Multiple Imputation and subsequent calculations

Simon Ress | Ruhr-Universität Bochum

September 22, 2022

Workshop at hr&c, Bochum, 2021

Content

Introduction

Introduction

What is this workshop about?

- Specific methods for working with multiple imputed datasets

Why are special methods for working with multiple imputed data important?

- We always want to estimate the true (causal) effect of an event

Does just using a specific algorithms leads to unbiased estimates?

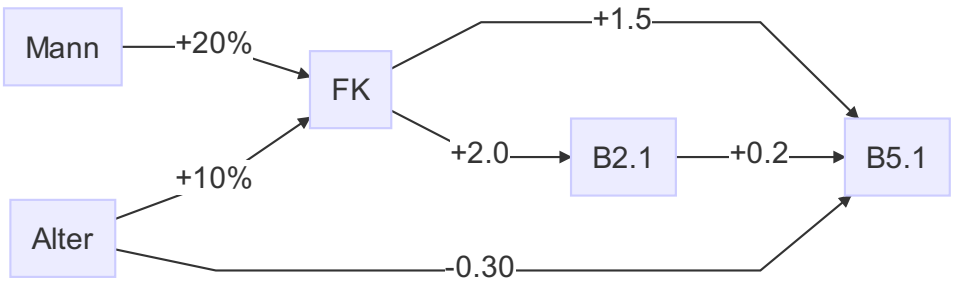
- No, there are other effects influencing the bias of an estimate

This workshop therefore follows a **holistic approach**. Based on methods for **calculating unbiased effects** in data without missing values, the different **missing patterns** are then introduced, the effect of different **missing-handling techniques** on the bias is demonstrated and then **multiple imputation** is introduced as a method for reducing the bias.

Examplary Data Set

Planing the interdependencies of variables

- Every variable is constructed by an *random term*
- Some variables are influenced by *values of other variables*
- **e.g. "Mann" = 1 increases the probability for FK=1 by 20%**



Data Set Creation

```
#Create variables
set.seed(415)
Mann = ifelse(runif(5000,0,1) < 0.50, 1, 0)
Alter = as.numeric(cut(runif(5000,20,70),
                       c(20,30,40,50,60,70)))
FK = ifelse((Mann*0.2 + Alter*0.1 +
             runif(5000,0,0.6)) > 0.95, 1, 0)
B2.1 = as.numeric(cut(FK*2 +
                      rnorm(5000,2,0.35), c(0,1,2,3,4,6)))
set.seed(1015)
B5.1 = as.numeric(cut(FK*1.5 + B2.1*0.2 + Alter*(-0.30) +
                      rnorm(5000,2.5,0.30), c(-2,1,2,3,4,8)))

#Build data frame
df = data.frame(Mann, Alter, FK, B2.1, B5.1)
```

View Data Set

```
head(df,10)
```

```
##    Mann Alter FK B2.1 B5.1
## 1     0     4  0    2    2
## 2     0     4  0    1    2
## 3     1     5  0    3    2
## 4     1     5  1    4    3
## 5     0     4  0    2    2
## 6     1     3  1    4    4
## 7     0     3  0    3    3
## 8     0     1  0    3    3
## 9     0     4  0    2    2
## 10    0     2  0    2    3
```

A. Check whether true effects can be estimated

Table 1: lm(FK~Mann) | Mann: +0.2

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.0596	0.0070	8.5438	0
Mann	0.2001	0.0099	20.1491	0

Table 2: lm(FK~Alter) | Alter: +0.1

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.1406	0.0110	-12.7686	0
Alter	0.1002	0.0033	30.0803	0

Table 3: | FK: +2

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.4980	0.0079	318.0197	0
FK	2.0115	0.0197	101.8565	0

A. Evaluation whether true effects was estimated

Yes, in all three models the true effect was estimated (taken the confidence interval into account).

B. Check whether true effects can be estimated

Table 4: lm(B5.1~FK) | FK: +1.5

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.6745	0.0088	302.2230	0
FK	1.4519	0.0222	65.2572	0

Table 5: lm(B5.1~B2.1) | B2.1: +0.2

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.3494	0.0283	47.6337	0
B2.1	0.5521	0.0096	57.5786	0

Table 6: lm(B5.1~Alter) | Alter: -0.3

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3.2286	0.0251	128.6139	0
Alter	-0.1088	0.0076	-14.3234	0

B. Evaluation whether true effects was estimated

- It is not obvious from the estimates and standard error whether the true effect was calculated
- Verification by the exact calculation of the confidence interval is needed

B.II Check if confidence interval encloses true effect

Recap Confidence Interval A confidence interval is an interval that contains the population parameter with probability $1 - \alpha$. A confidence interval takes on the form:

$$\overline{X} \pm t_{\alpha/2,N-1} * S_{\overline{X}}$$

where $t_{\alpha/2,N-1}$ is the (z-)value needed to generate an area of $\alpha/2$ in each tail of a t-distribution with n-1 degrees of freedom and $S_{\overline{X}} = \frac{s}{\sqrt{N}}$ is the standard error of the mean (s=standard deviation).

