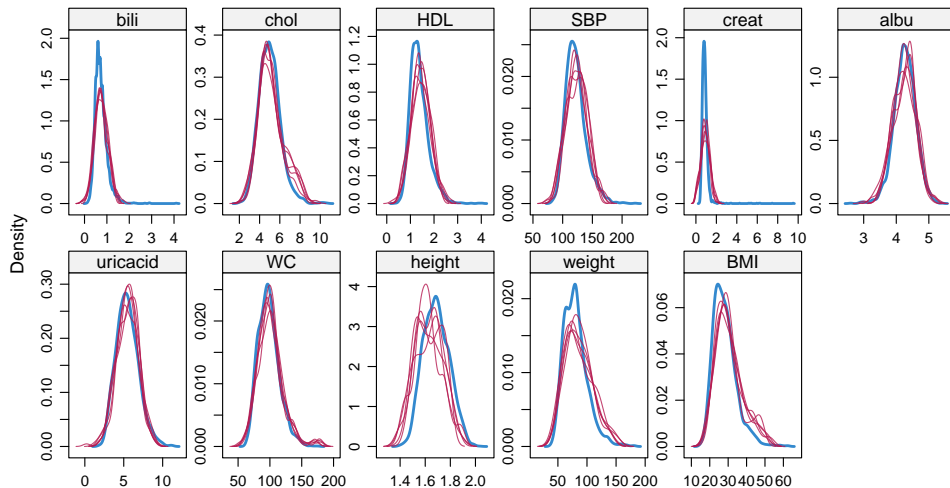
- `densityplot()` (for large datasets and variables with many NAs)
- `stripplot()` (for smaller datasets and/or variables with few NAs)
- `bwplot()`
- `xyplot()`

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

8. Convergence & Diagnostics

8.3. Diagnostics

```
densityplot(imp2)
```



8. Convergence & Diagnostics

8.3. Diagnostics

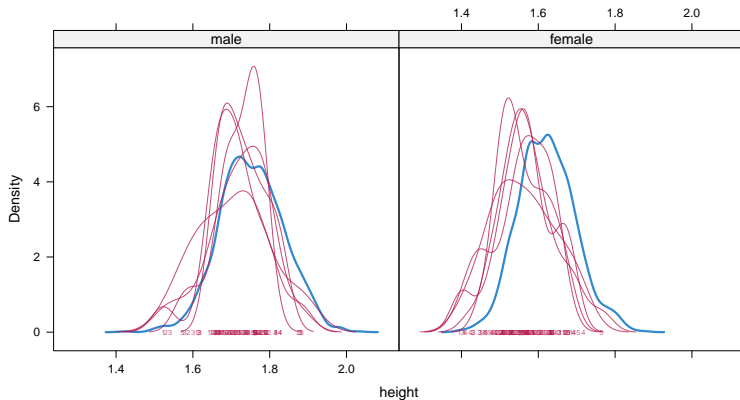
The `densityplot()` shows that the distribution of imputed values of `creat` is wider than the distribution of the observed values and that imputed values of `height` are smaller than the observed values.

8. Convergence & Diagnostics

8.3. Diagnostics

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, ~height|gender, plot.points = T)
```



8. Convergence & Diagnostics

8.3. Diagnostics

As an alternative, we might consider `race` to explain the differences

```
densityplot(imp2, ~height|race)
```

```
## Error in density.default(x = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, :  
need at least 2 points to select a bandwidth automatically
```


8. Convergence & Diagnostics

8.3. Diagnostics

As an alternative, we might consider `race` to explain the differences

```
densityplot(imp2, ~height|race)
```

```
## Error in density.default(x = c(NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, :  
need at least 2 points to select a bandwidth automatically
```

However, there are not enough missing values of `height` per categories of `race` to estimate densities.

```
with(NHANES, table(race = race, "height missing" = is.na(height)))
```

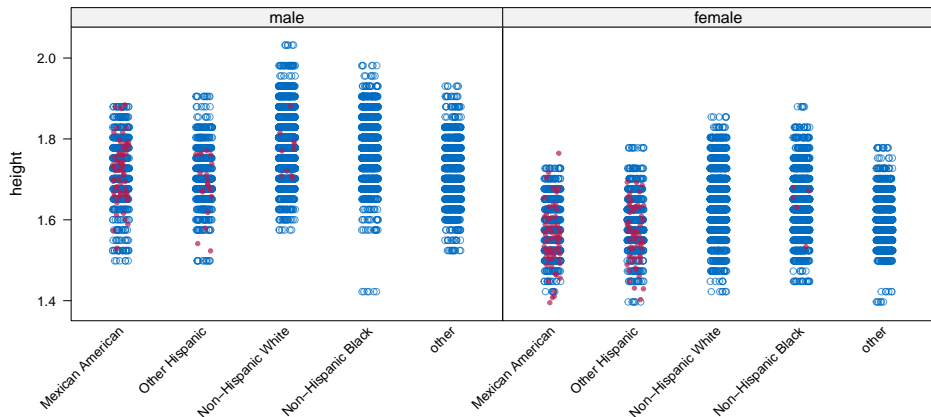
##		height missing	
##	race	FALSE	TRUE
##	Mexican American	233	26
##	Other Hispanic	252	16
##	Non-Hispanic White	884	2
##	Non-Hispanic Black	618	1
##	other	451	0

8. Convergence & Diagnostics

8.3. Diagnostics

In that case, a `stripplot()` may be better suited. Here we can also split the data for `gender` and `race`.

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

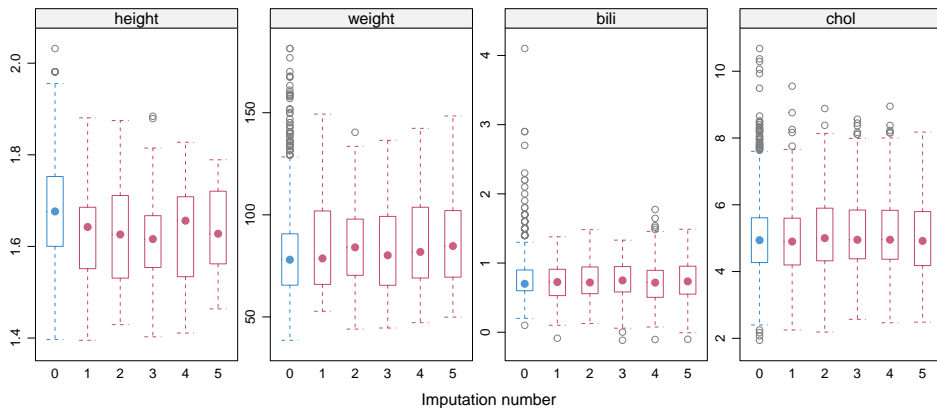


8. Convergence & Diagnostics

8.3. Diagnostics

Alternatively, observed and imputed data can be represented by box-and-whisker plots:

```
bwplot(imp2, height + weight + bili + chol ~.imp)
```

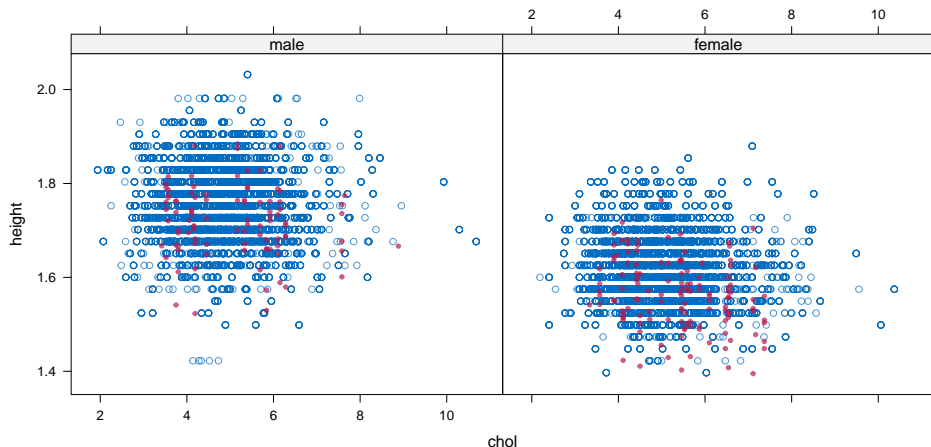


8. Convergence & Diagnostics

8.3. Diagnostics

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

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



8. Convergence & Diagnostics

8.3. Diagnostics

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

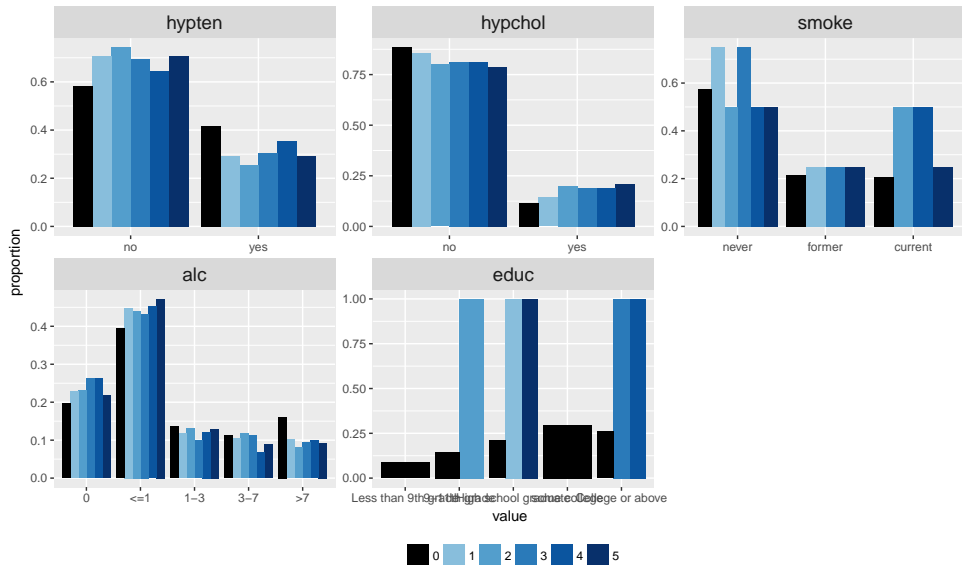
mice does not provide a function to do this, but we can write one ourselves, as for instance the function `probplot()`, for which the syntax can be found [here](#). The function can be loaded into R using:

