Missing Values in Clinical Research (EP16)

Multiple Imputation

Nicole Erler

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

13 - 17 May, 2019

Erasmus MC
University Medical Center Rotterdam

Outline

Part I: Multiple Imputation

How does multiple imputation work?

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

Part II: Multiple Imputation Workflow

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

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

- 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**.

- 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**.

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**.

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.

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.

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.

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.

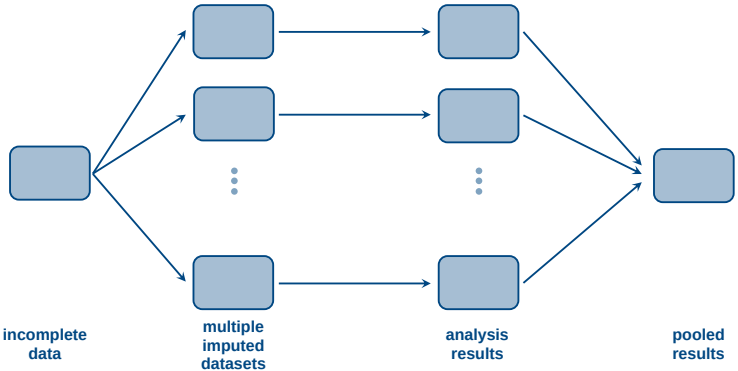
Example: survey about age, gender and height

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.

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.



In summary:

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

How can we actually get imputed values?

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

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

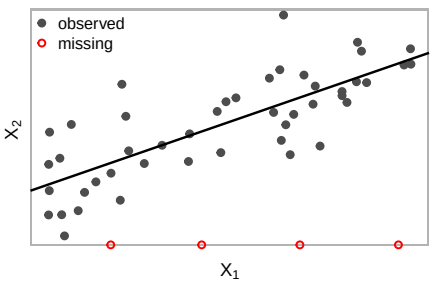
How can we actually get imputed values?

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



How can we actually get imputed values?

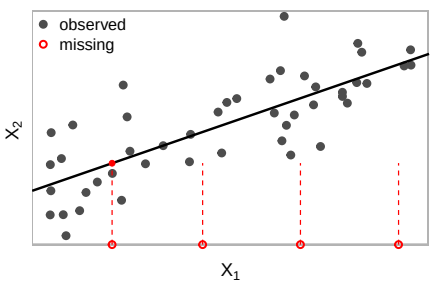
For now: assume only one continuous variable has missing values (**univariate missing data**)

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

Idea: Predict values

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

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



Problem:

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

Problem:

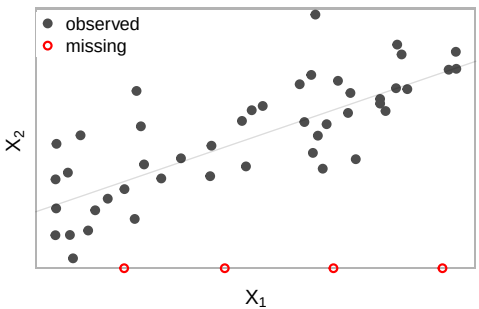
- We can obtain **only one imputed value** per missing value, but we wanted a whole distribution.
- The predicted values do not take into account the added **uncertainty** due to the missing values.

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

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

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

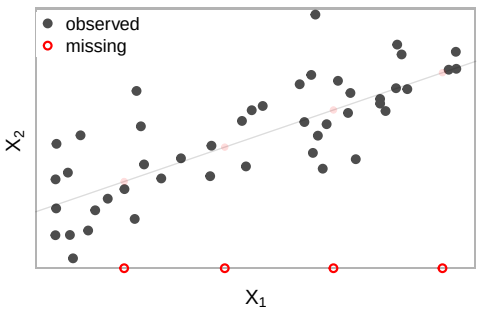
When plugging the different realizations of β into the predictive model, we obtain **slightly different regression lines**.



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

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

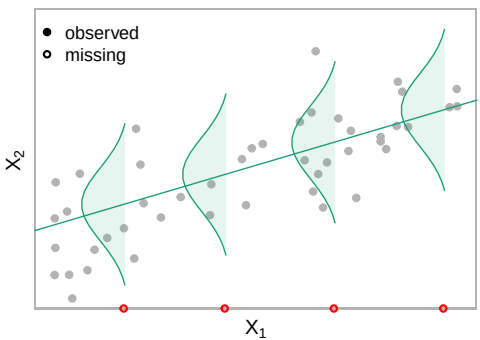
With each set of coefficients, we also get slightly **different predicted values**.



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

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

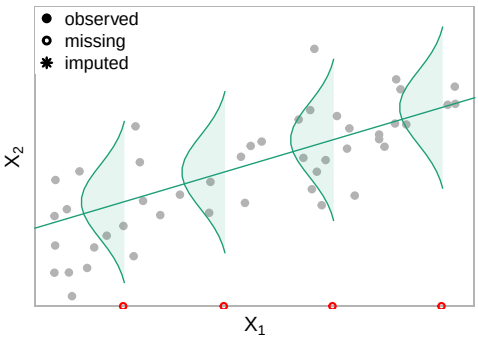
- the **mean** for each value is given by the **predictive model**, and



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

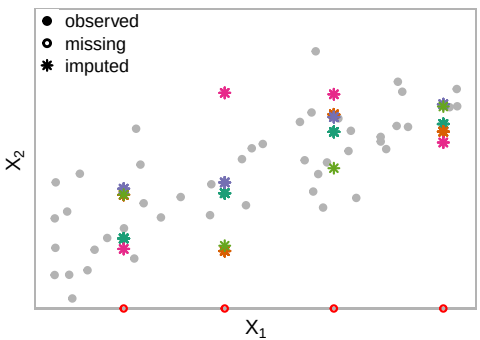
- the **mean** for each value is given by the **predictive model**, and
- the **variance** is determined by the variance of the residuals ε.



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.

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

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
⋮	⋮	⋮	⋮

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
⋮	⋮	⋮	⋮

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

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)

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.

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.

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.

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)

Notation

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

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

Notation

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

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

For example:

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

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

1: **for** j in $1, \ldots, p$: ▷ 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**

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

1: **for** j in $1, \ldots, p$: ▷ 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 $1, \ldots, T$: ▷ loop through iterations
6: **for** j in $1, \ldots, p$: ▷ loop through variables

10: **end for**
11: **end for**

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

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

10: **end for**
11: **end for**

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

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

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

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

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

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

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

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

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

1: **for** j in $1, \ldots, p$: ▷ 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
$\dot{X}_{-2}^1 = \left(\dot{X}_1^1, \dot{X}_3^0, \dot{X}_4^0\right)$.
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**

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

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

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

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

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

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

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

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

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

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

1: **for** j in $1, \ldots, p$: ▷ 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
$\dot{X}_{-2}^2 = \left(\dot{X}_1^2, \dot{X}_3^1, \dot{X}_4^1\right)$.
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**

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

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

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

One run through the algorithm ➡ one imputed dataset.

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

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

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

One run through the algorithm ➡ one imputed dataset.

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

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

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

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

One run through the algorithm ➡ one imputed dataset.

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

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.

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.

Why iterations?