Z-value (standard deviation)	p-value (probability)	Confidence level
$<$ -1,65 oder $>$ +1,65	$<$ 0,10	90%
$<$ -1,96 oder $>$ +1,96	$<$ 0,05	95%
$<$ -2,58 oder $>$ +2,58	$<$ 0,01	99%

B.II Check if confidence interval encloses true effect (B5.1~FK)

Table 8: lm(B5.1~FK) | FK: +1.5

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.6745	0.0088	302.2230	0
FK	1.4519	0.0222	65.2572	0

- **True effect: +1.5**
- Confidence Interval $= \overline{X} \pm t_{\alpha/2,N-1} * S_{\overline{X}}$
- **lower.bound** = 1.4519152 - 0.0222491 * 1.96 = **1.408**
- **upper.bound** = 1.4519152 + 0.0222491 * 1.96 = **1.496**

-> True effect is **not covered** by the confidence interval

B.II Check if confidence interval encloses true effect (B5.1~B2.1)

Table 9: lm(B5.1~B2.1) | B2.1: +0.2

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.3494	0.0283	47.6337	0
B2.1	0.5521	0.0096	57.5786	0

- **True effect: +0.2**
- Confidence Interval $= \overline{X} \pm t_{\alpha/2,N-1} * S_{\overline{X}}$
- **lower.bound** = 0.552082 - 0.0095883 * 1.96 = **0.533**
- **upper.bound** = 0.552082 + 0.0095883 * 1.96 = **0.571**

-> True effect is **not covered** by the confidence interval

B.II Check if confidence interval encloses true effect (B5.1~Alter)

Table 10: lm(B5.1~Alter) | Alter: -0.3

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	3.2286	0.0251	128.6139	0
Alter	-0.1088	0.0076	-14.3234	0

- **True effect: -0.3**
- Confidence Interval $= \overline{X} \pm t_{\alpha/2,N-1} * S_{\overline{X}}$
- **lower.bound** = -0.1087938 - 0.0075955 * 1.96 = **-0.124**
- **upper.bound** = -0.1087938 + 0.0075955 * 1.96 = **-0.094**

-> True effect is **not covered** by the confidence interval

Brainstroming

What's the problem? Why can't we estimate the true (causal) effect?

Excursus: Modern Causal Analysis

- Satisfaction of the **Conditional Independence Assumption (CIA)** necessary to estimate true causal effects
- Meet the CIA using an **appropriate set of control variables**
- Choose control variables by a **Directed Acyclic Graph (DAG)**

Excursus MCA: DAG (B5.1 <- FK)

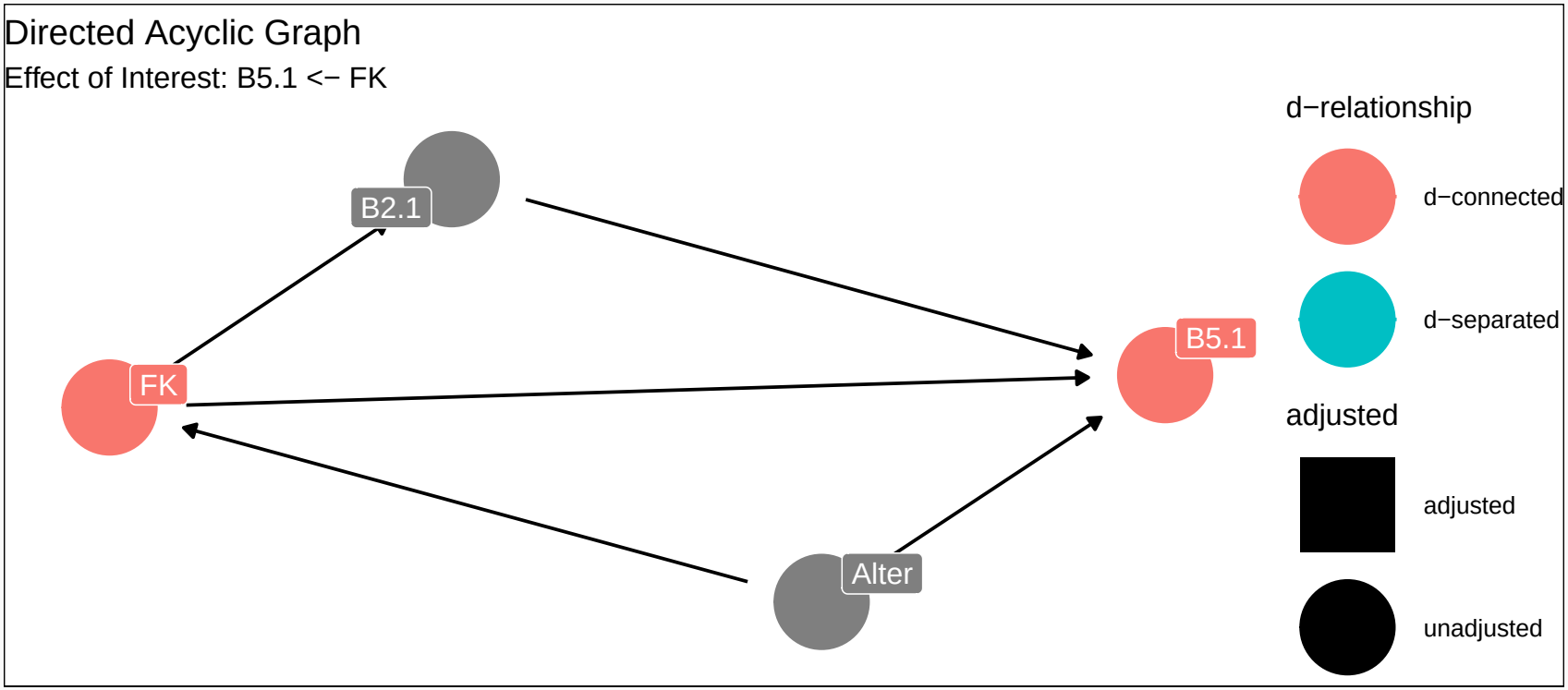


Table 11: lm(B5.1~FK+B2.1+Alter) | FK: +1.5

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.9909	0.0311	96.2506	0
FK	1.5033	0.0283	53.1769	0

```
## [1] "Effect without fulfilling the CIA: 1.4519"
```