```
devtools::source_gist("0d00375da460dd33839b98faeee2fdab",  
                      filename = "probplot.R")
```

8. Convergence & Diagnostics

8.3. Diagnostics

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



8. Convergence & Diagnostics

8.3. Diagnostics

`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.

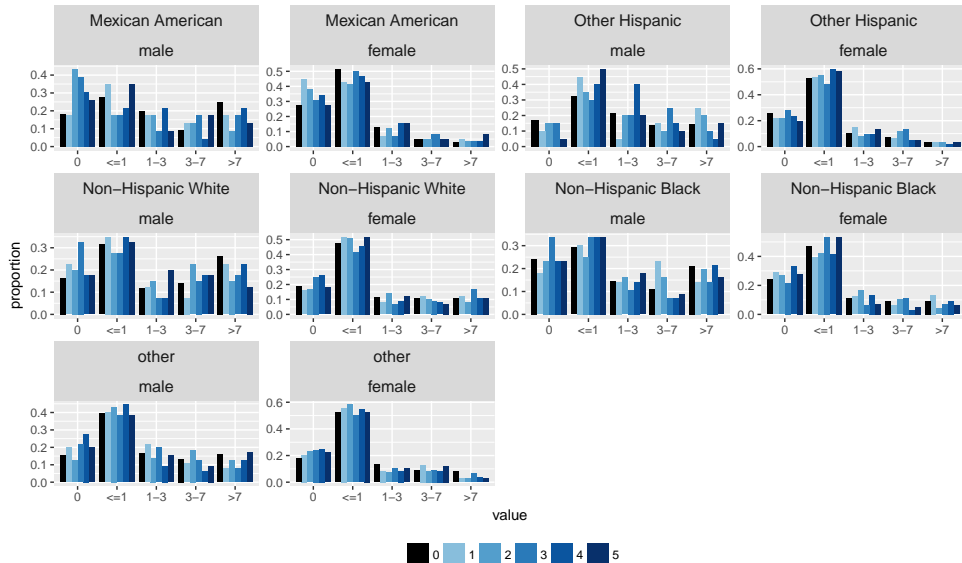
For `alc`, missing values are imputed by the lower consumption categories more often than we would expect from the observed data, `hyp ten` is less frequent and `hyp chol` a bit more frequent, in the imputed data compared to the observed.

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

8. Convergence & Diagnostics

8.3. Diagnostics

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



8. Convergence & Diagnostics

8.3. Diagnostics

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
```

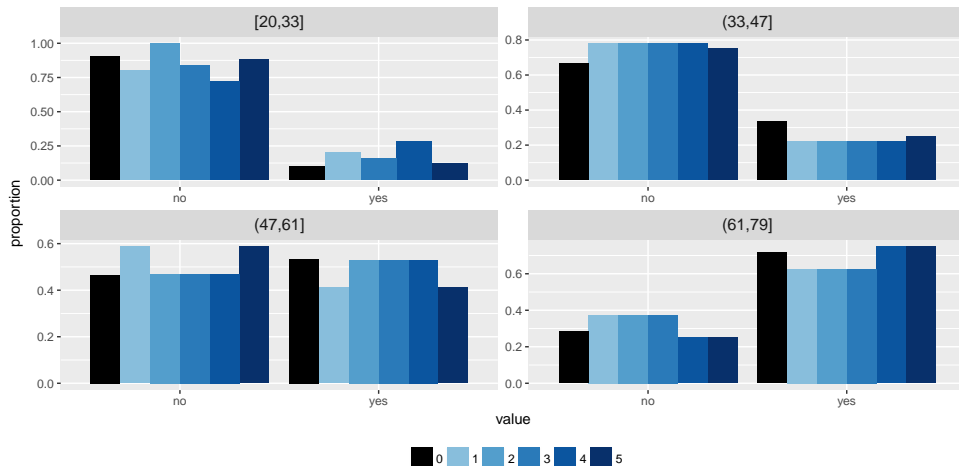
The table shows that the distribution of `age` in participants with missing `hypten` is very similar to the distribution of `age` in participants without `hypten`.

8. Convergence & Diagnostics

8.3. Diagnostics

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

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



9. Analyse & pool the imputed data

9.1. Analysing imputed data

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`.

9. Analyse & pool the imputed data

9.2. Pooling results

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

```
options(width = 90)
res1 <- summary(pool(mod1))
round(res1, 3)
```

##	est	se	t	df	Pr(> t)	lo 95	hi 95	nmis	fmi	lambda
## (Intercept)	-7.133	0.429	-16.616	2240.573	0.000	-7.975	-6.291	NA	0.013	0.012
## age	0.056	0.004	12.952	2468.059	0.000	0.048	0.065	0	0.001	0.001
## gender2	-0.422	0.128	-3.304	1749.968	0.001	-0.673	-0.172	NA	0.026	0.025
## hypchol2	-0.064	0.188	-0.342	403.591	0.732	-0.434	0.305	NA	0.095	0.090
## BMI	0.106	0.009	11.576	2265.730	0.000	0.088	0.123	73	0.012	0.011
## smoke2	0.129	0.144	0.896	2432.312	0.370	-0.153	0.411	NA	0.005	0.004
## smoke3	0.080	0.166	0.479	1953.715	0.632	-0.246	0.405	NA	0.021	0.020
## alc2	-0.277	0.150	-1.845	475.882	0.066	-0.573	0.018	NA	0.085	0.082
## alc3	-0.570	0.220	-2.585	1192.205	0.010	-1.002	-0.137	NA	0.042	0.041
## alc4	-0.466	0.246	-1.894	154.639	0.060	-0.952	0.020	NA	0.165	0.155
## alc5	-0.741	0.220	-3.375	747.163	0.001	-1.172	-0.310	NA	0.063	0.060

9. Analyse & pool the imputed data

9.2. Pooling results

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

Generally, it works for models for which the functions `coef()` and `vcov()` can extract the (fixed effects) **coefficients and variance-covariance matrix** of these coefficients.

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

9. Analyse & pool the imputed data

9.3. Functions for pooled results

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 0.3243655 0.2940335 0.3547556 0.006885114
```

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

9. Analyse & pool the imputed data

9.3. Functions for pooled results

The function `pool.compare()` allows to compare **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,] 0.6577086
```

9. Analyse & pool the imputed data

9.3. Functions for pooled results

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.

9. Analyse & pool the imputed data

9.3. Functions for pooled results

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.

10. Additional functions in `mice()`

10.1. Extract & export imputed data

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

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

- `x`: 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"`)

10. Additional functions in mice()

10.1. Extract & export imputed data

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()`.

10. Additional functions in `mice()`

10.2. Combining `mids` objects

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
```

10. Additional functions in mice()

10.3. Adding variables to mids objects

The function `cbind.mids()` allows to **add columns** to a `mids` object. The extra columns can either be a `data.frame`, `matrix`, `vector` or `factor` or another `mids` object.

For example data columns that should be part of the imputed data for completeness, but are not needed in the imputation.