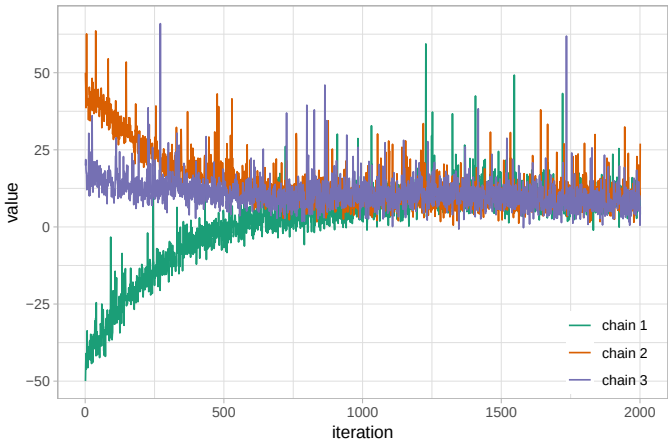
- Imputed values in one variable depend on the imputed values of the other variables (Gibbs sampling).
- If the starting values (random draws) are far from the actual distribution, imputed values from the first few iterations are not draws from the distribution of interest.

How many iterations?

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

How to evaluate convergence?

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



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

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).

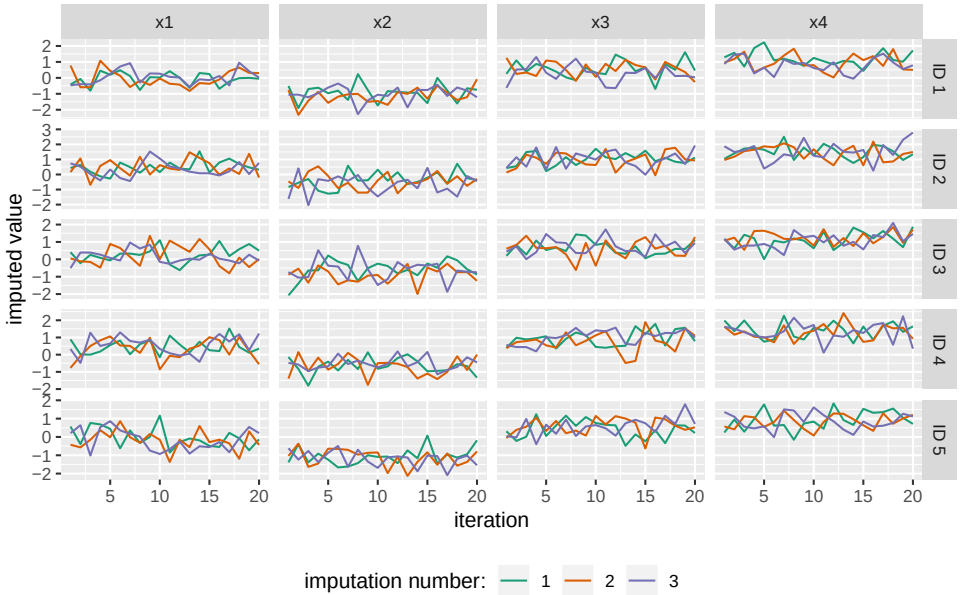
In imputation we have

- 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

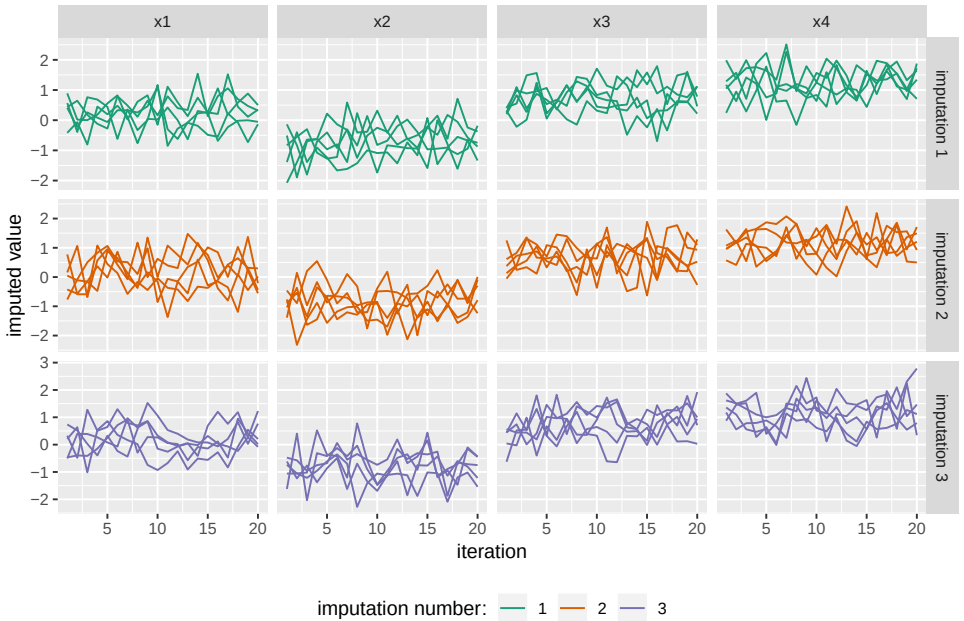
To check all chains separately could be very time consuming in large datasets (and storing all iterations from all imputed values is inefficient).

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


x1
x2
x3
x4
ID 1
ID 2
ID 3
ID 4
ID 5
imputed value
iteration
imputation number: 1 2 3

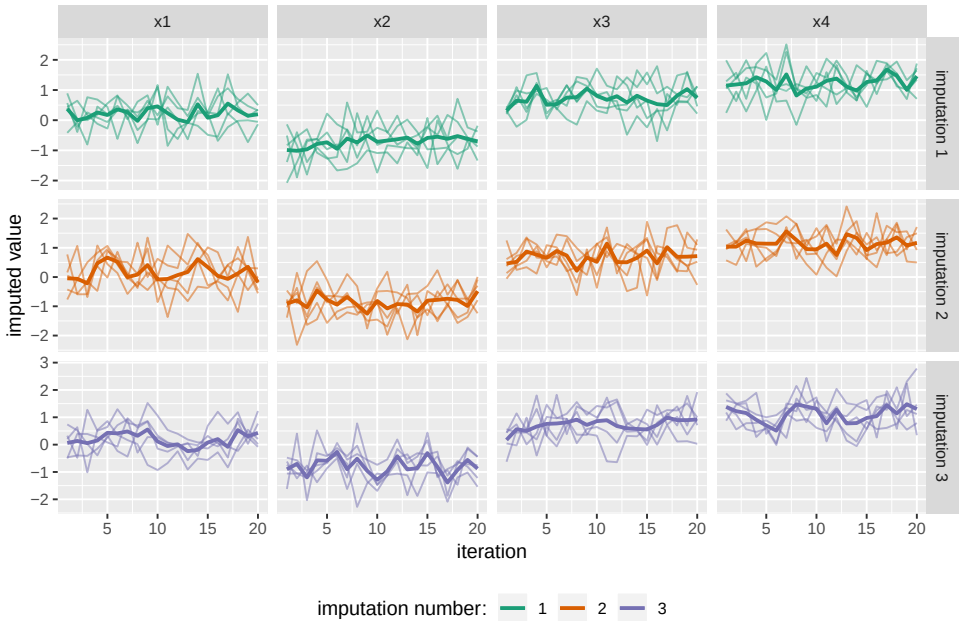
2. Imputation step

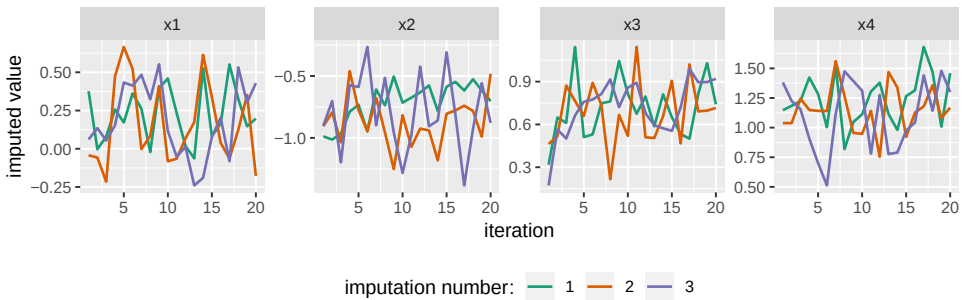
2.4. Checking convergence



2. Imputation step

2.4. Checking convergence




x1
x2
x3
x4
imputed value
iteration
imputation number: 1 2 3

3. Analysis step

Multiple imputed datasets:

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

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

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

3. Analysis step

Multiple imputed datasets:

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

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

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

Analysis model of interest, e.g.,

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

3. Analysis step

Multiple imputed datasets:

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

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

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

Analysis model of interest, e.g.,

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

Multiple sets of results:

	est.	se
β_0	0.35	0.21
β_1	0.14	0.02
β_2	-0.64	0.03
β_3	0.18	0.03

	est.	se
β_0	0.21	0.17
β_1	0.14	0.01
β_2	-0.62	0.03
β_3	0.24	0.03

	est.	se
β_0	-0.05	0.23
β_1	0.15	0.02
β_2	-0.57	0.03
β_3	0.2	0.04

Recall from slide 6:
We need to represent missing values by a **number of imputations**.
➡ m imputed datasets

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.

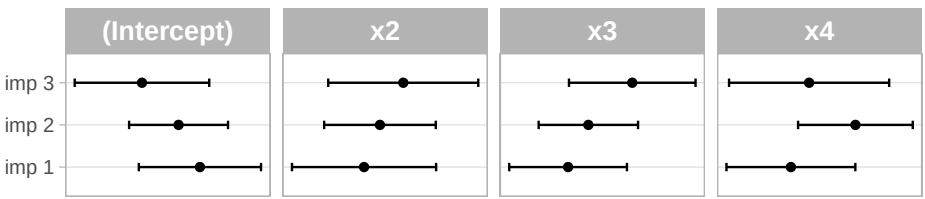
Recall from slide 6:
We need to represent missing values by a **number of imputations**.
➡ m imputed datasets

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.

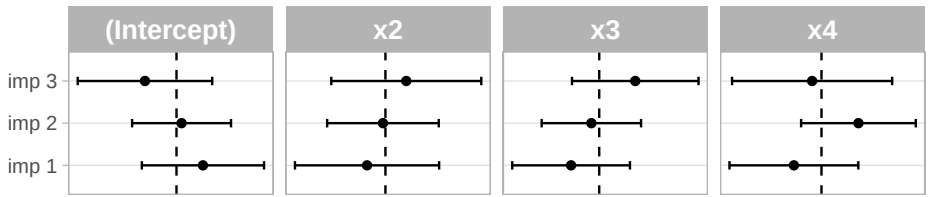
We want to **summarize** the results and describe **how (much) the results vary** between the imputed datasets.

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

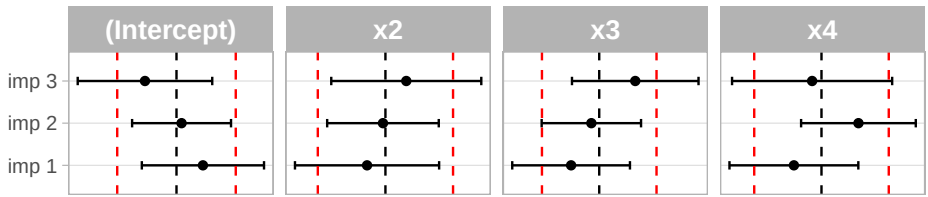
- **within** imputation (represented by the confidence intervals)
- **between** imputation (horizontal shift between imputations)



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

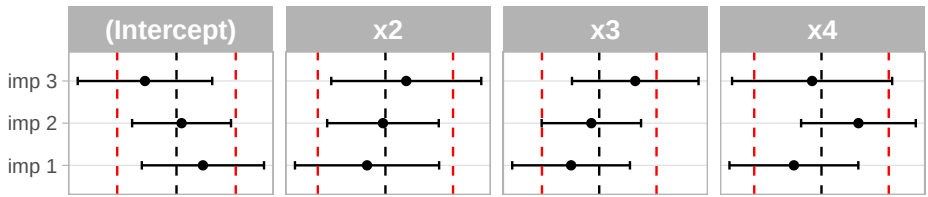


To summarize the results, we can take the mean of the results from the separate analyses. This is the **pooled point estimate**.



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

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).

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., $var(\beta) = se(\beta)^2$)

Pooled parameter estimate:

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

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

Average within imputation variance:

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

Between imputation variance:

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

Total variance:

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

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

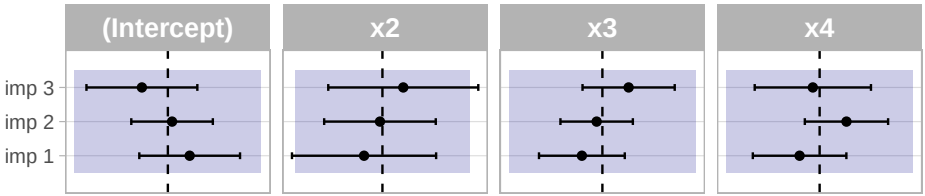
$$\nu = (m-1)\left(1 + r_m^{-1}\right)^2,$$

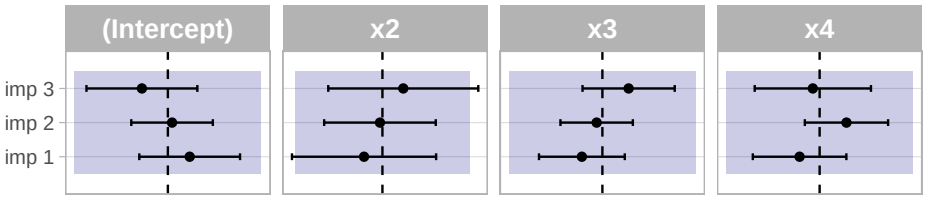
where $r_m = \frac{(B+B/m)}{\bar{U}}$ is the relative increase in variance that is due to the missing values.

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.


(Intercept)
x2
x3
x4
imp 3
imp 2
imp 1
parameter estimate & 95% confidence interval



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

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

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).

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.

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.

$$\mathbf{y} = \begin{matrix} \mathbf{y}_{obs} \\ \\ \\ \\ \mathbf{y}_{mis} \end{matrix} \left\{ \begin{bmatrix} y_1 \\ \vdots \\ y_q \\ NA \\ \vdots \\ NA \end{bmatrix} \right. \qquad \mathbf{X} = \begin{matrix} \mathbf{X}_{obs} \\ \\ \\ \\ \mathbf{X}_{mis} \end{matrix} \left\{ \begin{bmatrix} 1 & x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_{q1} & \dots & x_{qp} \\ 1 & x_{q+1,1} & \dots & x_{q+1,p} \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_{n1} & \dots & x_{np} \end{bmatrix} \right.$$

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 $\mathbf{X}_{mis}$.

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 $\mathbf{X}_{mis}$.

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 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 $\mathbf{X}_{mis}$.

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.