Excursus MCA: DAG (B5.1 <- B2.1)

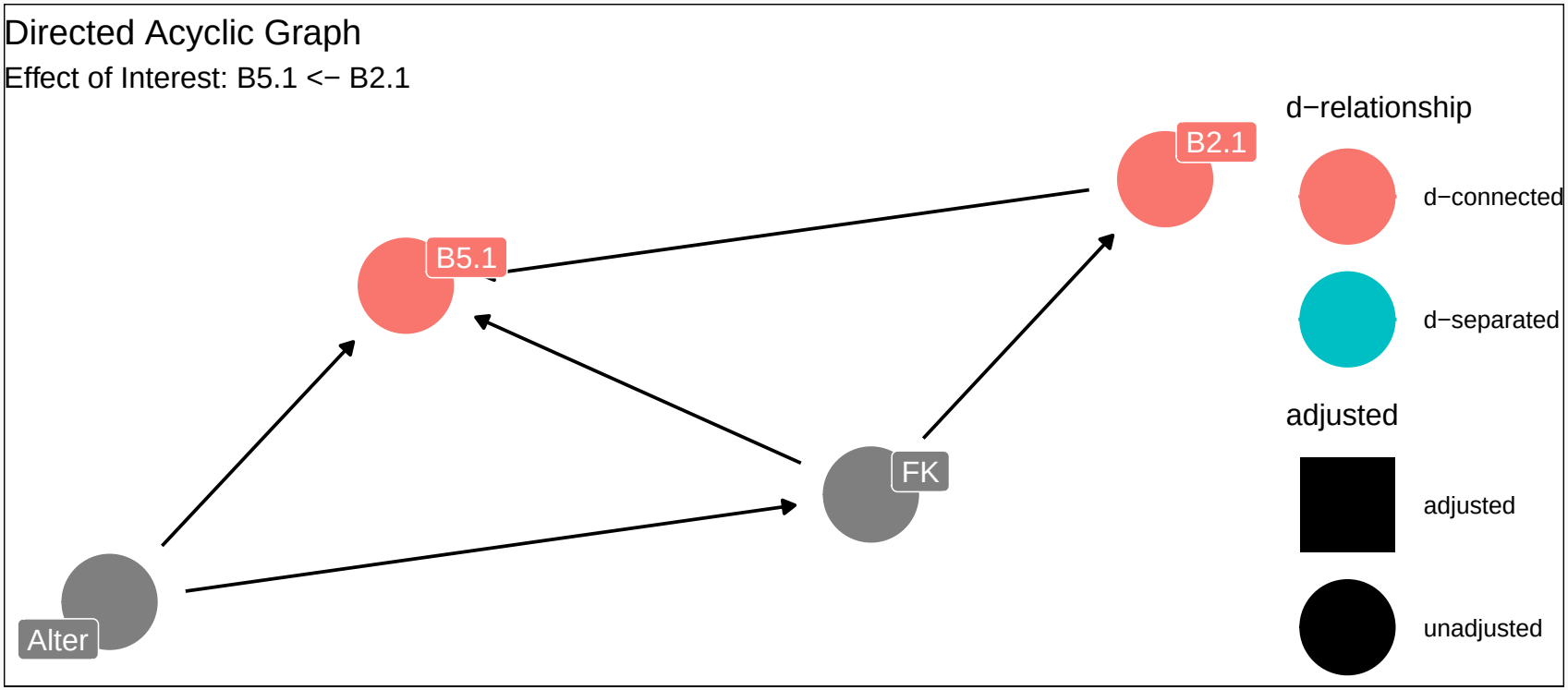


Table 12: lm(B5.1~B2.1+FK+Alter) | B2.1: +0.2

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.9909	0.0311	96.2506	0
B2.1	0.2031	0.0112	18.0842	0

```
## [1] "Effect without fulfilling the CIA: 0.5521"
```

Excursus MCA: DAG (B5.1 <- Alter)