```
extravar <- rnorm(nrow(NHANES))  
impextra <- mice::cbind.mids(x = imp2, extravar = extravar)
```

Note: `cbind()` just adds columns to the data, you need to make sure they are **sorted correctly** so that the rows of the new data are from the same subjects as the corresponding rows in the impute data.

11. Multiple Imputation in SPSS

11.1. Where to get help

A walk-through how to do multiple imputation in SPSS can be found

- **for older versions of SPSS**

- > Help
 - > Case Studies
 - > Missing Values Option
 - > Multiple Imputation
 - > Using Multiple Imputation to Complete and Analyze a Dataset

- **for newer versions online**

https://www.ibm.com/support/knowledgecenter/en/SSLVMB_24.0.0/spss/tutorials/mi_table.html

11. Multiple Imputation in SPSS

11.1. Where to get help

A walk-through how to do multiple imputation in SPSS can be found

- **for older versions of SPSS**

- > Help
 - > Case Studies
 - > Missing Values Option
 - > Multiple Imputation
 - > Using Multiple Imputation to Complete and Analyze a Dataset

- **for newer versions online**

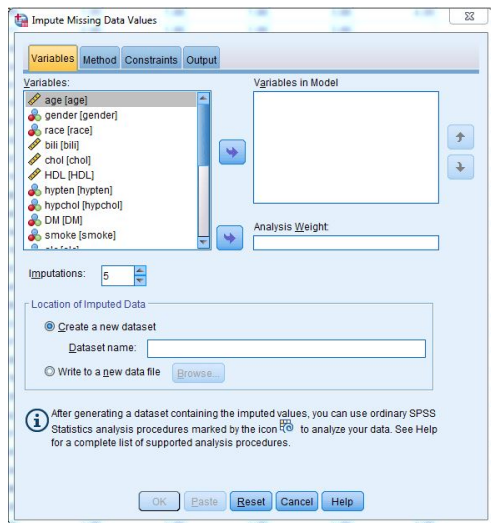
https://www.ibm.com/support/knowledgecenter/en/SSLVMB_24.0.0/spss/tutorials/mi_table.html

The **procedure** itself is located in the menu

- > Analyze
 - > Multiple Imputation
 - > Impute Missing Data Values

11. Multiple Imputation in SPSS

11.2. Multiple Imputation Features

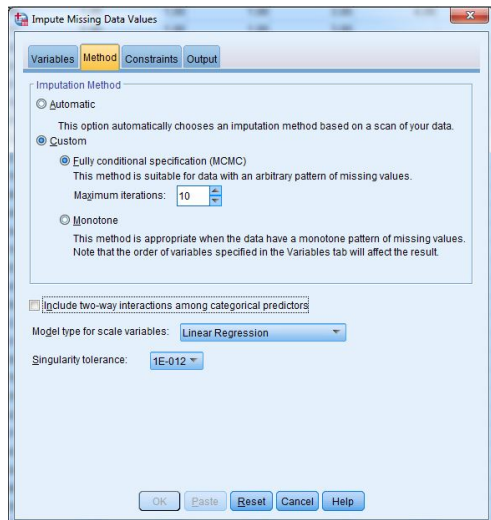


SPSS lets you

- specify number of imputations

11. Multiple Imputation in SPSS

11.2. Multiple Imputation Features

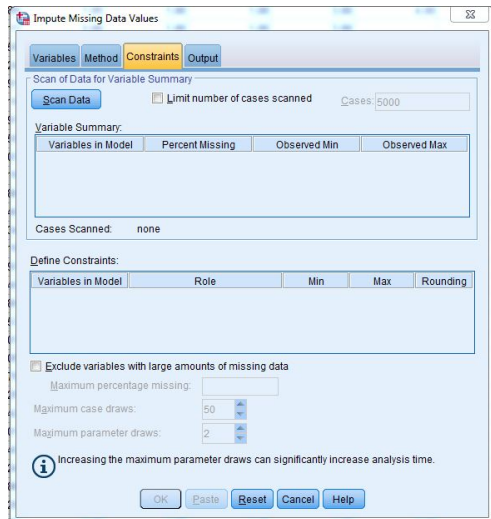


SPSS lets you

- specify number of imputations
- specify number of iterations
- include interactions
- choose between lin. regression and pmm for continuous variables

11. Multiple Imputation in SPSS

11.2. Multiple Imputation Features

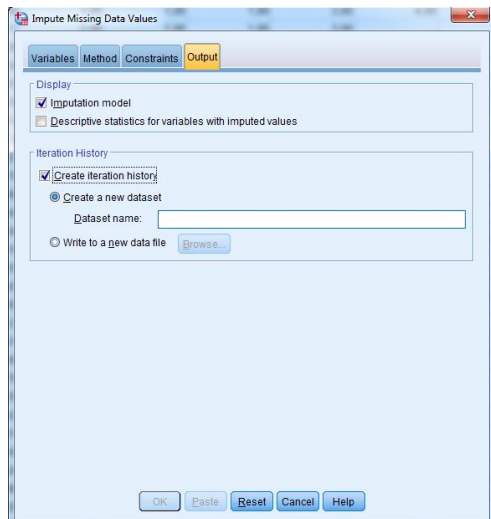


SPSS lets you

- specify number of imputations
- specify number of iterations
- include interactions
- chose between lin. regression and pmm for continuous variables
- restrict variables to certain values
- select which variables to impute
- select which variables are used as predictors

11. Multiple Imputation in SPSS

11.2. Multiple Imputation Features



SPSS lets you

- specify number of imputations
- specify number of iterations
- include interactions
- chose between lin. regression and pmm for continuous variables
- restrict variables to certain values
- select which variables to impute
- select which variables are used as predictors
- save the iteration history

11. Multiple Imputation in SPSS

11.2. Multiple Imputation Features

SPSS does not let you

- select between linear regression imputation and pmm **per** variable (only jointly for all variables)
- uses **only one donor** in predictive mean matching
- uses logistic regression for any categorical variable
- chose per imputation model which variables should be used as predictors
- re-calculate variables during the iterations
- ...

11. Multiple Imputation in SPSS

11.2. Multiple Imputation Features

In SPSS the **list of models that can be pooled** is available in the help under

- > Help
 - > Missing Values Option
 - > Multiple Imputation
 - > Analyzing Multiple Imputation Data

https://www.ibm.com/support/knowledgecenter/en/SSLVMB_24.0.0/spss/mva/mi_analysis.html

Practical

To practice all that we have seen above, go to

https://emcbiostatistics.shinyapps.io/MICourse_MICE

or download the instructions and data for the practical from Canvas
(Files > Principal documents > Multiple Imputation > Practical MICE).

Part III

When MICE might fail

Outline of Part III

12. Settings where MICE may have problems

- 12.1 Example: Quadratic effect

- 12.2 Example: Interaction effect

- 12.3 Example: Longitudinal outcome

- 12.4 Example: Survival data

13. Requirements for MICE to work (well)

- 13.1 Joint and conditional distributions

- 13.2 Some conditions and definitions

- 13.3 Why imputation with MICE can go wrong

14. Alternatives to MICE

- 14.1 Joint model imputation

- 14.2 Multivariate Normal Model

- 14.3 Sequential Factorization

Outline of Part III (cont.)

15. Imputation with non-linear functional forms

- 15.1 R package mice
- 15.2 R package JointAI
- 15.3 R package smcfcs
- 15.4 R package jomo
- 15.5 Comparison of results

16. Imputation of longitudinal data

- 16.1 R package mice
- 16.2 R package JointAI
- 16.3 R package jomo
- 16.4 Comparison of results

Outline of Part III (cont.)

17. Imputation of survival data

- 17.1 Results from literature

- 17.2 R package mice

- 17.3 R package smcfcs

- 17.4 R package jomo

- 17.5 Comparison of results

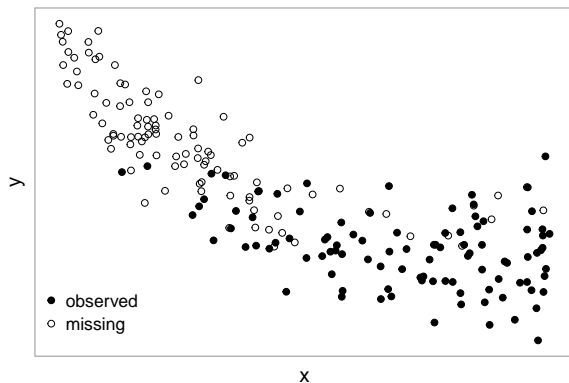
12. Settings where MICE may have problems

12.1. Example: Quadratic effect

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

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

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



The original data show a curved pattern.

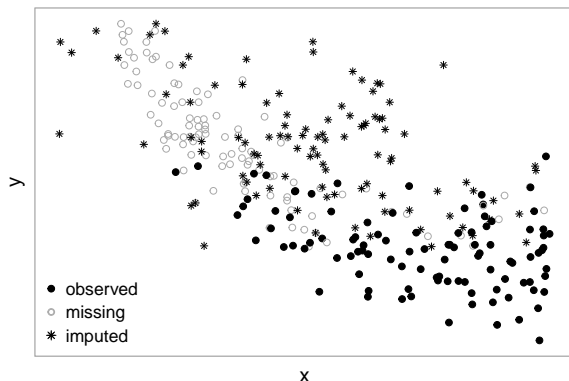
12. Settings where MICE may have problems

12.1. Example: Quadratic effect

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

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

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

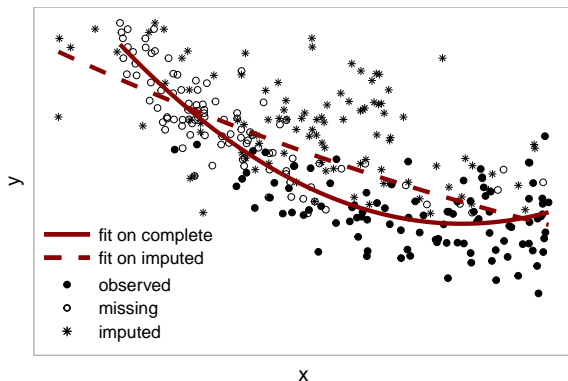


The imputed values **distort the curved pattern** of the original data.

12. Settings where MICE may have problems

12.1. Example: Quadratic effect

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]

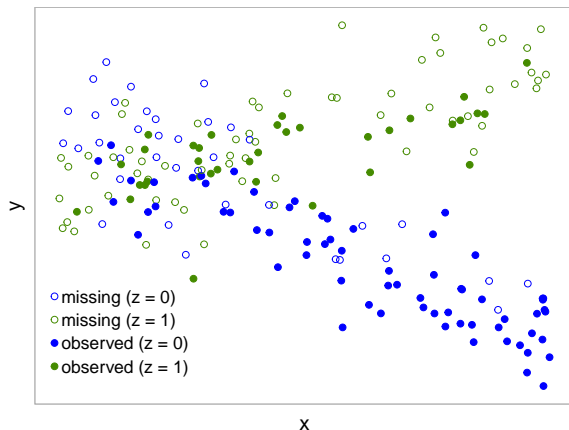
12. Settings where MICE may have problems

12.2. Example: Interaction effect

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

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

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



The original data shows a “<” shaped pattern.

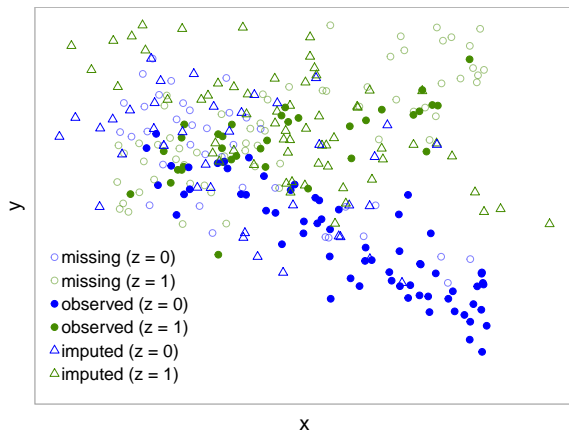
12. Settings where MICE may have problems

12.2. Example: Interaction effect

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

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

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

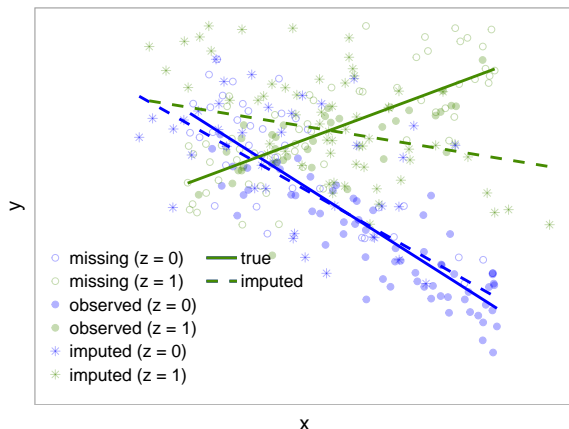


The “<” shaped pattern of the true data is **distorted by the imputed values**.

12. Settings where MICE may have problems

12.2. Example: Interaction effect

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

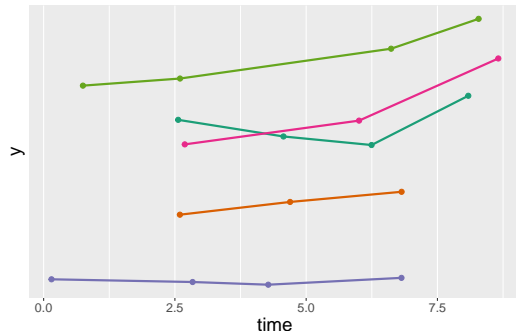


	β	95% CI
Original		
Intercept	-0.96	[-1.00, -0.92]
x	-0.59	[-0.65, -0.53]
z	0.5	[0.45, 0.56]
x:z	0.94	[0.85, 1.03]
Imputed		
Intercept	-0.96	[-1.01, -0.91]
x	-0.52	[-0.61, -0.44]
z	0.46	[0.39, 0.54]
x:z	0.37	[0.24, 0.51]

12. Settings where MICE may have problems

12.3. Example: Longitudinal outcome

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	✓	✓	NA	✓	✓	2.56
5	✓	✓	NA	✓	✓	4.57
5	✓	✓	NA	✓	✓	6.25
5	✓	✓	NA	✓	✓	8.09
6	✓	✓	NA	NA	✓	2.60
6	✓	✓	NA	NA	✓	4.69
6	✓	✓	NA	NA	✓	6.82
8	✓	✓	✓	✓	NA	2.69
8	✓	✓	✓	✓	NA	6.01
8	✓	✓	✓	✓	NA	8.66
18	✓	✓	NA	✓	✓	0.75
18	✓	✓	NA	✓	✓	2.60
18	✓	✓	NA	✓	✓	6.62
18	✓	✓	NA	✓	✓	8.28
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Here, x_1, \dots, x_4 are baseline covariates, i.e., not measured repeatedly.

12. Settings where MICE may have problems

12.3. Example: Longitudinal outcome

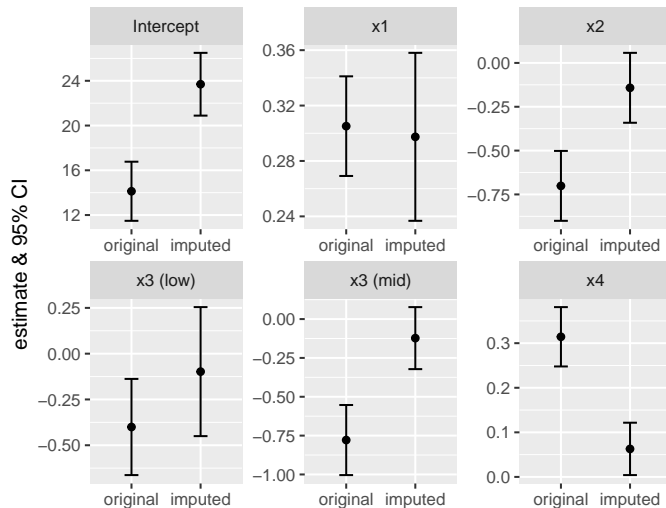
If we use MICE in the data in this (long) format, each row would be regarded as independent, which may cause bias and **inconsistent imputations**.

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	✓	✓	boy	✓	✓	2.56
5	✓	✓	girl	✓	✓	4.57
5	✓	✓	girl	✓	✓	6.25