To determine the **expectation** of the posterior distribution of the missing values, usually a **regression model** is used, that depends on the unknown coefficients $\boldsymbol{\beta}$.

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

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

To determine the **expectation** of the posterior distribution of the missing values, usually a **regression model** is used, that depends on the unknown coefficients $\boldsymbol{\beta}$.

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

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

To **impute** missing values, while **taking into account the uncertainty about $\boldsymbol{\beta}$ and σ**, the estimated posterior distributions of the missing values and parameters are multiplied

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

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

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

Then $p(\mathbf{y}_{mis} \mid \mathbf{X}_{mis}, \boldsymbol{\beta}, \boldsymbol{\sigma})$ 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 $\tilde{\sigma}^2$.

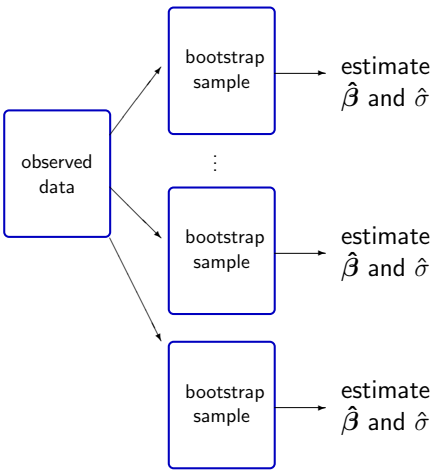
This is actually the approach we have seen previously on Slides 12/13 and 19.

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.

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.

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

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

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

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

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

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

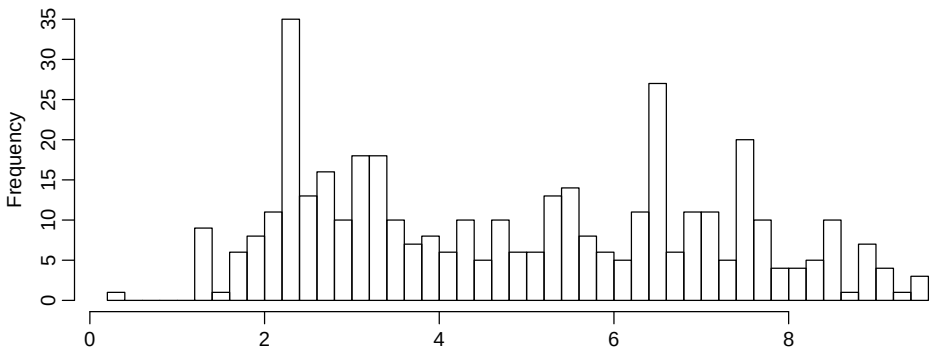
Analogous to Bayesian multiple imputation, for a normal imputation model, $p()$ is the normal distribution and

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

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

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.



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.

The steps in PMM:

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

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

3. Calculate the predicted value for the incomplete data

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

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.

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

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

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

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

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

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

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

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

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**.

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**.

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.

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.

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.

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.

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()`
- ...

Summary of Part I

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

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

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

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

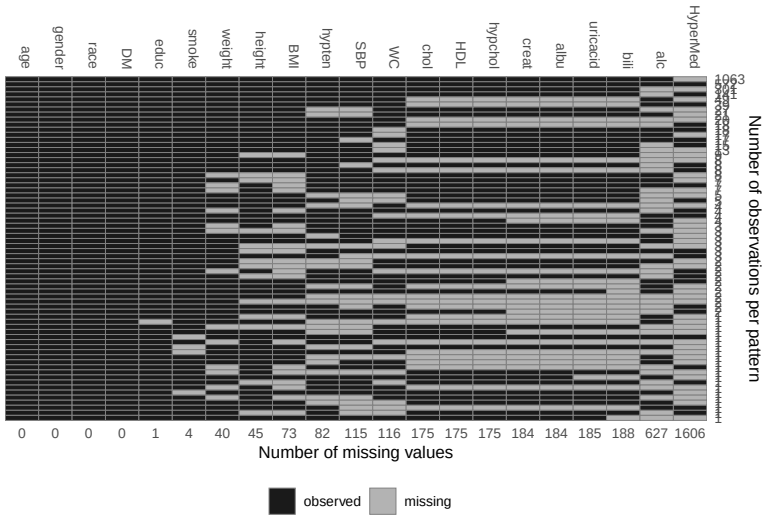
```
mdp <- mice::md.pattern(NHANES, plot = FALSE)
head(mdp[, -c(7:14)]) # omit some columns to fit it on the slide

##      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
## 1063   1      1    1  1    1     1       1     1    1        1    1   1        0 1
## 141    1      1    1  1    1     1       1     1    1        1    1   0        1 1
## 301    1      1    1  1    1     1       1     1    1        1    1   0        0 2
## 2      1      1    1  1    1     1       1     1    1        1    0   1        0 2
## 1      1      1    1  1    1     1       1     1    1        1    0   0        0 3

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

##   age gender race DM educ smoke hypchol creat albu uricacid bili alc HyperMed
## 1   1      1    1  1    1     0       1     1    1        1    1   1        1    1
## 1   1      1    1  1    1     0       1     1    1        1    1   1        0    2
## 1   1      1    1  1    1     0       0     0    0        0    0   0        0   10
## 1   1      1    1  1    1     0       1     1    1        1    1   1        0    4
## 1   1      1    1  1    0     1       0     0    0        0    0   1        0   12
##     0      0    0  0    1     4     175   184  184      185  188 627     1606 3975
```

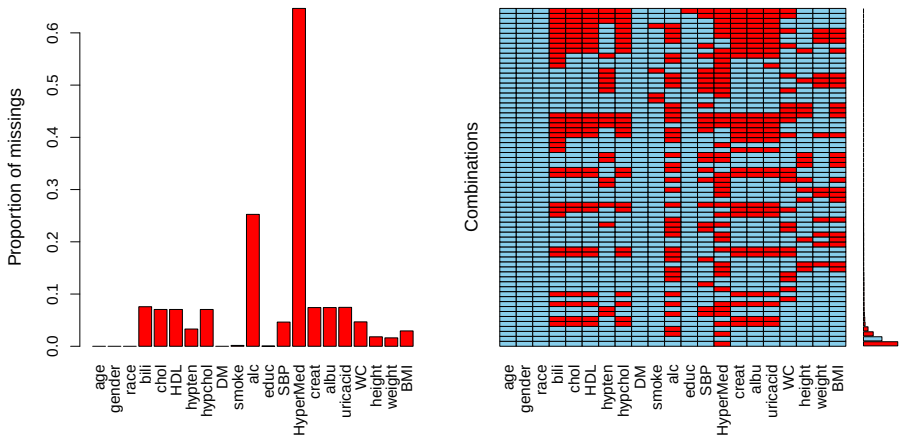
```
par(mar = c(5, 0.5, 1, 3), mgp = c(2, 0.6, 0))
JointAI::md_pattern(NHANES)
```