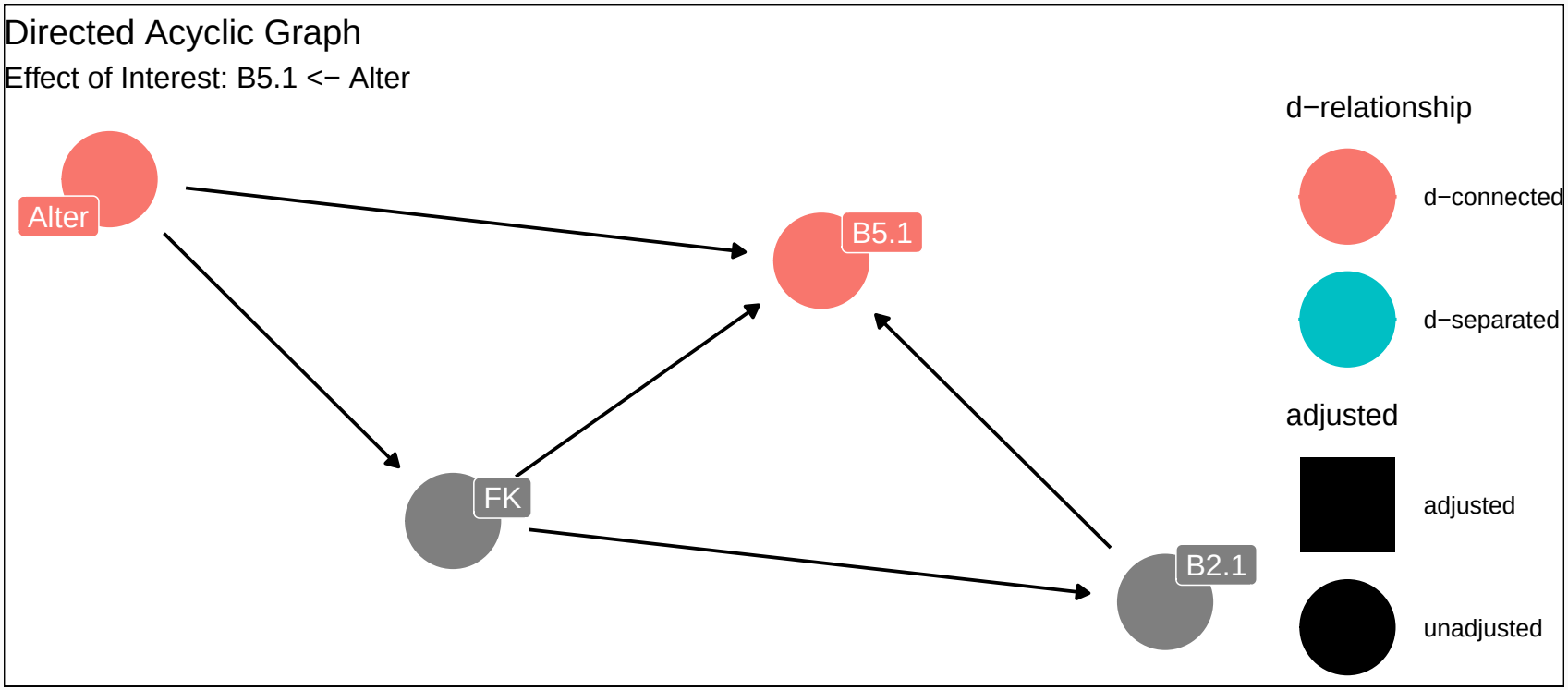


Table 13: lm(B5.1~Alter+FK+B2.1) | Alter: -0.3

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.9909	0.0311	96.2506	0
Alter	-0.3007	0.0044	-68.9450	0

```
## [1] "Effect without fulfilling the CIA: -0.1088"
```

Conclusion on the calculation of the unbiased effect

- Even in case of non-missing data the estimation of unbiased estimates is not a sure-fire success
- The CIA needs to be satisfied otherwise there will be **always a bias** in the estimates
- Fulfilling the CIA is possible by deriving an appropriate set of variables from an **DAGs**

Missing Patterns

Three types of missing patterns

- Missing completely at random (MCAR)
- Missing Patterns: Missing at random (MAR)
- Missing Patterns: Missing not at random (MNAR)

Missing Patterns: Missing completely at random (MCAR)

- Missing values are **completely randomly distributed** in the variable of interest
  - There is no pattern to the actual values of the missing variable themselves
    - *e.g. higher or lower values are more likely be missing*
  - Missing data values do not relate to any distribution of another variable
    - *e.g. woman are more likely to have missing values on income*
- For instance, when smoking status is not recorded in a random subset of patients
- This missing-type is easy to handle, but unfortunately, data are **almost never** missing completely at random

MCAR: Inserting Missing values in data frame

```r
if(!require(missMethods)) install.packages("missMethods")
library(missMethods)

set.seed(322)

df.MCAR = delete_MCAR(df, 0.3, "Mann")

df.MCAR = delete_MCAR(df.MCAR, 0.2, "Alter")

df.MCAR = delete_MCAR(df.MCAR, 0.25, "FK")

df.MCAR = delete_MCAR(df.MCAR, 0.15, "B2.1")

df.MCAR = delete_MCAR(df.MCAR, 0.35, "B5.1")
```

MCAR: Check missing pattern (Mann)

```
## [1] "Mean 'Mann' (no missings): 0.493"
```

```
## [1] "Mean 'Mann' (MCAR): 0.4934"
```

```
##
##     FALSE TRUE
##   0  1773  762
##   1  1727  738
```

```
##
##           FALSE   TRUE
##   0        50.7   50.8
##   1        49.3   49.2
##   Total   100.0  100.0
##   Count  3500.0 1500.0
```

MCAR: Check missing pattern (Alter)

```
## [1] "Mean 'Alter' (no missings): 2.9816"
```

```
## [1] "Mean 'Alter' (MCAR): 2.988"
```

```
##
##     FALSE TRUE
##   1   843  208
##   2   775  200
##   3   777  213
##   4   797  186
##   5   808  193
```

```
##
##          FALSE   TRUE
##   1       21.1   20.8
##   2       19.4   20.0
##   3       19.4   21.3
##   4       19.9   18.6
##   5       20.2   19.3
##   Total  100.0  100.0
##   Count 4000.0 1000.0
```