5	✓	✓	girl	✓	✓	8.09
6	✓	✓	girl	high	✓	2.60
6	✓	✓	boy	mid	✓	4.69
6	✓	✓	girl	high	✓	6.82
8	✓	✓	✓	✓	38.27	2.69
8	✓	✓	✓	✓	38.45	6.01
8	✓	✓	✓	✓	40.71	8.66
18	✓	✓	boy	✓	✓	0.75
18	✓	✓	boy	✓	✓	2.60
18	✓	✓	boy	✓	✓	6.62
18	✓	✓	boy	✓	✓	8.28
⋮	⋮	⋮	⋮	⋮	⋮	⋮

12. Settings where MICE may have problems

12.3. Example: Longitudinal outcome

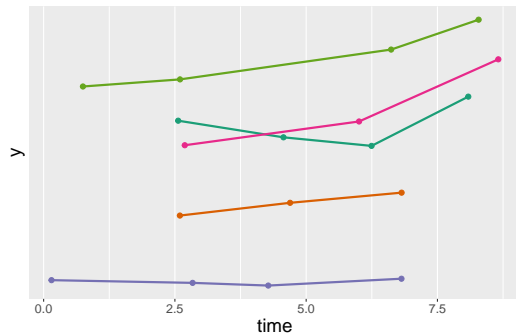


Estimates can be severely biased.

12. Settings where MICE may have problems

12.3. Example: Longitudinal outcome

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

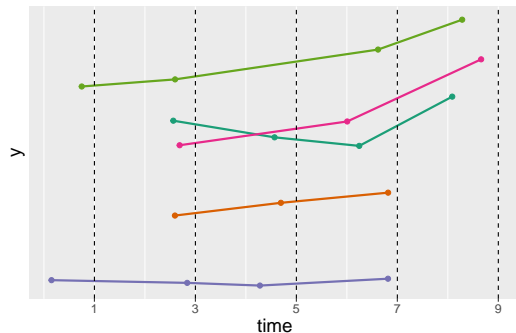


ID	y	x ₁	x ₂	x ₃	x ₄	time
5	✓	✓	NA	✓	✓	2.56
5	✓	✓	NA	✓	✓	4.57
5	✓	✓	NA	✓	✓	6.25
5	✓	✓	NA	✓	✓	8.09
6	✓	✓	NA	NA	✓	2.60
6	✓	✓	NA	NA	✓	4.69
6	✓	✓	NA	NA	✓	6.82
8	✓	✓	✓	✓	NA	2.69
8	✓	✓	✓	✓	NA	6.01
8	✓	✓	✓	✓	NA	8.66
18	✓	✓	NA	✓	✓	0.75
18	✓	✓	NA	✓	✓	2.60
18	✓	✓	NA	✓	✓	6.62
18	✓	✓	NA	✓	✓	8.28
⋮	⋮	⋮	⋮	⋮	⋮	⋮

12. Settings where MICE may have problems

12.3. Example: Longitudinal outcome

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



ID	y	x ₁	x ₂	x ₃	x ₄	time
5	✓	✓	NA	✓	✓	2.56
5	✓	✓	NA	✓	✓	4.57
5	✓	✓	NA	✓	✓	6.25
5	✓	✓	NA	✓	✓	8.09
6	✓	✓	NA	NA	✓	2.60
6	✓	✓	NA	NA	✓	4.69
6	✓	✓	NA	NA	✓	6.82
8	✓	✓	✓	✓	NA	2.69
8	✓	✓	✓	✓	NA	6.01
8	✓	✓	✓	✓	NA	8.66
18	✓	✓	NA	✓	✓	0.75
18	✓	✓	NA	✓	✓	2.60
18	✓	✓	NA	✓	✓	6.62
18	✓	✓	NA	✓	✓	8.28
⋮	⋮	⋮	⋮	⋮	⋮	⋮

12. Settings where MICE may have problems

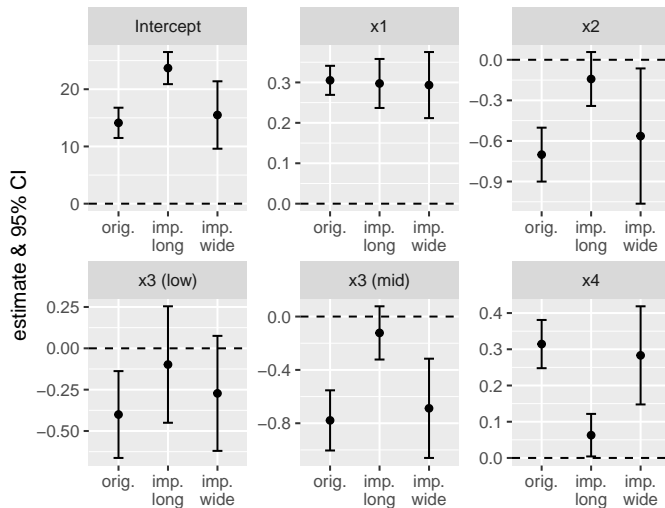
12.3. Example: Longitudinal outcome

id	y.1	y.3	y.5	y.7	y.9	time.1	time.3	time.5	time.7	time.9	...
5	NA	35.19	34.94	34.81	35.56	NA	2.56	4.57	6.25	8.09	...
6	NA	33.74	33.94	34.09	NA	NA	2.6	4.69	6.82	NA	...
8	NA	34.82	NA	35.18	36.13	NA	2.69	NA	6.01	8.66	...
18	35.71	35.82	NA	36.28	36.73	0.75	2.6	NA	6.62	8.28	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

In this **wide format data** frame, missing values in 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 valid when outcome measurements are M(C)AR).

12. Settings where MICE may have problems

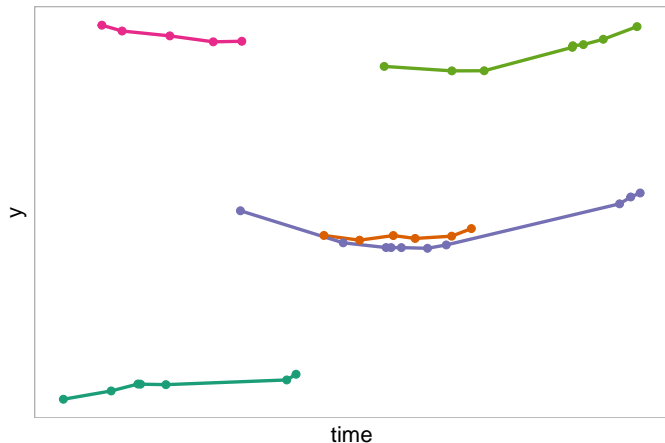
12.3. Example: Longitudinal outcome



Better, but very large confidence intervals.

12. Settings where MICE may have problems

12.3. Example: Longitudinal outcome

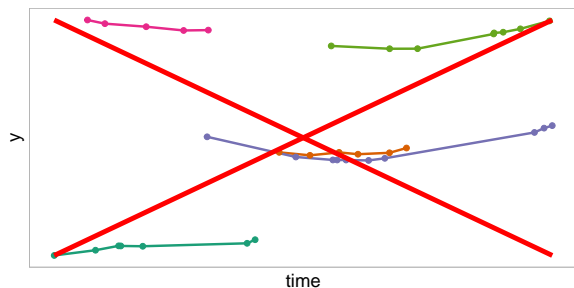


When the data is very **unbalanced**, i.e., there are no clear cut-offs in time, transformation to wide format is not possible.

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

12. Settings where MICE may have problems

12.3. Example: Longitudinal outcome

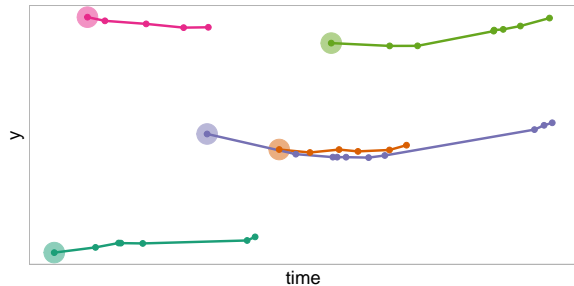
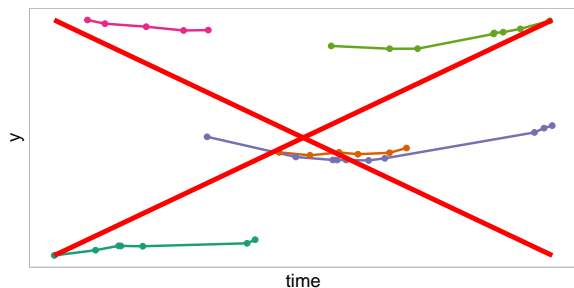


Naive approaches that are sometimes used are to

- **ignore the outcome** in the imputation

12. Settings where MICE may have problems

12.3. Example: Longitudinal outcome

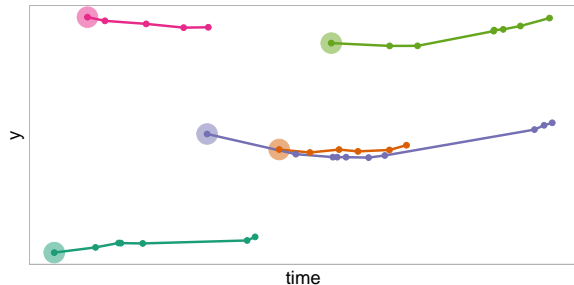
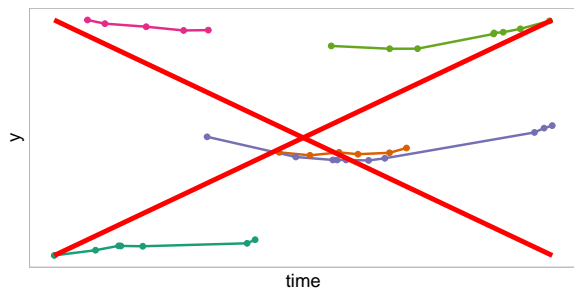


Naive approaches that are sometimes used are to

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

12. Settings where MICE may have problems

12.3. Example: Longitudinal outcome



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.

12. Settings where MICE may have problems

12.4. Example: Survival data

In **survival analysis**, the aim is to estimate the effect of covariates on the **time until an event** of interest happens.

12. Settings where MICE may have problems

12.4. Example: Survival data

In **survival analysis**, the aim is to estimate the effect of covariates on the **time until an event** of interest happens.

In the commonly used method: **Cox proportional hazards model**

$$h(t) = h_0(t) \exp(x\beta_x + z\beta_z),$$

- $h(t)$: hazard = the instantaneous risk of an event at time t , given that the event has not occurred until time t
- $h_0(t)$: unspecified baseline hazard
- x and z : **incomplete** and **complete** covariates, respectively

12. Settings where MICE may have problems

12.4. Example: Survival data

In **survival analysis**, the aim is to estimate the effect of covariates on the **time until an event** of interest happens.

In the commonly used method: **Cox proportional hazards model**

$$h(t) = h_0(t) \exp(x\beta_x + z\beta_z),$$

- $h(t)$: hazard = the instantaneous risk of an event at time t , given that the event has not occurred until time t
- $h_0(t)$: unspecified baseline hazard
- x and z : **incomplete** and **complete** covariates, respectively

Survival outcomes are usually represented by the **observed event time** T and the **event indicator** D ($D = 1$: event, $D = 0$: censored).

12. Settings where MICE may have problems

12.4. Example: Survival data

Naive use of MICE treats the columns in the data set containing T and D just like any other variable, and the resulting imputation model for X would have the form

$$p(x \mid T, D, \mathbf{z}) = \theta_0 + \theta_1 T + \theta_2 D + \theta_3 z + \dots$$

12. Settings where MICE may have problems

12.4. Example: Survival data

Naive use of MICE treats the columns in the data set containing T and D just like any other variable, and the resulting imputation model for X would have the form

$$p(x \mid T, D, z) = \theta_0 + \theta_1 T + \theta_2 D + \theta_3 z + \dots$$

The correct conditional distribution of x given the other variables is, however,

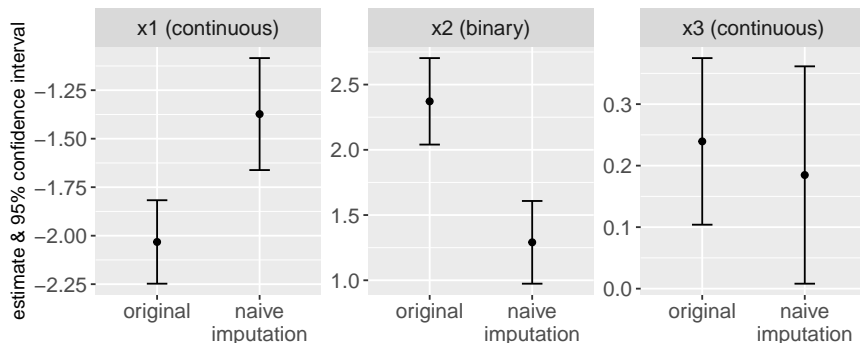
$$\log p(x \mid T, D, z) = \log p(x \mid z) + D(\beta_x x + \beta_z z) - \\ H_0(T) \exp(\beta_x x + \beta_z z) + \text{const.},$$

where $H_0(T)$ is the cumulative baseline hazard.[20]

12. Settings where MICE may have problems

12.4. Example: Survival data

Using the naively assumed imputation model can lead to **severe bias**:



(Results from MICE imputation with two incomplete normal and one incomplete binary covariate.)

13. Requirements for MICE to work (well)

13.1. Joint and conditional distributions

Recall: 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.

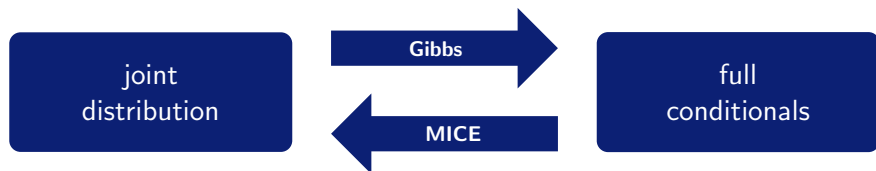


13. Requirements for MICE to work (well)

13.1. Joint and conditional distributions

Recall: 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.

13. Requirements for MICE to work (well)

13.1. Joint and conditional distributions

The **uncertainty about whether a joint distribution exists** for the specified set of imputation models is often considered to be mainly a theoretical problem.

In practice, violations only have little impact on results in many applications.

However, as we have seen in the examples on the previous slides, there are **settings where the direct specification** of the full conditionals/imputation models **may lead to problems**, causing biased results.

13. Requirements for MICE to work (well)

13.2. Some conditions and definitions

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.

13. Requirements for MICE to work (well)

13.2. Some conditions and definitions

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)

13. Requirements for MICE to work (well)

13.3. Why imputation with MICE can go wrong

What went wrong in our previous examples?

When incomplete variables have **non-linear associations** with the outcome, or with each other, the requirement(s) of **compatibility and/or congeniality are violated**.

Omission, or inadequate inclusion, of the outcome may result in **MNAR** missing mechanisms. The same is the case when other relevant predictor variables are not used as predictor variables in the imputation.

Furthermore, **omission of variables** may lead to **mis-specified models**, however, models may also be mis-specified when all relevant covariates are included, but **distributional assumptions** or the specified **form of associations** are incorrect.