6. Know your data

6.1. Missing data patterns

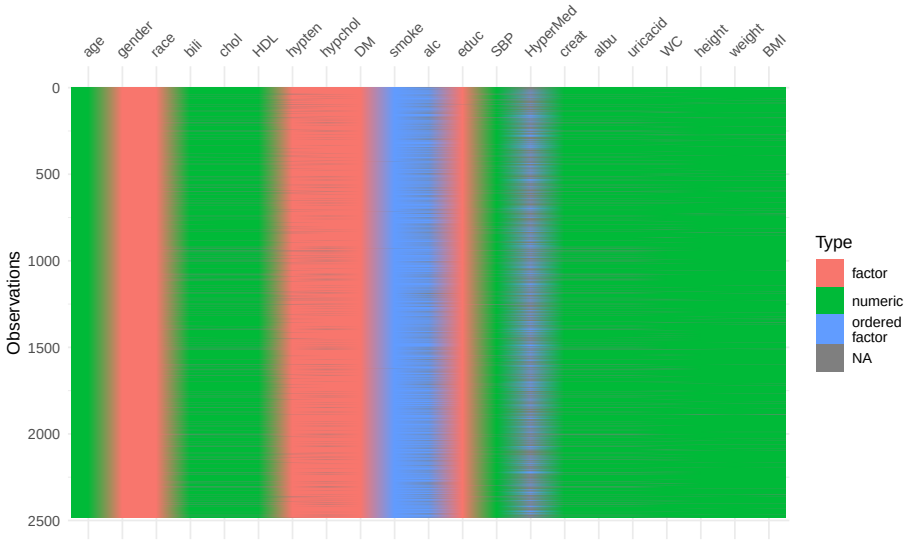
```
par(mar = c(6, 3, 2, 1))
VIM::aggr(NHANES, prop = T, numbers = FALSE)
```



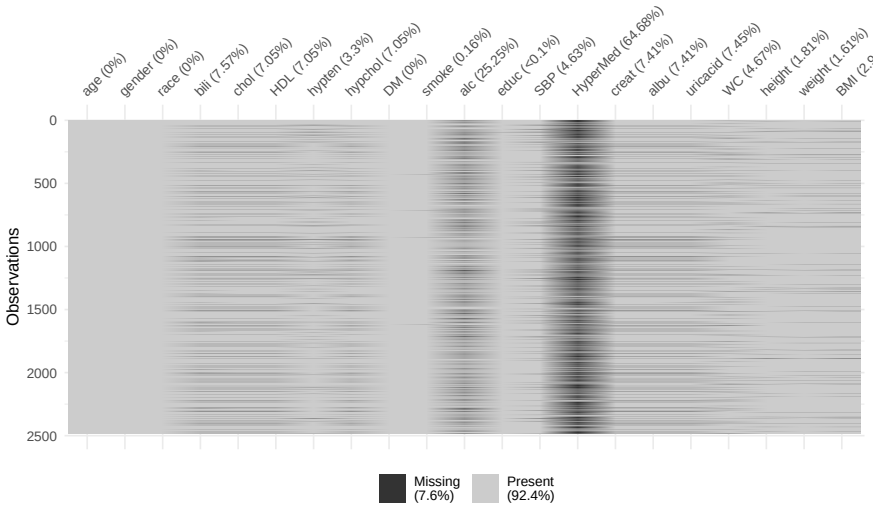
6. Know your data

6.1. Missing data patterns

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



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



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

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

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

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

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

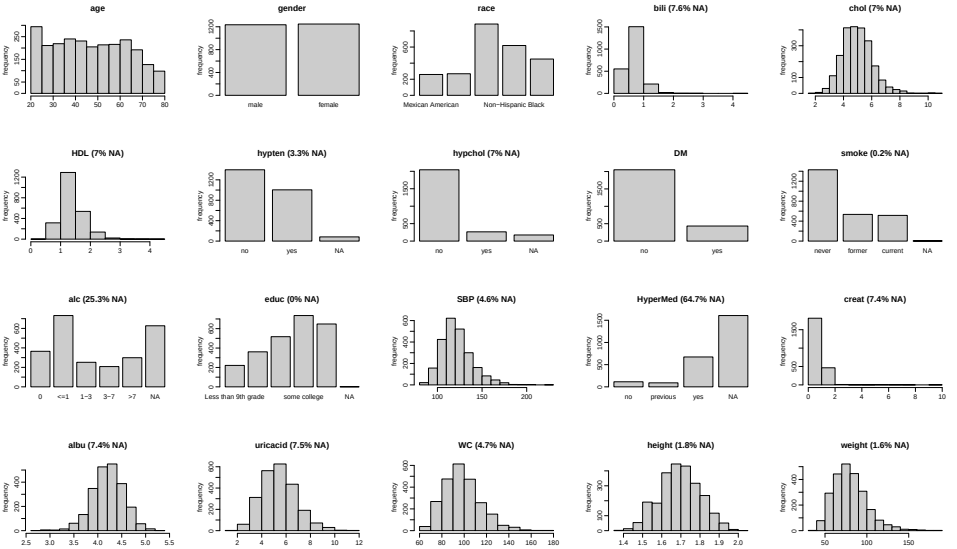
```
##          # NA % NA
## age         0 0.00
## gender      0 0.00
## race        0 0.00
## DM          0 0.00
## educ        1 0.04
## smoke       4 0.16
## weight     40 1.61
## height     45 1.81
## BMI        73 2.94
## hypten     82 3.30
## SBP       115 4.63
```

```
##           # 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

```
par(mgp = c(2, 0.6, 0))
JointAI::plot_all(NHANES)
```



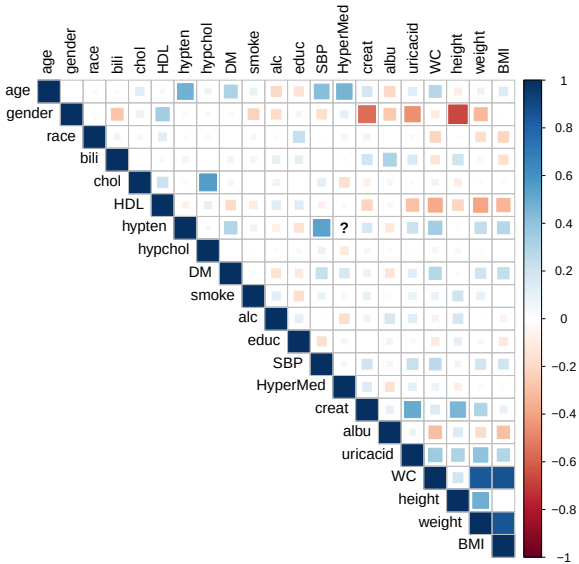
A quick (and dirty) way to check for strong correlations between variables is:

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

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

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

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


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

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

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

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

Knowing your data also means 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?

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.

Practical

To practice the content of the previous section in an **interactive tutorial**, go to

https://emcbiostatistics.shinyapps.io/EP16_IncompleteData

or find the **html version** of the practical here:

https://nerler.github.io/EP16_Multiple_Imputation/practical/incompletedata/Practical_IncompleteData.html

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
##       age    gender      race      bili      chol       HDL
##        ""        ""        ""     "pmm"     "pmm"     "pmm"
##    hypten   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)
```

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      1    1    1    1   1      1       1  1     1   1
## gender    1      0    1    1    1   1      1       1  1     1   1
## race      1      1    0    1    1   1      1       1  1     1   1
## bili      1      1    1    0    1   1      1       1  1     1   1
## 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.