MCAR: Check missing pattern (FK)

```
## [1] "Mean 'FK' (no missings): 0.1582"
```

```
## [1] "Mean 'FK' (MCAR): 0.1568"
```

```
##
##     FALSE TRUE
##   0  3162 1047
##   1   588  203
```

```
##
##           FALSE   TRUE
##   0        84.3   83.8
##   1        15.7   16.2
##   Total   100.0  100.0
##   Count  3750.0 1250.0
```

MCAR: Check missing pattern (FK II)

```r
#define standard error of mean function
  std.error <- function(x) sd(x[!is.na(x)])/sqrt(length(x[!is.na(x)]))

#Calculate the confidence interval
  mean(df.MCAR$FK, na.rm = TRUE) + 1.96*std.error(df.MCAR$FK) # upper.bound
```

```
## [1] 0.1684396
```

```r
  mean(df.MCAR$FK, na.rm = TRUE) - 1.96*std.error(df.MCAR$FK) # lower.bound
```

```
## [1] 0.1451604
```

MCAR: Check missing pattern (B2.1)

```
## [1] "Mean 'B2.1' (no missings): 2.8162"
```

```
## [1] "Mean 'B2.1' (MCAR): 2.8169"
```

```
##
##     FALSE TRUE
##   1    10    4
##   2  1783  311
##   3  1776  317
##   4   337   58
##   5   344   60
```

```
##
##          FALSE  TRUE
##   1        0.2   0.5
##   2       42.0  41.5
##   3       41.8  42.3
##   4        7.9   7.7
##   5        8.1   8.0
##   Total  100.0 100.0
##   Count 4250.0 750.0
```

MCAR: Check missing pattern (B5.1)

```
## [1] "Mean 'B5.1' (no missings): 2.9042"
```

```
## [1] "Mean 'B5.1' (MCAR): 2.8972"
```

```
##
##     FALSE TRUE
##   1    25   23
##   2   990  506
##   3  1626  865
##   4   512  305
##   5    97   51
```

```
##
##          FALSE   TRUE
##   1        0.8    1.3
##   2       30.5   28.9
##   3       50.0   49.4
##   4       15.8   17.4
##   5        3.0    2.9
##   Total  100.1   99.9
##   Count 3250.0 1750.0
```

Missing Patterns: Missing at random (MAR)

- Confusing and would be better stated as *missing conditionally at random*
- Missing data do have a relationship with other variables in the dataset
  - Whether a value is missing or not depends on other variables
  - *e.g. woman are more likely to have missing values on income*
- The actual shaping of the missing-values are random
  - *e.g. the missing income data of woman are not especially low or high, but even distributed*
- For example, smoking status is not documented in female patients because the doctor was too shy to ask

MAR: Inserting Missing values in data frame

```r
if(!require(missMethods)) install.packages("missMethods")
library(missMethods)


# Generate MAR values using a censoring mechanism. This leads to a missing value in "X", if
  # ds_mar <- delete_MAR_censoring(ds_comp, 0.3, "X", cols_ctrl = "Y")

#The censoring mechanism is a rather strong form of MAR. A function that allows to control
  # ds_mar <- delete_MAR_1_to_x(ds_comp, 0.3, "X", cols_ctrl = "Y", x = 2)

set.seed(322)

df.MAR = delete_MAR_1_to_x(df, 0.3, "Mann", cols_ctrl = "Alter", x = 2)

df.MAR = delete_MAR_1_to_x(df.MAR, 0.4, "Alter", cols_ctrl = "FK", x = 3)

df.MAR = delete_MAR_1_to_x(df.MAR, 0.7, "FK", cols_ctrl = "B2.1", x = 2.5)

df.MAR = delete_MAR_1_to_x(df.MAR, 0.15, "B2.1", cols_ctrl = "B5.1", x = 2)

df.MAR = delete_MCAR(df.MAR, 0.30, "B5.1")
```

MAR: Check missing pattern (Mann)

```
## [1] "Mean 'Mann' (no missings): 0.493"
```

```
## [1] "Mean 'Mann' (MAR): 0.4871"
```

```
##
##     FALSE TRUE
##   0  1795  740
##   1  1705  760
```

```
##
##           FALSE   TRUE
##   0        51.3   49.3
##   1        48.7   50.7
##   Total   100.0  100.0
##   Count  3500.0 1500.0
```

MAR: Check missing pattern (Alter)

```
## [1] "Mean 'Alter' (no missings): 2.9816"
```

```
## [1] "Mean 'Alter' (MAR): 2.797"
```

```
##
##     FALSE TRUE
##   1   726  325
##   2   637  338
##   3   625  365
##   4   544  439
##   5   468  533
```

```
##
##           FALSE   TRUE
##   1        24.2   16.2
##   2        21.2   16.9
##   3        20.8   18.2
##   4        18.1   22.0
##   5        15.6   26.7
##   Total    99.9  100.0
##   Count  3000.0 2000.0
```

MAR: Check missing pattern (FK)

```
## [1] "Mean 'FK' (no missings): 0.1582"
```

```
## [1] "Mean 'FK' (MAR): 0.0307"
```

```
##
##     FALSE TRUE
##   0  1454 2755
##   1    46  745
```

```
##
##           FALSE   TRUE
##   0        96.9   78.7
##   1         3.1   21.3
##   Total   100.0  100.0
##   Count  1500.0 3500.0
```