14. Alternatives to MICE

14.1. Joint model imputation

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.

14. Alternatives to MICE

14.1. Joint model imputation

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:

Especially in settings with several **variables of mixed type**, the joint distribution is usually not of any known form:

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

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

14. Alternatives to MICE

14.1. Joint model imputation

Approach 1: Multivariate Normal Model

Approximate the joint distribution by a known multivariate distribution (usually the normal distribution; this is the approach mentioned in Part I on slide 15)

Approach 2: Sequential Factorization

Factorize the joint distribution into a (sequence of) conditional and a marginal distributions

14. Alternatives to MICE

14.2. Multivariate Normal Model

Assumption:

The outcome and incomplete variables follow a **joint multivariate normal distribution**, conditional on the completely observed covariates \mathbf{X}_c , parameters $\boldsymbol{\theta}$ and, possibly, random effects, \mathbf{b} :

$$p(\mathbf{y}, \mathbf{x}_1, \dots, \mathbf{x}_p \mid \mathbf{X}_c, \boldsymbol{\theta}, \mathbf{b}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

14. Alternatives to MICE

14.2. Multivariate Normal Model

Assumption:

The outcome and incomplete variables follow a **joint multivariate normal distribution**, conditional on the completely observed covariates \mathbf{X}_c , parameters θ and, possibly, random effects, \mathbf{b} :

$$p(\mathbf{y}, \mathbf{x}_1, \dots, \mathbf{x}_p \mid \mathbf{X}_c, \theta, \mathbf{b}) \sim N(\mu, \Sigma)$$

How do we get that multivariate normal distribution?

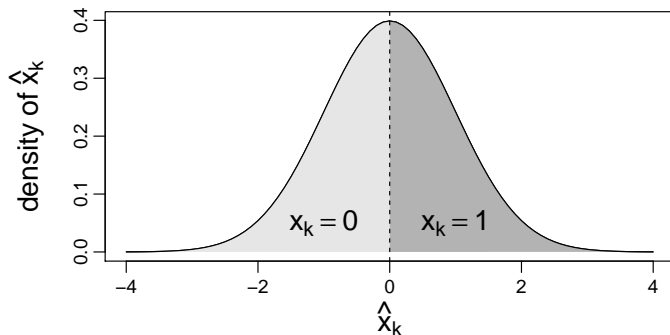
1. Assume **all** incomplete variables and the outcome are **(latent) normal**.
2. Specify linear (mixed) **models based on observed covariates**.
3. **Connect** using multivariate normal for **random effects & error terms**.

14. Alternatives to MICE

14.2. Multivariate Normal Model

1. Latent normal assumption:

e.g.: \mathbf{x}_k binary \rightarrow latent $\hat{\mathbf{x}}_k$ is standard normal:
$$\begin{cases} \mathbf{x}_k = 1 & \text{if } \hat{\mathbf{x}}_k \geq 0 \\ \mathbf{x}_k = 0 & \text{if } \hat{\mathbf{x}}_k < 0 \end{cases}$$



14. Alternatives to MICE

14.2. Multivariate Normal Model

2. Specify models:

$$\mathbf{y} = \mathbf{X}_c \boldsymbol{\beta}_y + \mathbf{Z}_y \mathbf{b}_y + \boldsymbol{\varepsilon}_y$$

$$\mathbf{w} = \mathbf{X}_c \boldsymbol{\beta}_w + \mathbf{Z}_w \mathbf{b}_w + \boldsymbol{\varepsilon}_w$$

$$\begin{aligned} \hat{x}_1 &= \mathbf{X}_c \boldsymbol{\beta}_{x_1} + \boldsymbol{\varepsilon}_{x_1} \\ &\vdots \qquad \qquad \qquad \vdots \end{aligned}$$

$$\hat{x}_p = \mathbf{X}_c \boldsymbol{\beta}_{x_p} + \boldsymbol{\varepsilon}_{x_p}$$

14. Alternatives to MICE

14.2. Multivariate Normal Model

2. Specify models / 3. Connect random effects & error terms:

$$\begin{aligned} y &= \mathbf{X}_c \beta_y + \mathbf{Z}_y \mathbf{b}_y + \epsilon_y \\ w &= \mathbf{X}_c \beta_w + \mathbf{Z}_w \mathbf{b}_w + \epsilon_w \\ \hat{x}_1 &= \mathbf{X}_c \beta_{x_1} + \epsilon_{x_1} \\ &\vdots \\ \hat{x}_p &= \mathbf{X}_c \beta_{x_p} + \epsilon_{x_p} \end{aligned}$$

multivariate normal (optional, but suggested)

multivariate normal

14. Alternatives to MICE

14.2. Multivariate Normal Model

Advantages:

- Easy to specify
- Relatively easy to implement
- Relatively easy to sample from
- Works for longitudinal outcomes

Disadvantages:

- Assumes linear associations

Imputation with **non-linear associations** or **survival data** is possible with **extensions** of the multivariate normal approach.

14. Alternatives to MICE

14.3. Sequential Factorization

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

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

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

14. Alternatives to MICE

14.3. Sequential Factorization

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

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

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

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

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

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

14. Alternatives to MICE

14.3. Sequential Factorization

That the analysis model is part of the specification of the joint distribution has several 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.

14. Alternatives to MICE

14.3. Sequential Factorization

That the analysis model is part of the specification of the joint distribution has several 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

14. Alternatives to MICE

14.3. Sequential Factorization

That the analysis model is part of the specification of the joint distribution has several 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**

14. Alternatives to MICE

14.3. Sequential Factorization

That the analysis model is part of the specification of the joint distribution has several 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.

14. Alternatives to MICE

14.3. Sequential Factorization

Advantages:

- **flexible** with regards to outcome type
- univariate conditional distributions of incomplete covariates can be chosen according to **type of variable**
- **non-linear associations** and interactions can be taken into account
- assures **congeniality and compatible imputation models**

Disadvantages:

- separate models need to be specified per incomplete variable: **takes more time and consideration**
- the joint distribution is of unknown form and sampling may be more **computationally intensive**

15. Imputation with non-linear functional forms

In the following we will not only consider the R package **mice**, but also three additional packages, **JointAI**, **smcfcs** and **jomo**, that provide alternatives to **mice**.

These three packages use **Bayesian methodology** to impute values, but once imputed datasets are obtained, standard complete data methods can be used.

jomo and **smcfcs** perform **multiple imputation** and create imputed datasets that can then be analysed the same way data imputed by **mice** would be analysed.

JointAI works **fully Bayesian** and performs the analysis and imputation simultaneously, so that the results from the analysis model of interest are obtained directly.

15. Imputation with non-linear functional forms

15.1. R package mice

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 `norm`,
- Just Another Variable approach can reduce bias in interactions,
- `quadratic` can help to impute variables with quadratic association.

15. Imputation with non-linear functional forms

15.1. R package mice

In the **Just Another Variable (JAV)** approach the non-linear form (or interaction term) is calculated in the incomplete data, added as a column to the dataset and imputed as if it was just another variable.

`quadratic` provides imputation of covariates that have a quadratic association with the outcome, using the “polynomial combination” method.[17, pp. 139–141], [19].

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.

15. Imputation with non-linear functional forms

15.1. R package mice

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

15. Imputation with non-linear functional forms

15.1. R package mice

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)
```