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 automatically chosen imputation models are appropriate.

```
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 = FALSE)
```

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,]     26.23 27.68
## [2,]     25.99 26.89
## [3,]     24.06 23.80
## [4,]     28.35 27.84
## [5,]     25.82 24.75
```

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

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.

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**.

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.

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

Passive imputation can also be useful in settings where

- imputation models include 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.

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

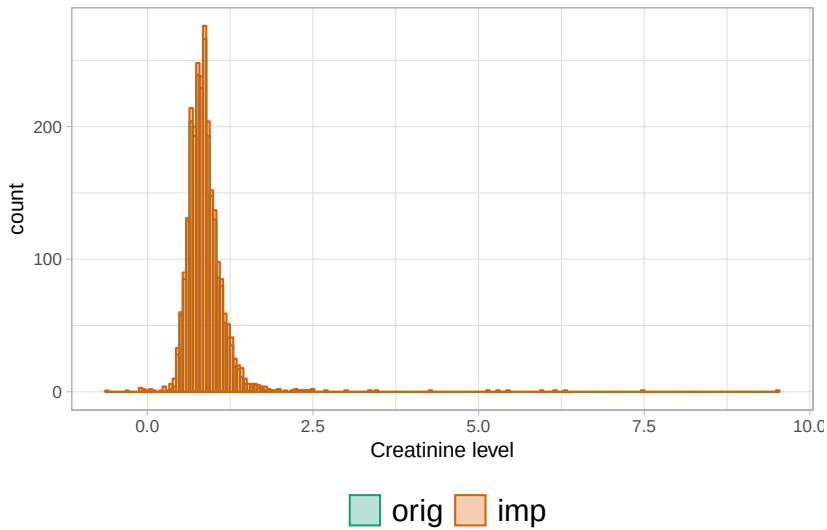
Helpful functions are

- `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.
## -0.1014  0.6908  0.8300  0.8863  0.9900  9.5100
```


count
200
100
0
0.0
2.5
5.0
7.5
10.0
Creatinine level
orig
imp

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

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

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

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

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

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

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

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

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

Example:
If `BMI` is passively imputed (calculated) before the new imputations for `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`.

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

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

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

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

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

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

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

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

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

`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)

`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.

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

"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]*

Practical

To practice the content of the previous section in an **interactive tutorial**,

https://emcbiostatistics.shinyapps.io/EP16_MImice

or find the **html version** of the practical here:

https://nerler.github.io/EP16_Multiple_Imputation/practical/mimice/Practical_MImice.html

The log of the automatic changes (slide 88) is returned as part of the `mids` object:

```
head(imp2$loggedEvents)

## NULL
```

With columns

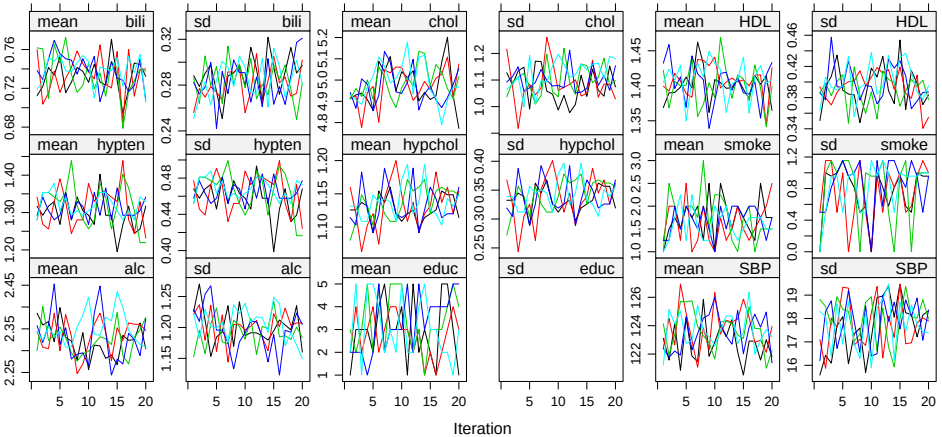
`it`	iteration number
`im`	imputation number
`co`	column number in the data
`dep`	dependent variable
`meth`	imputation method used
`out`	names of altered or removed predictors

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).

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



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.

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

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

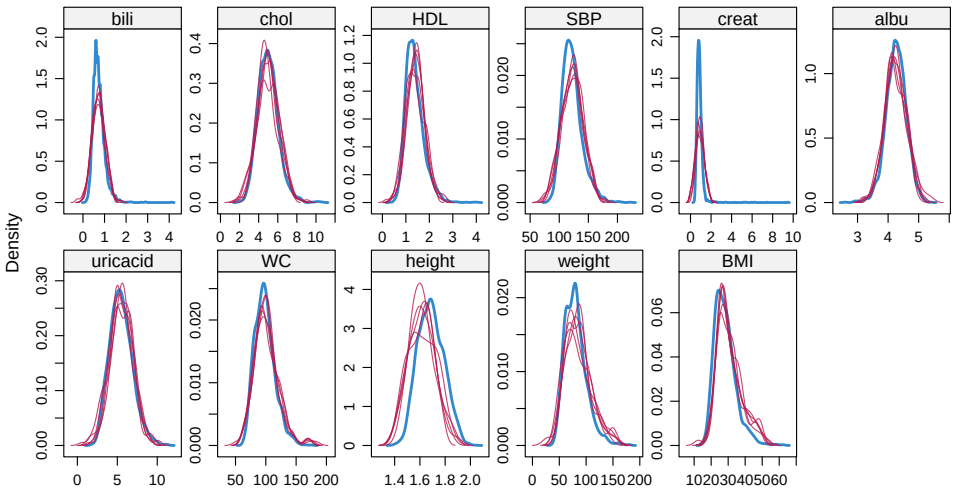
Recall:
The **conditional** distributions (given all the other variables in the imputation model) of the imputed values are assumed to be the same as the conditional distributions of the observed data.

mice provides several functions for visual diagnosis of imputed values:

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

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

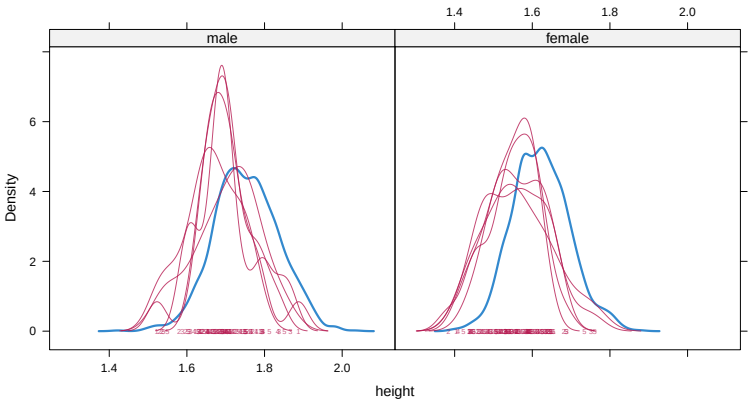
```
densityplot(imp2)
```



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

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

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



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
```

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
```

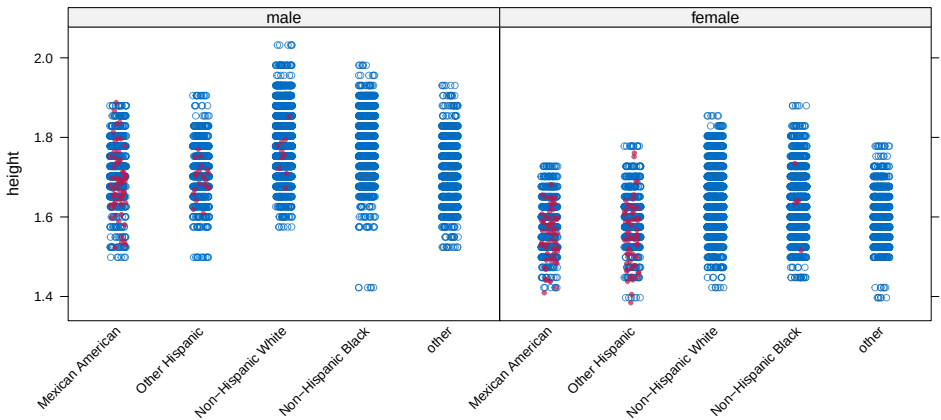
However, there are not enough missing values of `height` per categories of `race` to estimate densities.

```
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
```