MAR: Check missing pattern (B2.1)

```
## [1] "Mean 'B2.1' (no missings): 2.8162"
```

```
## [1] "Mean 'B2.1' (MAR): 2.8082"
```

```
##
##     FALSE TRUE
##   0  1454 2755
##   1    46  745
```

```
##
##           FALSE  TRUE
##   1         0.3   0.3
##   2        42.3  39.6
##   3        41.6  43.2
##   4         8.0   7.6
##   5         7.9   9.3
##   Total   100.1 100.0
##   Count  4250.0 750.0
```

MAR: Check missing pattern (B5.1)

- Due to estimation reasons B5.1 is MCAR
- Missing patterns is the as shown before

Missing Patterns: Missing not at random (MNAR)

- The pattern of missingness is related to other variables in the dataset
- In addition, the shaping of the missing-values are **NOT** random
  - Whether a value is missing or not depends on its shaping
  - *e.g. higher or lower incomes are more likely be missing*
- For example, when smoking status is not recorded in patients admitted as an emergency, who are also more likely to have worse outcomes from surgery

MNAR: Inserting Missing values in data frame

```r
if(!require(missMethods)) install.packages("missMethods")
library(missMethods)


# Generate MNAR values using a censoring mechanism. This leads to a missing value in "X", i
  #ds_mnar <- delete_MNAR_censoring(ds_comp, 0.3, "X")

# Create missing not at random (MNAR) values using MNAR1:x in a data frame
  #delete_MNAR_1_to_x(ds, 0.2, "X", x = 3)

set.seed(322)

df.MNAR = delete_MNAR_1_to_x(df, 0.3, "Mann", x = 3)

df.MNAR = delete_MNAR_1_to_x(df.MNAR, 0.4, "Alter", x = 3.5)

df.MNAR = delete_MNAR_1_to_x(df.MNAR, 0.45, "FK", x = 2)

df.MNAR = delete_MNAR_1_to_x(df.MNAR, 0.35, "B2.1", x = 2.5)

df.MNAR = delete_MNAR_1_to_x(df.MNAR, 0.30, "B5.1", x = 4)
```

MNAR: Check missing pattern (Mann)

```
## [1] "Mean 'Mann' (no missings): 0.493"
```

```
## [1] "Mean 'Mann' (MNAR): 0.3851"
```

```
##
##     FALSE TRUE
##   0  2152  383
##   1  1348 1117
```

```
##
##           FALSE   TRUE
##   0        61.5   25.5
##   1        38.5   74.5
##   Total   100.0  100.0
##   Count  3500.0 1500.0
```

MNAR: Check missing pattern (Alter)

```
## [1] "Mean 'Alter' (no missings): 2.9816"
```

```
## [1] "Mean 'Alter' (MNAR): 2.571"
```

```
##
##     FALSE TRUE
##   1   883  168
##   2   817  158
##   3   437  553
##   4   430  553
##   5   433  568
```

```
##
##           FALSE   TRUE
##   1        29.4    8.4
##   2        27.2    7.9
##   3        14.6   27.7
##   4        14.3   27.7
##   5        14.4   28.4
##   Total    99.9  100.1
##   Count  3000.0 2000.0
```

MNAR: Check missing pattern (FK)

```
## [1] "Mean 'FK' (no missings): 0.1582"
```

```
## [1] "Mean 'FK' (MNAR): 0.064"
```

```
##
##     FALSE TRUE
##   0  2574 1635
##   1   176  615
```

```
##
##           FALSE   TRUE
##   0        93.6   72.7
##   1         6.4   27.3
##   Total   100.0  100.0
##   Count  2750.0 2250.0
```

MNAR: Check missing pattern (B2.1)

```
## [1] "Mean 'B2.1' (no missings): 2.8162"
```

```
## [1] "Mean 'B2.1' (MNAR): 2.6662"
```

```
##
##     FALSE TRUE
##   1    12    2
##   2  1701  393
##   3  1105  988
##   4   224  171
##   5   208  196
```

```
##
##          FALSE   TRUE
##   1        0.4    0.1
##   2       52.3   22.5
##   3       34.0   56.5
##   4        6.9    9.8
##   5        6.4   11.2
##   Total  100.0  100.1
##   Count 3250.0 1750.0
```

MNAR: Check missing pattern (B5.1)

```
## [1] "Mean 'B5.1' (no missings): 2.9042"
```

```
## [1] "Mean 'B5.1' (MNAR): 2.7774"
```

```
##
##     FALSE TRUE
##   1    43    5
##   2  1350  146
##   3  1534  957
##   4   489  328
##   5    84   64
```

```
##
##          FALSE   TRUE
##   1        1.2    0.3
##   2       38.6    9.7
##   3       43.8   63.8
##   4       14.0   21.9
##   5        2.4    4.3
##   Total  100.0  100.0
##   Count 3500.0 1500.0
```

Missing data handling techniques

Overview of missing data handling techniques

Traditional Methods

- Delete Rows with Missing Values (list-wise deletion)
- Mean/Median Imputation
- Regression Imputation