15. Imputation with non-linear functional forms

15.1. R package mice

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)
```

15. Imputation with non-linear functional forms

15.1. R package mice

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)")

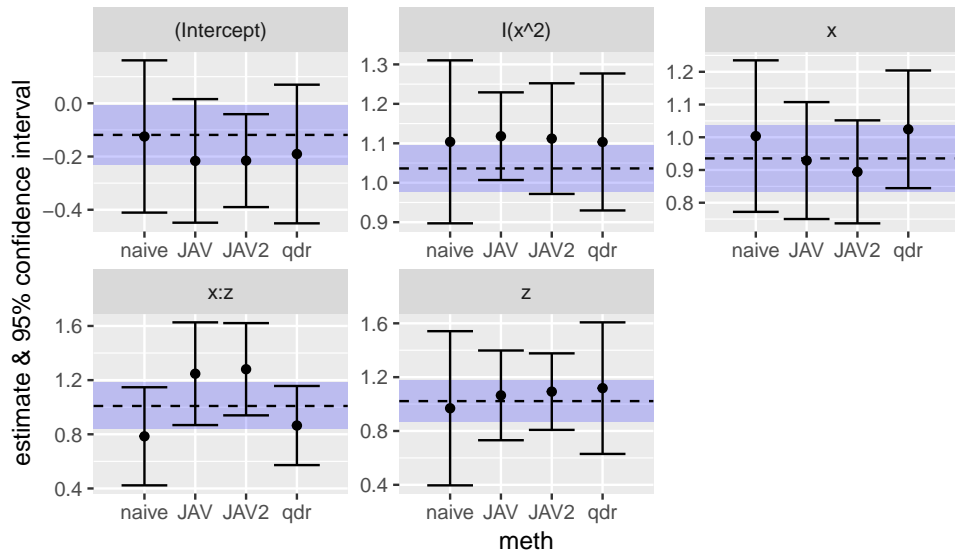
# adapt the predictor matrix
predqdr <- impJAV$pred
predqdr[, "xx"] <- 0

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

Note: there were warning messages about numerical issues for this approach.

15. Imputation with non-linear functional forms

15.1. R package mice



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

15. Imputation with non-linear functional forms

15.2. R package JointAI

The package **JointAI** uses the **sequential factorization approach** to perform simultaneous analysis and imputation.[4, 3]

JointAI (version 0.1.0) can handle

- linear regression
- generalized linear regression
- linear mixed 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/>.

15. Imputation with non-linear functional forms

15.2. R package JointAI

JointAI can be installed from CRAN

```
install.packages("JointAI")
```

The development version (containing bug fixes and other improvements) can be installed from [GitHub](#)

```
install.packages("devtools")  
devtools::install_github(repo = "JointAI", username = "NErler")
```

A detailed explanation of the functionality is given in the help files of the package, and a vignette with an in-depth example analysis will be available soon.

15. Imputation with non-linear functional forms

15.2. R package JointAI

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)
```

15. Imputation with non-linear functional forms

15.2. R package JointAI

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)
```

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

```
res_JointAI_nonlin <- summary(JointAI_nonlin)
```

15. Imputation with non-linear functional forms

15.3. R package smcfcs

The package **smcfcs** performs multiple imputation using *substantive model compatible fully conditional specification*, a **hybrid approach between FCS and sequential factorization**.^[1]

smcfcs (version 1.3.0) can handle

- linear regression,
- logistic regression,
- poisson regression,
- Cox proportional hazard models, and
- competing risk survival models,

while ensuring compatibility between analysis model and imputation models.

For more information see the help files and the [vignette](#).

15. Imputation with non-linear functional forms

15.3. R package smcfcs

The syntax to impute the data in the current example using the package **smcfcs** is:

```
library(smcfcs)
smcfcs_nonlin <- smcfcs(originaldata = DF_nonlin, smtype = "lm",
                        smformula = "y~x*z + I(x^2)",
                        method = c("", "norm", ""),
                        rjlimit = 3000, numit = 20)
```

The convergence of the procedure should be checked, for example with the following syntax:

```
par(mfrow = c(2,3), mar = c(2, 2, 0.5, 0.5), mgp = c(2, 0.6, 0))
for(i in 1:dim(smcfcs_nonlin$smCoefIter)[2]) {
  matplot(t(smcfcs_nonlin$smCoefIter[, i, ]), type = 'l', ylab = '')
}
```

15. Imputation with non-linear functional forms

15.3. R package smcfcs

To be able to use the convenient pooling function from the **mice** package we first need to convert the imputed data (which is a list) to a `mids` object.

This can be done with the function `datalist2mids()` from the **miceadds** package.

```
library(miceadds)
impobj_smcfcs_nonlin <- datalist2mids(smcfcs_nonlin$impDatasets)
```

The `mids` object can then be pooled and summarized as we have seen before with `mids` objects created by `mice()`.

```
models_smcfcs_nonlin <- with(impobj_smcfcs_nonlin, lm(y ~ x*z + I(x^2)))
res_smcfcs_nonlin <- summary(pool(models_smcfcs_nonlin))
```

15. Imputation with non-linear functional forms

15.4. R package jomo

The package **jomo** performs **joint model imputation** using the multivariate normal approach, with **extensions to assure compatibility** between analysis and imputation models.[2]

jomo (version 2.6-2) can handle

- linear regression,
- generalized linear regression,
- linear mixed models,
- generalized linear mixed models, and
- Cox proportional hazards models.

15. Imputation with non-linear functional forms

15.4. R package jomo

Using **jomo** we can impute the data in the current example as follows:

```
library(jomo)
jomo_nonlin <- jomo.lm(y ~ x*z + I(x^2), data = DF_nonlin)
```


15. Imputation with non-linear functional forms

15.4. R package jomo

Using **jomo** we can impute the data in the current example as follows:

```
library(jomo)
jomo_nonlin <- jomo.lm(y ~ x*z + I(x^2), data = DF_nonlin)
```

To check the convergence of the model, the corresponding function with ending **.MCMCchain()** as to be used.

```
jomo_nonlinMCMC <- jomo.lm.MCMCchain(y ~ x*z + I(x^2), data = DF_nonlin)

par(mfcol = c(2, 3), mar = c(3, 2.5, 0.5, 0.5), mgp = c(2, 0.6, 0))
apply(jomo_nonlinMCMC$collectbeta[1, ], 1, plot, type = "l",
      xlab = 'iteration', ylab = '')
for (k in 1:dim(jomo_nonlinMCMC$collectomega)[1]) {
  apply(jomo_nonlinMCMC$collectomega[k, ], 1, plot, type = "l",
        xlab = 'iteration', ylab = '')
}

apply(jomo_nonlinMCMC$collectbetaY[1, ], 1, plot, type = "l",
      xlab = 'iteration', ylab = '')
plot(jomo_nonlinMCMC$collectvarY, type = 'l')
```

15. Imputation with non-linear functional forms

15.4. R package jomo

Again, we need to convert the output to a `mids` object using `datalist2mids()`. However, `jomo.lm()` returns a data frame, in which the original data and all imputed datasets are stacked onto each other.

`split()` splits the dataset by imputation number into a list of datasets, from which we need to exclude the first element (the original/incomplete data).

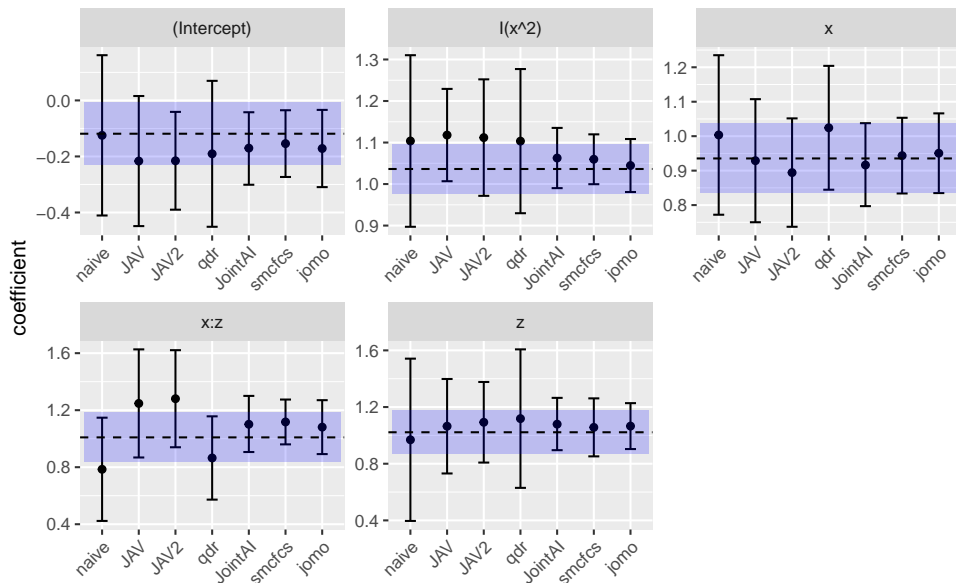
```
impobj_jomo_nonlin <- datalist2mids(split(jomo_nonlin,  
                                         jomo_nonlin$Imputation)[-1])
```

With the resulting `mids` object, analysis of the imputed data and pooling of the results works as in the above examples.

```
models_jomo_nonlin <- with(impobj_jomo_nonlin, lm(y ~ x*z + I(x^2)))  
res_jomo_nonlin <- summary(pool(models_jomo_nonlin))
```

15. Imputation with non-linear functional forms

15.5. Comparison of results



Practical

To practice imputation with non-linear forms or interaction terms, go to

https://emcbiostatistics.shinyapps.io/MICourse_MIadvanced

or download the instructions and data for the practical from Canvas
(Files > Principal documents > Multiple Imputation > Practical MIadvanced).

16. Imputation of longitudinal data

16.1. R package mice

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`

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

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

16. Imputation of longitudinal data

16.1. R package mice

`2l.pan` uses a linear two-level model with **homogeneous within group variances** using Gibbs sampling [14]. 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
```

16. Imputation of longitudinal data

16.1. R package mice

`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).

16. Imputation of longitudinal data

16.1. R package mice

`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` imputes univariate systematically and sporadically missing data using a two-level normal model using `lmer()` from package **lme4** (developed in the context of individual patient meta analysis. [7, 6])

16. Imputation of longitudinal data

16.1. R package mice

`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` imputes univariate systematically and sporadically missing data using a two-level normal model using `lmer()` from package **lme4** (developed in the context of individual patient meta analysis. [7, 6])

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

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

16. Imputation of longitudinal data

16.1. R package mice

`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.

16. Imputation of longitudinal data

16.1. R package mice

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

- x_1 (complete)
- x_2 (binary, 30% missing values)
- x_3 (3 categories, 30% missing values)
- x_4 (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 x_2 and x_3 .

16. Imputation of longitudinal data

16.1. R package mice

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 = F)
```

16. Imputation of longitudinal data

16.1. R package mice

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)))
mice_longimp <- summary(pool(models))
```

16. Imputation of longitudinal data

16.1. R package mice

Currently, there is only limited documentation and examples available that show how to use these functions in **mice**.

Technical details can be obtained from the methodological references given in the help files of the R functions.

A **vignette** on multi-level imputation with **mice** is available. It gives a more elaborate example of how to analyse such data.

16. Imputation of longitudinal data

16.2. R package JointAI

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 + x3 + x2 + x4 + time + I(time^2),
                      random = ~time|id, data = DFexlong2,
                      n.iter = 5000)
```

Again, convergence of the Gibbs sampler should be checked using a traceplot,

```
traceplot(JointAI_long)
```

before obtaining the results:

```
res_JointAI_long <- summary(JointAI_long)
```

Contrary to the two-level imputation of **mice**, non-linear associations are appropriately handled.

16. Imputation of longitudinal data

16.3. R package jomo

In **jomo**, the functions `jomo.lmer()` and `jomo.glmer()` can be used to impute longitudinal data with normal or non-normal outcomes.

In the multi-level setting, the **level of each variable** needs to be specified (1: repeated measurements, 2: baseline covariates), and **ordered the same way** the variables occur in the data frame.