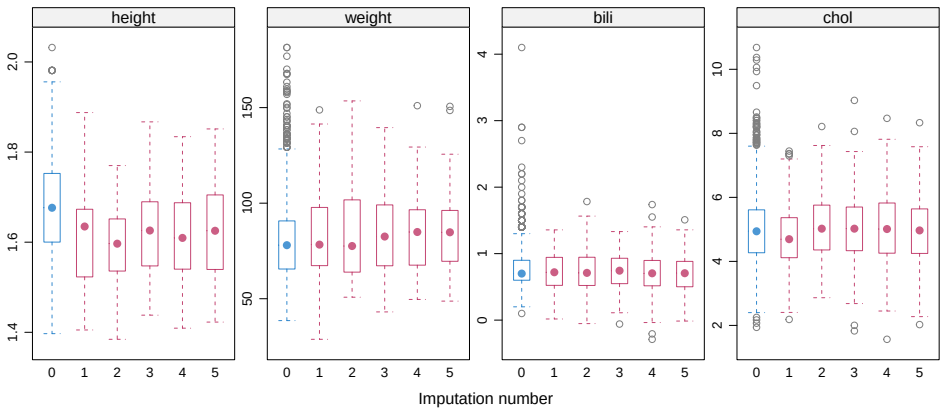
In that case, a `stripplot()` may be better suited. Here we can also split the data for `gender` and `race`.

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



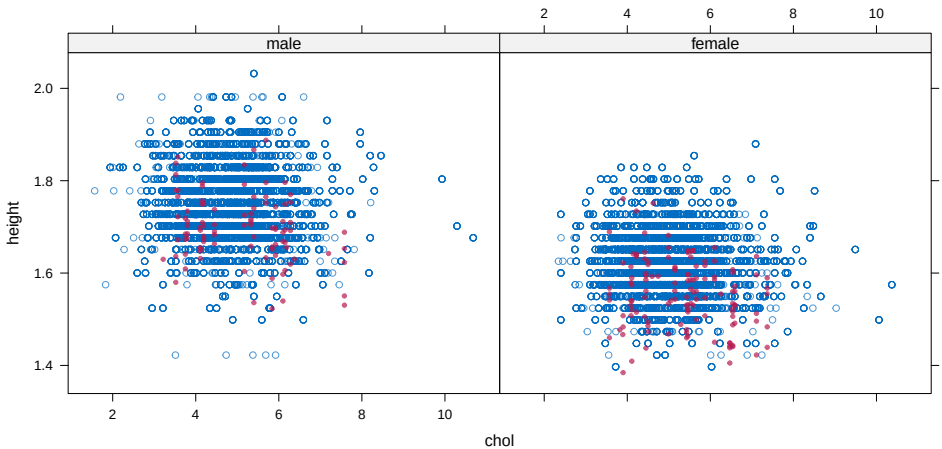
Alternatively, observed and imputed data can be represented by box-and-whisker plots:

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



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

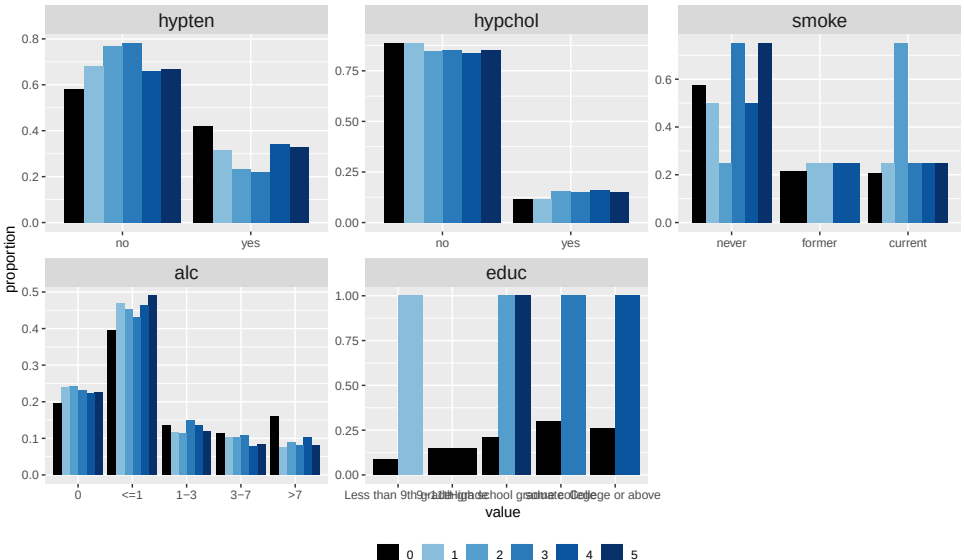
```
xyplot(imp2, height ~ chol|gender, pch = c(1,20))
```



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

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

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

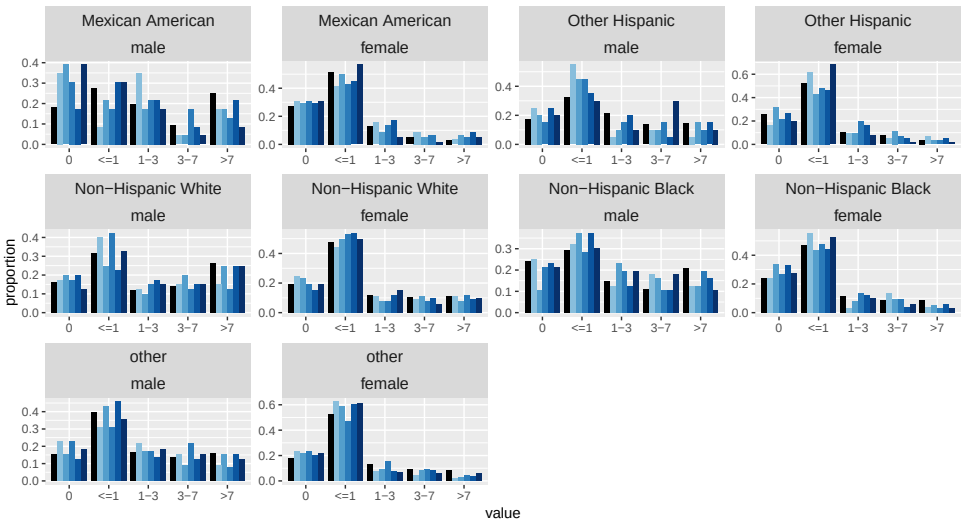


`smoke` and `educ` have very few missing values (4 and 1, respectively), so we do not need to worry about differences between observed and imputed data for those variables.