Modern Methods

- Multiple Imputation

Missingness	List-wise deletion	Mean/Median Imputation	Regression Imputation	Multiple Imputation
MCAR	unbiased estimate \ (biases SE)	unbiased estimate \ (biases SE)	unbiased estimate \ (biases SE)	unbiased estimate \ (**unbiased** SE)
MAR	biased estimate \ (biases SE)	biased estimate \ (biases SE)	unbiased estimate \ (biases SE)	unbiased estimate \ (**unbiased** SE)
MNAR	biased estimate \ (biases SE)	biased estimate \ (biases SE)	unbiased estimate \ (biases SE)	unbiased estimate \ (**unbiased** SE)

MCAR: list-wise deletion (Point estimate Mann) CODE

```r
# Point estimates
  paste0("Mean 'Mann' (no missings): ", mean(df$Mann))
  paste0("Mean 'Mann' (MCAR): ", round(mean(df.MCAR$Mann, na.rm = TRUE),4))

  #define standard error of mean function
    std.error <- function(x) sd(x[!is.na(x)])/sqrt(length(x[!is.na(x)]))

  #Calculate the confidence interval
    paste0("upper.bound: ", round(mean(df.MCAR$Mann, na.rm = TRUE) + 1.96*
                                    std.error(df.MCAR$Mann), 4)) # upper.bound
    paste0("lower.bound: ", round(mean(df.MCAR$Mann, na.rm = TRUE) - 1.96*
                                    std.error(df.MCAR$Mann), 4)) # lower.bound
```

MCAR: list-wise deletion (Point estimate Mann)

Full Dataset (true coefficients)

```
## [1] "Mean 'Mann' (no missings): 0.493"
```

```
## [1] "upper.bound: 0.5069"
```

```
## [1] "lower.bound: 0.4791"
```

Missing Dataset

```
## [1] "Mean 'Mann' (MCAR): 0.4934"
```

```
## [1] "upper.bound: 0.51"
```

```
## [1] "lower.bound: 0.4769"
```

Evaluation: Unbiased Estimate (**biases** SE & Confidence Interval)

MCAR: Mean/Median Imputation (Point estimate Mann) CODE

```r
# Mean Imputation
  df.MCAR.Mean = df.MCAR
  df.MCAR.Mean$Mann <- ifelse(is.na(df.MCAR$Mann), mean(df.MCAR$Mann, na.rm = TRUE),
                              df.MCAR$Mann)

# Point estimates
  paste0("Mean 'Mann' (no missings): ", mean(df$Mann))
  paste0("Mean 'Mann' (MCAR): ", round(mean(df.MCAR.Mean$Mann, na.rm = TRUE),4))

  #define standard error of mean function
    std.error <- function(x) sd(x[!is.na(x)])/sqrt(length(x[!is.na(x)]))

  #Calculate the confidence interval
    paste0("upper.bound: ", round(mean(df.MCAR.Mean$Mann, na.rm = TRUE) + 1.96*
                                  std.error(df.MCAR.Mean$Mann), 4)) # upper.bound
    paste0("lower.bound: ", round(mean(df.MCAR.Mean$Mann, na.rm = TRUE) - 1.96*
                                  std.error(df.MCAR.Mean$Mann), 4)) # lower.bound
```

MCAR: Mean/Median Imputation (Point estimate Mann)

Full Dataset (true coefficients)

```
## [1] "Mean 'Mann' (no missings): 0.493"
```

```
## [1] "upper.bound: 0.5069"
```

```
## [1] "lower.bound: 0.4791"
```

Missing Dataset

```
## [1] "Mean 'Mann' (MCAR): 0.4934"
```

```
## [1] "upper.bound: 0.505"
```

```
## [1] "lower.bound: 0.4818"
```

Evaluation: Unbiased Estimate (**biases** SE & Confidence Interval)

Excursus: Rules for MI inference ("Rubin's rules")

Literature-Link

Combining parameter estimates

For a single population parameter of interest, Q, e.g. a regression coefficient, the MI overall point estimate is the average of the **m**-estimates of Q from the imputed datasets: $\overline{Q} = \frac{1}{m} \sum_{i=1}^{m} \hat{Q}$

The **standard error** is calculated with the **total variance**(TIV), which is uses the **within imputation variance**(WIV) and the **between imputation variance**(BIV):

$WIV = \frac{1}{m} \sum_{i=1}^{m} \sigma^2$, where $\sigma^2 = \frac{\sum (X_i - \mu)^2}{N}$

$BIV = \frac{1}{m} \sum_{i=1}^{m} (\hat{Q}_i - \overline{Q})$

$TIV = WIV + (1 + \frac{1}{m}) BIW$

Now the standard error is calculated by $SD = \sqrt{TIV}$ and $SE = \frac{SD}{\sqrt{N/m}}$