```
library(jomo)
# specify the level of each variable
lvl <- c("id" = 1, y = 1, x1 = 2, x2 = 2, x3 = 2, x4 = 2, time = 1)

jomo_long <- jomo.lmer(formula = y ~ x1 + x2 + x3 + x4 +
                        time + I(time^2) + (1 + time|id),
                        data = DFexlong2[, names(lvl)], level = lvl)
```

Like in the example with non-linear effects, convergence of the imputation needs to be checked.

16. Imputation of longitudinal data

16.3. R package jomo

Again, the stacked dataframe returned by `jomo.lmer()` needs to be split by imputation number and the original data excluded, before fitting the model and pooling the results.

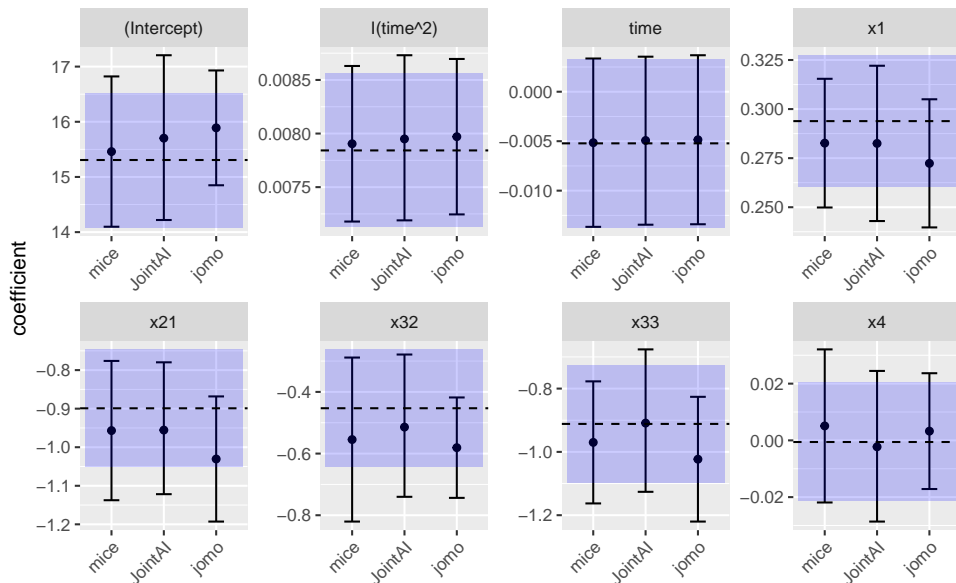
```
library(miceadds)
impobj_jomo_long <- datalist2mids(split(jomo_long,
                                       jomo_long$Imputation)[-1])
models_jomo_long <- with(impobj_jomo_long,
                         lmer(y ~ x1 + x3 + x2 + x4 + time + I(time^2) +
                             (time|clus)))
res_jomo_long <- summary(pool(models_jomo_long))
```

(Note: `jomo.lmer()` re-names the grouping variable to `clus`).

As in the examples for non-linear functional forms, congeniality of imputation models is maintained.

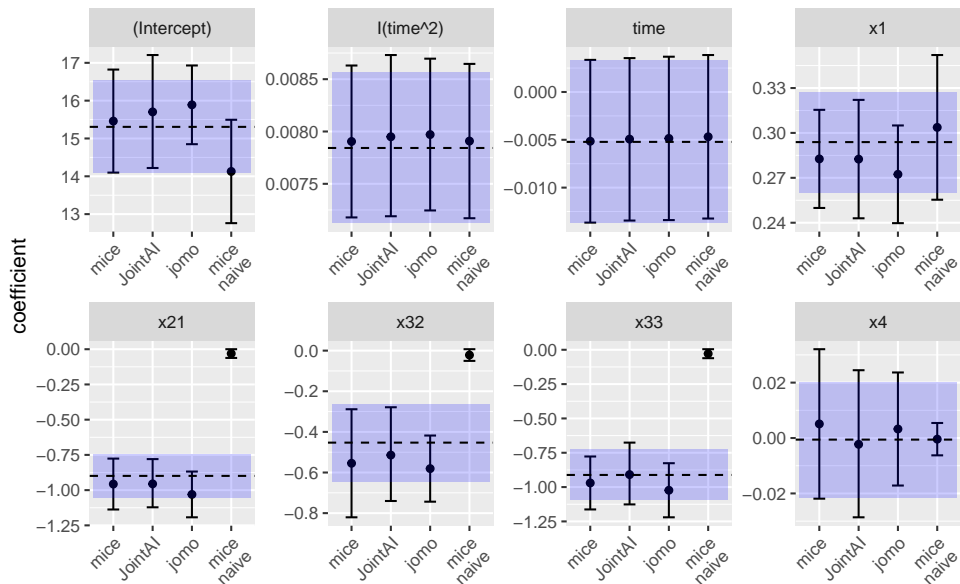
16. Imputation of longitudinal data

16.4. Comparison of results



16. Imputation of longitudinal data

16.4. Comparison of results



Practical

To practice imputation with longitudinal data, continue with the practical at

https://emcbiostatistics.shinyapps.io/MICourse_MIadvanced

or the offline version that can be downloaded from Canvas

(Files > Principal documents > Multiple Imputation > Practical MIadvanced).

17. Imputation of survival data

17.1. Results from literature

On slide 145 we have seen the rather complex formula for imputation of an incomplete covariate in survival data.

White et al. [20] derived versions of this model for different settings (binary or continuous incomplete covariate X , and continuous, categorical or no complete covariate Z) and investigated how to best approximate it.

They found that when **covariate effects and cumulative incidences are rather small**, using Z , D and $H_0(T)$, and possibly an interaction term, as predictor variables in the imputation for X in MICE may work satisfactorily.

However, in practice $H_0(T)$ is unspecified.

17. Imputation of survival data

17.1. Results from literature

Two main ideas:

- If covariate effects β_x and β_z are small, $H_0(t) \approx H(t)$, which can be approximated by the **Nelson-Aalen estimator**.
- **Estimate $H_0(T)$ in an additional step** inside the MICE procedure by fitting a Cox model on the imputed data.

Neither of these approaches takes into account uncertainty about $H_0(t)$ (but the impact is likely to be small).

17. Imputation of survival data

17.1. Results from literature

Two main ideas:

- If covariate effects β_x and β_z are small, $H_0(t) \approx H(t)$, which can be approximated by the **Nelson-Aalen estimator**.
- **Estimate $H_0(T)$ in an additional step** inside the MICE procedure by fitting a Cox model on the imputed data.

Neither of these approaches takes into account uncertainty about $H_0(t)$ (but the impact is likely to be small).

Based on results from their simulation study, White et al. conclude that **using Z , D and the Nelson-Aalen estimator $\hat{H}(T)$** as predictors for the imputation of X worked best.

However, some **bias towards the null** should be expected when covariates have large effects.

17. Imputation of survival data

17.2. R package mice

In **mice**, `nelsonaalen()` can be used to **calculate the Nelson-Aalen estimator**, to use it as covariate in the imputation.

```
survdat$H0 <- nelsonaalen(survdat, timevar = Time, statusvar = event)
```

Then, we can prepare the imputation using the same steps as in previous examples:

```
# setup run
imp0 <- mice(survdat, maxit = 0)
meth <- imp0$method
pred <- imp0$predictorMatrix

# specify normal imputation for continuous covariates
meth[c("x1", "x3")] <- "norm"

# remove event time from predictor (high correlation with H0)
pred[, "Time"] <- 0
```


17. Imputation of survival data

17.2. R package mice

With the modified arguments `method` and `predictorMatrix` we run the imputation:

```
survimp <- mice(survdat, maxit = 10, method = meth,  
               predictorMatrix = pred, printFlag = F)
```

To obtain the pooled results, we first fit the model of interest

```
cox_mice <- with(survimp, coxph(Surv(Time, event) ~ x1 + x2 + x3))
```

and pool and summarize the results.

```
res_mice_surv <- summary(pool(cox_mice))  
  
## Warning in mice.df(m, lambda, dfcom, method): Large sample  
assumed.
```

The warning message refers to the way the degrees of freedom for the formulas we saw in Part I (slide 32) are calculated and can be ignored.

17. Imputation of survival data

17.3. R package smcfcs

Using the package **smcfcs**, the same data can be imputed with the following syntax:

```
library(smcfcs)
smcfcs_surv <- smcfcs(originaldata = survdat, smtype = "coxph",
                      smformula = "Surv(Time, event) ~ x1 + x2 + x3",
                      method = c("", "", "logreg", "norm", "norm", ""),
                      numit = 20, rjlimit = 1500)
```

Convergence of the procedure should be checked, analogously to the previous example (see slide 172).

After the resulting object is converted to a `mids` object, fitting the model and pooling the results is identical to what was done with the data imputed by **mice**.

```
impobj_smcfcs_surv <- datalist2mids(smcfcs_surv$impDatasets)
models_smcfcs_surv <- with(impobj_smcfcs_surv,
                           coxph(Surv(Time, event) ~ x1 + x2 + x3))
res_smcfcs_surv <- summary(pool(models_smcfcs_surv))
```

17. Imputation of survival data

17.4. R package jomo

In the package **jomo**, the function `jomo.coxph()` can be used to impute our example survival data:

```
library(jomo)
jomo_surv <- jomo.coxph(formula = Surv(Time, event) ~ x1 + x2 + x3,
                        data = survdat)
```

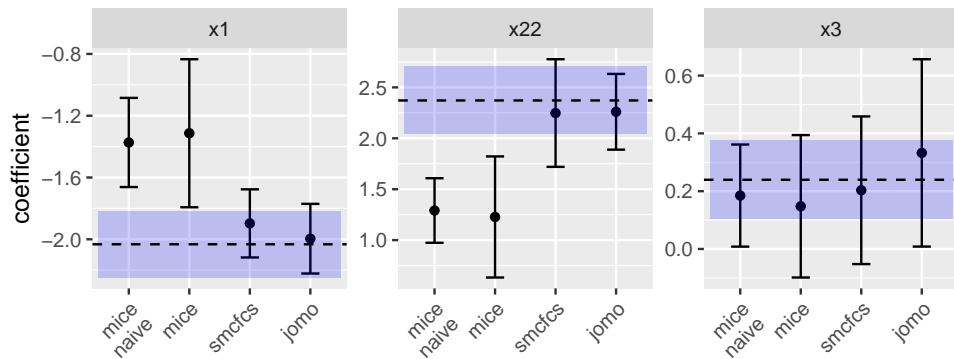
Note that the convergence of the procedure should be checked using `jomo.coxph.MCMCchain()` (see the previous examples using **jomo**).

To analyse & pool the imputed data the steps are identical to the other examples:

```
impobj_jomo_surv <- datalist2mids(split(jomo_surv,
                                       jomo_surv$Imputation)[-1])
models_jomo_surv <- with(impobj_jomo_surv,
                        coxph(Surv(Time, event) ~ x1 + x2 + x3))
res_jomo_surv <- summary(pool(models_jomo_surv))
```

17. Imputation of survival data

17.5. Comparison of results



The naive mice approach, and mice using the Nelson-Aalen estimator give very biased results for the effects of x1 and x2, but performed acceptably well for x3.

Note that the **true effects** (log HR) of x1 and x2 are **very large** (-2 and 2.5, respectively), and represent the setting where the approximation by the Nelson-Aalen estimate is **expected to be biased**.

Practical

To practice imputation with survival data, continue with the practical at

https://emcbiostatistics.shinyapps.io/MICourse_MIadvanced

or the offline version that can be downloaded from Canvas

(Files > Principal documents > Multiple Imputation > Practical MIadvanced).

Summary & Conclusion of Part III

- MICE requires **congenial & compatible imputation models** to work well.
- When this is not the case, (naive) use of MICE can lead to **biased results**.
- Common settings that require special attention are
 - **non-linear functional forms & interaction terms**
 - **longitudinal data**
 - **survival data**

Summary & Conclusion of Part III (cont.)

- When using the package **mice**, there are choices that can **reduce bias**
 - **pmm** tends to be less biased than **norm** for interactions or non-linear associations
 - JAV approach reduces bias in settings with interactions or non-linear associations
 - **special 2-level imputation methods** are available for longitudinal data
 - The **Nelson-Aalen estimator** can be used instead of the time variable for imputing survival data when effects are not too large.
- Generally, **problems** are more severe when
 - **proportions of missing values are large**,
 - effect sizes are large,
 - little other **covariate information** is available.

(Note that in the examples we had all of the above.)

Summary & Conclusion of Part III (cont.)

- In settings where MICE may not provide valid imputations, **alternative approaches** are available and should be considered.
- R packages that provide such alternative approaches are for example:
 - **JointAI** (non-linear & longitudinal)
 - **smcfcs** (non-linear & survival)
 - **jomo** (non-linear, longitudinal & survival)
- These packages are very young.
 - Hence, they may still have some problems.
 - ➔ **Use them carefully!** (and email the maintainer about problems)
 - They are under **active development**, so resolutions of bugs and features are frequently added.

Part IV

Multiple Imputation Strategies

Outline of Part IV

18. Strategies for using MICE

- 18.1 Imputation methods

- 18.2 Tips & Tricks

- 18.3 Number of imputed datasets

- 18.4 What to do with large datasets?

- 18.5 How much missing is too much?

- 18.6 Imputation of outcomes

- 18.7 Notes of caution & things to keep in mind

19. MICE and MI in the bigger picture

- 19.1 Other R packages that do imputation

- 19.2 Imputation in other software

- 19.3 Other approaches to handle missing values

18. Strategies for using MICE

18.1. Imputation methods

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:

<code>mice.impute.2l.lmer</code>	<code>mice.impute.logreg</code>	<code>mice.impute.passive</code>
<code>mice.impute.2l.norm</code>	<code>mice.impute.logreg.boot</code>	<code>mice.impute.pmm</code>
<code>mice.impute.2l.pan</code>	<code>mice.impute.mean</code>	<code>mice.impute.polr</code>
<code>mice.impute.2lonly.mean</code>	<code>mice.impute.midastouch</code>	<code>mice.impute.polyreg</code>
<code>mice.impute.2lonly.norm</code>	<code>mice.impute.norm</code>	<code>mice.impute.quadratic</code>
<code>mice.impute.2lonly.pmm</code>	<code>mice.impute.norm.boot</code>	<code>mice.impute.rf</code>
<code>mice.impute.cart</code>	<code>mice.impute.norm.nob</code>	<code>mice.impute.ri</code>
<code>mice.impute.lda</code>	<code>mice.impute.norm.predict</code>	<code>mice.impute.sample</code>

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

18. Strategies for using MICE

18.1. Imputation methods

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

<code>mice.impute.2l.binary</code>	<code>mice.impute.bygroup</code>
<code>mice.impute.2l.contextual.norm</code>	<code>mice.impute.eap</code>
<code>mice.impute.2l.contextual.pmm</code>	<code>mice.impute.grouped</code>
<code>mice.impute.2l.continuous</code>	<code>mice.impute.hotDeck</code>
<code>mice.impute.2l.eap</code>	<code>mice.impute.ml.lmer</code>
<code>mice.impute.2l.groupmean</code>	<code>mice.impute.plausible.values</code>
<code>mice.impute.2l.groupmean.elim</code>	<code>mice.impute.pls</code>
<code>mice.impute.2l.latentgroupmean.mcmc</code>	<code>mice.impute.pmm3</code>
<code>mice.impute.2l.latentgroupmean.ml</code>	<code>mice.impute.pmm4</code>
<code>mice.impute.2l.plausible.values</code>	<code>mice.impute.pmm5</code>
<code>mice.impute.2l.pls</code>	<code>mice.impute.pmm6</code>
<code>mice.impute.2l.pls2</code>	<code>mice.impute.tricube.pmm</code>
<code>mice.impute.2l.pmm</code>	<code>mice.impute.tricube.pmm2</code>
<code>mice.impute.2lonly.function</code>	<code>mice.impute.weighted.norm</code>
<code>mice.impute.2lonly.norm2</code>	<code>mice.impute.weighted.pmm</code>
<code>mice.impute.2lonly.pmm2</code>	

18. Strategies for using MICE

18.2. Tips & Tricks

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 from the flow-diagram in the second practical (e.g., `as.mids()`, `datalist2mids()`, `imputationList()`, ...)

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

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

18. Strategies for using MICE

18.3. Number of imputed datasets

Early publications on multiple imputation suggested that 3 – 5 imputations are sufficient and this is still a common assumption in practice.[12]

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.[13]

More **recent work** from various authors [21, 17, 5] 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.

18. Strategies for using MICE

18.3. Number of imputed datasets

A **suggested rule of thumb** is that the **number of imputed datasets** should be similar to the **percentage of incomplete cases**.^[21] Since this percentage depends on the size of the dataset, the **average percentage of missing values** per variable could be used as an alternative.^[17]

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.

18. Strategies for using MICE

18.4. What to do with large datasets?

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) and 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**.^[17]

18. Strategies for using MICE

18.4. What to do with large datasets?

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.

18. Strategies for using MICE

18.5. How much missing is too much?

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 handled **depends on the information that is available** to impute it.

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

18. Strategies for using MICE

18.5. How much missing is too much?

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 handled **depends on the information that is available** to impute it.

- Are there **strong predictor variables** 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 leave 25 cases to estimate the parameters of the predictive distribution.
- In a large set of $N = 5000$ subjects, 50% missing cases leave 2500 observed cases to estimate parameters.

18. Strategies for using MICE

18.6. Imputation of outcomes

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).

18. Strategies for using MICE

18.7. Notes of caution & things to keep in mind

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.

18. Strategies for using MICE

18.7. Notes of caution & things to keep in mind

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.
- **Imputation models must fit the data**
(correct assumption of error distribution and functional forms and possible interactions of predictor variables).
- Think carefully how to handle variables that are derived from other variables.
- Consider the impact the **visit sequence** may have.
- Choose an appropriate **number of imputations**.
- Make sure the imputation algorithm has **converged**.
- Use **common sense** when evaluating if the imputed values are plausible.

19. MICE and MI in the bigger picture

19.1. Other R packages that do imputation

In Part III of this course (and the second practical) we have worked with some R packages that perform imputation or provide functionality for missing data other than **mice**.

Currently, there are **218 packages** available on CRAN that **use the word “missing”** in either the title or description of the package, **132** that **use either “impute” or “imputation”** and **47** that **use the word “incomplete”**.

Not all of these packages perform imputation or are useful for our purposes, but even if we excluded those packages, the number of useful packages for dealing with missing data would still be too large to mention them all.

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

19. MICE and MI in the bigger picture

19.2. Imputation in other software

In this second half of the course, we have focused on (multiple) imputation using R.

Naturally, R is not the only statistical software that can perform multiple imputation.

- **Stata, SAS and MPLUS** provide packages/functions to perform multiple imputation and pool the results.
- There are macros and additional packages available, e.g., **smcfcs** is implemented for **Stata** as well
- **SPSS** provides some functionality to perform MI

19. MICE and MI in the bigger picture

19.3. Other approaches to handle missing values

Finally, we should not forget that **MICE is not the only method to handle missing values**.

Besides MICE, **multiple imputation** can be performed in a **joint model approach** (as for instance implemented in the R package **jomo**).

Furthermore,

- **direct likelihood methods**,
- **fully Bayesian methods**(as implemented in **JointAI**), or
- **weighted estimating equations**

are valid alternative approaches when data are incomplete.

References

References

- [1] Jonathan W Bartlett, Shaun R Seaman, Ian R White, James R Carpenter, and Alzheimer's Disease Neuroimaging Initiative.
Multiple imputation of covariates by fully conditional specification: accommodating the substantive model.
Statistical methods in medical research, 24(4):462–487, 2015.
- [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, 2017.

References (cont.)

- [4] 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.
- [5] 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.
- [6] Shahab Jolani.
Hierarchical imputation of systematically and sporadically missing data: An approximate bayesian approach using chained equations.
[Biometrical Journal](#), 60(2):333–351, 2018.

References (cont.)

- [7] Shahab Jolani, Thomas Debray, Hendrik Koffijberg, Stef Buuren, and Karel GM Moons.
Imputation of systematically missing predictors in an individual participant data meta-analysis: a generalized approach using mice.
Statistics in medicine, 34(11):1841–1863, 2015.
- [8] Roderick JA Little.
Missing-data adjustments in large surveys.
Journal of Business & Economic Statistics, 6(3):287–296, 1988.
- [9] Donald B Rubin.
Statistical matching using file concatenation with adjusted weights and multiple imputations.
Journal of Business & Economic Statistics, 4(1):87–94, 1986.
- [10] Donald B. Rubin.
Multiple Imputation for Nonresponse in Surveys.
Wiley Series in Probability and Statistics. Wiley, 1987.

References (cont.)

- [11] Donald B Rubin.
Multiple imputation after 18+ years.
[Journal of the American statistical Association](#), 91(434):473–489, 1996.
- [12] 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.
- [13] Joseph L Schafer.
[Analysis of incomplete multivariate data](#).
CRC press, 1997.
- [14] 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.

References (cont.)

- [15] Nathaniel Schenker and Jeremy MG Taylor.
Partially parametric techniques for multiple imputation.
Computational statistics & data analysis, 22(4):425–446, 1996.
- [16] 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.
- [17] Stef van Buuren.
Flexible Imputation of Missing Data.
Chapman & Hall/CRC Interdisciplinary Statistics. Taylor & Francis, 2012.
- [18] Stef Van Buuren, Hendrick C Boshuizen, Dick L Knook, et al.
Multiple imputation of missing blood pressure covariates in survival analysis.
Statistics in Medicine, 18(6):681–694, 1999.

References (cont.)

- [19] Gerko Vink and Stef van Buuren.
Multiple imputation of squared terms.
Sociological Methods & Research, 42(4):598–607, 2013.
- [20] Ian R White and Patrick Royston.
Imputing missing covariate values for the cox model.
Statistics in medicine, 28(15):1982–1998, 2009.
- [21] 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.
- [22] Recai M Yucel.
Multiple imputation inference for multivariate multilevel continuous data with ignorable non-response.
Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 366(1874):2389–2403, 2008.

Erasmus MC

University Medical Center Rotterdam



n.erler@erasmusmc.nl



N_Erler



NErler

Dep. Biostatistics: www.erasmusmc.nl/biostatistiek