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

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

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



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

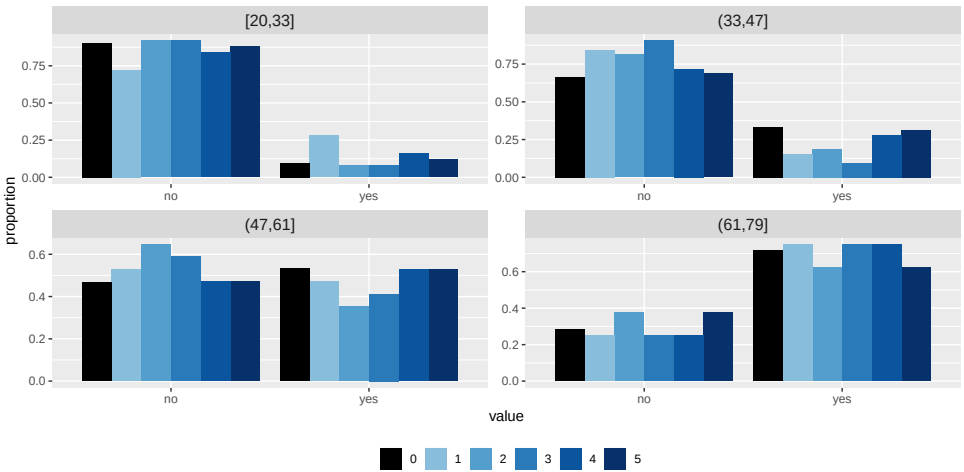
```
round(sapply(split(NHANES[, "age"], addNA(NHANES$hypten)), summary), 1)

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

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

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

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



Practical

To practice the content of the previous section in an **interactive tutorial**,

`https://emcbiostatistics.shinyapps.io/EP16_MIcheck`

or find the **html version** of the practical here:

`https://nerler.github.io/EP16_Multiple_Imputation/practical/mimice/Practical_MIcheck.html`

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

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

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

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

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

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

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

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

##              estimate std.error statistic       df p.value
## (Intercept)    -7.479     0.400   -18.689 2451.822   0.000
## age             0.056     0.004    12.931 2346.524   0.000
## genderfemale   -0.423     0.126    -3.349 2428.403   0.001
## hypcholyes     -0.011     0.187    -0.059  440.569   0.953
## BMI             0.105     0.009    11.569 2419.645   0.000
## smoke.L         0.062     0.117     0.530 2220.186   0.596
## smoke.Q        -0.076     0.115    -0.660 2439.911   0.510
## alc.L          -0.521     0.161    -3.244  409.383   0.001
## alc.Q           0.106     0.194     0.545   38.677   0.586
## alc.C          -0.042     0.194    -0.215   30.534   0.830
## alc^4          -0.089     0.185    -0.480  177.468   0.631
```

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()`.

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 0.3229105 0.2908562 0.3550403 NaN
```

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

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,] 0.6993471
```

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.

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

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

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

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

Calculate correlation or covariance of imputed data:

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

These functions take `mids` objects as input.

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

```
mice::complete(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 orginal data?
  (if `action` is `"long"`, `"broad"` or `"repeated"`)

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

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

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

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

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

```
## Error in ibind(imp2, imp2b):  Differences detected between
‘x$blocks‘ and ‘y$blocks‘
```

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

## Error in eval(expr, envir, enclos):  object 'imp2combi' not found
```

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.

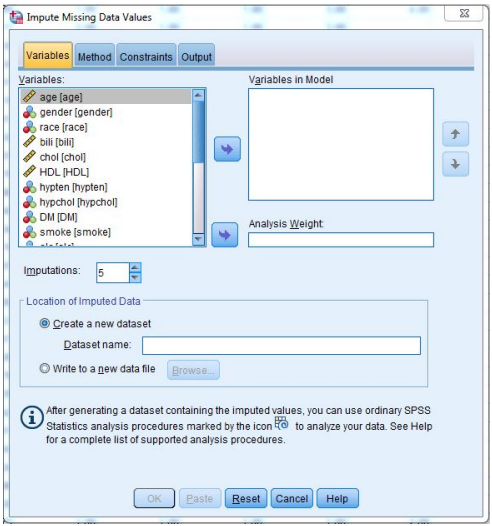
A walk-through how to do multiple imputation in SPSS can be found

- **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

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

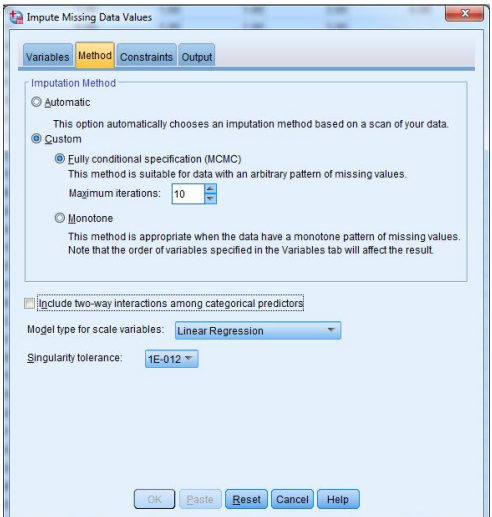
- **for older versions of SPSS**
  > Help
  > Case Studies
  > Missing Values Option
  > Multiple Imputation
  > Using Multiple Imputation to Complete and Analyze a Dataset
- **for newer versions online**
  https://www.ibm.com/support/knowledgecenter/en/SSLVMB_24.0.0/spss/tutorials/mi_table.html

The **procedure** itself is located in the menu
> Analyze
> Multiple Imputation
> Impute Missing Data Values



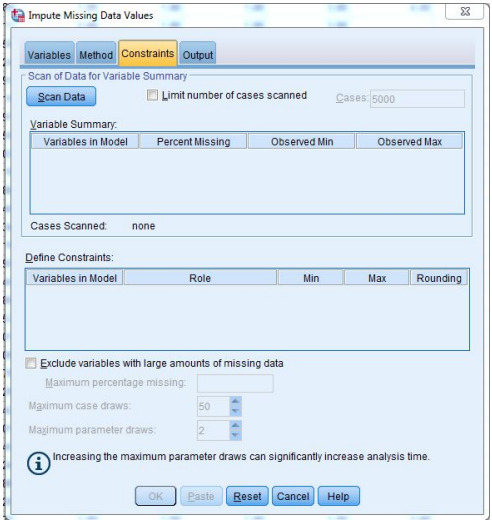
SPSS lets you

- specify number of imputations



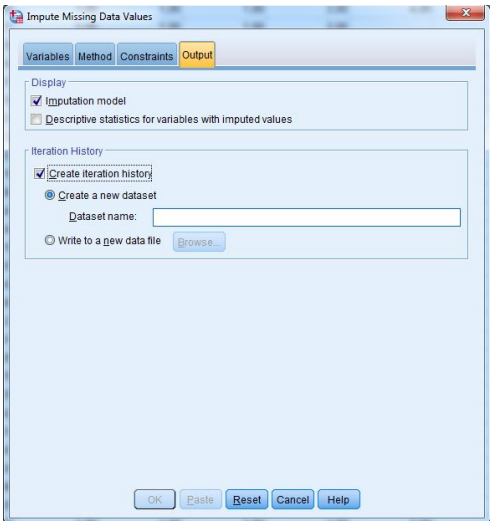
SPSS lets you

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



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



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

SPSS does not let you

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

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

Summary of Part I

6. Know your data

- ...

Summary of Part I (cont.)

7. Imputation with mice

- ...

Summary of Part I (cont.)

8. Convergence & Diagnostics

- ...

9. Analysis & Pooling

- ...

Summary of Part I (cont.)

10. Additional functions in mice

- ...

11. Multiple Imputation in SPSS

- ...

Part III
When MICE might fail

Part IV
Multiple Imputation Strategies

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, Hendriek 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.

n.erler@erasmusmc.nl
N_Erler
NErler

Dep. Biostatistics: www.erasmusmc.nl/biostatistiek