Testing “Rubin’s rules” code

```r
#Create artificial mi-dataset (no missings)
  df.mi = rbind(df,df,df,df,df) %>% mutate(mi_id = rep(1:5, each = 5000))
  n_mi = length(unique(df.mi$mi_id))
#Calculate and compare point-estimates
  round(mean(df$Mann),4)
  round(mean(df.mi$Mann),4)
#Calculate and compare standard errors & confidence intervals
  #1.Step: Calculate the total variance by the "within imputation variance" and the "between imputation variance"
  #2.Step: Calculate the SE from the "total variance"
    WIV = df.mi %>% group_by(mi_id) %>% summarise(var = var(Mann)) %>% summarise(var = mean(var)) %>% .[[1]]

    BIV = df.mi %>% group_by(mi_id) %>% summarise(mean = mean(Mann)) %>% mutate(diff.square = (mean - colMeans(.[2]))^2) %>%
      summarise(B = mean(diff.square)) %>% .[[1]]

    TIV = WIV+(1+1/5)*BIV
    SD = sqrt(TIV)
    SE = SD/sqrt(nrow(df.mi)/5)

  # Alternative: directly use SE as within-component
    W = df.mi %>% group_by(mi_id) %>% summarise(std.error = sd(Mann[!is.na(Mann)])/sqrt(length(Mann[!is.na(Mann)]))) %>%
      summarise(std.error = mean(std.error)) %>%
      .[[1]]

    B = df.mi %>% group_by(mi_id) %>% summarise(mean = mean(Mann)) %>% mutate(diff.square = (mean - colMeans(.[2]))^2) %>%
      summarise(B = mean(diff.square)) %>% .[[1]]
    SE.short = W+(1+1/n_mi)*B


  #Alternative: Neglect between-component
    #define standard error of mean function
      mi_std.error <- function(x) sd(x[!is.na(x)])/sqrt(length(x[!is.na(x)])/n_mi) # UNbiased!!!!

  #Define standard error without considering MI-structure (WRONG)
    std.error <- function(x) sd(x[!is.na(x)])/sqrt(length(x[!is.na(x)])) #Biased!!!!
```

Testing “Rubin’s rules” code

Mean Calculation:

```
## [1] "Mean (Full-Dataset): 0.493"
```

```
## [1] "Mean (MI-Dataset): 0.493"
```

SE Calculation:

```
## [1] "SE (Not considering MI-structure): 0.00316203098302302"
```

```
## [1] "SE (No between effect): 0.0070705162249995"
```

```
## [1] "SE (Rubin's rule, SE indead of VAR): 0.00707108195681594
```

```
## [1] "SE (Rubin's rule): 0.0077453624592749"
```

MCAR: Multiple Imputation (Point estimate Mann) CODE

```r
if(!require(mice)) install.packages("mice")
library(mice)

# Multiple Imputation
  df.MCAR.MI <- mice(df.MCAR,m=5,maxit=50,meth='pmm',seed=500, print=FALSE) # Object of mice class, can be used with mice-speci
  df.MCAR.MI <- complete(df.MCAR.MI, 'long') # create dataframe of the n-imputed datasets

# Point estimates
  paste0("Mean 'Mann' (MCAR): ", round(mean(df.MCAR.MI$Mann),4))

#Define function for SE-calculation
  mi_se = function(data, var, mi_id) {
    WIV = data %>%
      group_by(eval(parse(text =mi_id))) %>%
      summarise(variance = var(eval(parse(text=var)))) %>%
      summarise(var = mean(variance)) %>%
      .[[1]]
    BIV = data %>%
      group_by(eval(parse(text =mi_id))) %>%
      summarise(mean = mean(eval(parse(text=var)))) %>%
      mutate(diff.square = (mean - colMeans(.[2]))^2) %>%
      summarise(B = mean(diff.square)) %>%
      .[[1]]
    TIV = WIV+(1+1/length(unique(data[,mi_id])))*BIV
    SD = sqrt(TIV)
    SE = SD/sqrt(nrow(data)/length(unique(data[,mi_id])))
    return(SE)
  }

#Using the function
  mi_se(df.mi,"Mann", "mi_id")
```

MCAR: Multiple Imputation (Point estimate Mann)

Full Dataset (true coefficients)

```
## [1] "Mean 'Mann' (no missings): 0.493"
```

```
## [1] "upper.bound: 0.5069"
```

```
## [1] "lower.bound: 0.4791"
```

Missing Dataset

```
## [1] "Mean 'Mann' (MCAR): 0.4952"
```

```
## [1] "upper.bound (wrong): 0.5014"
```

```
## [1] "lower.bound (wrong): 0.489"
```

```
## [1] "upper.bound (true): 0.5091"
```

```
## [1] "lower.bound (true): 0.4814"
```

Evaluation: Unbiased Estimate (unbiases SE & Confidence Interval)

MAR: Multiple Imputation (Point estimate Mann)

Full Dataset (true coefficients)

```
## [1] "Mean 'Mann' (no missings): 0.493"
```

```
## [1] "upper.bound: 0.5069"
```

```
## [1] "lower.bound: 0.4791"
```

Missing Dataset

```
## [1] "Mean 'Mann' (MAR): 0.4973"
```

```
## [1] "upper.bound (wrong): 0.5035"
```

```
## [1] "lower.bound (wrong): 0.4911"
```

```
## [1] "upper.bound (true): 0.5111"
```

```
## [1] "lower.bound (true): 0.4834"
```

Evaluation: Unbiased Estimate (**unbiases SE** & Confidence Interval)

MNAR: Multiple Imputation (Point estimate Mann)

Full Dataset (true coefficients)

```
## [1] "Mean 'Mann' (no missings): 0.493"
```

```
## [1] "upper.bound: 0.5069"
```

```
## [1] "lower.bound: 0.4791"
```

Missing Dataset

```
## [1] "Mean 'Mann' (MNAR): 0.3913"
```

```
## [1] "upper.bound (wrong): 0.3974"
```

```
## [1] "lower.bound (wrong): 0.3853"
```

```
## [1] "upper.bound (true): 0.4049"
```

```
## [1] "lower.bound (true): 0.3778"
```

Evaluation: Unbiased Estimate (**unbiases SE** & Confidence Interval)

Questions

Questions

Any Questions so far?