

EP16: Missing Values in Clinical Research: Multiple Imputation

6. Imputation with mice

Nicole S. Erler

Department of Biostatistics, Erasmus Medical Center

✉ n.erler@erasmusmc.nl

Main Function Arguments

The main function for imputation with the R package **mice** is `mice()`. Its most important arguments are:

- ▶ `data`: the dataset

Main Function Arguments

The main function for imputation with the R package **mice** is `mice()`. Its most important arguments are:

- ▶ `data`: the dataset
- ▶ `m`: number of imputed datasets

Main Function Arguments

The main function for imputation with the R package **mice** is `mice()`. Its most important arguments are:

- ▶ `data`: the dataset
- ▶ `m`: number of imputed datasets
- ▶ `maxit`: number of iterations

Main Function Arguments

The main function for imputation with the R package **mice** is `mice()`. Its most important arguments are:

- ▶ `data`: the dataset
- ▶ `m`: number of imputed datasets
- ▶ `maxit`: number of iterations
- ▶ `method`: vector of imputation methods

Main Function Arguments

The main function for imputation with the R package **mice** is `mice()`. Its most important arguments are:

- ▶ `data`: the dataset
- ▶ `m`: number of imputed datasets
- ▶ `maxit`: number of iterations
- ▶ `method`: vector of imputation methods
- ▶ `defaultMethod`: vector of default imputation methods

Main Function Arguments

The main function for imputation with the R package **mice** is `mice()`. Its most important arguments are:

- ▶ `data`: the dataset
- ▶ `m`: number of imputed datasets
- ▶ `maxit`: number of iterations
- ▶ `method`: vector of imputation methods
- ▶ `defaultMethod`: vector of default imputation methods
- ▶ `predictorMatrix`: matrix specifying roles of variables

Iterations and Imputations

By default, **mice** will use

- ▶ `maxit` = 5 (i.e., the algorithm is run for 5 iterations)
- ▶ `m` = 5 (i.e., 5 imputed datasets will be created)

Often, these values need to be larger, but we will come back to this later in the course.

Imputation Methods

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

- ▶ **pmm**: predictive mean matching (any)

Imputation Methods

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

- ▶ **pmm**: predictive mean matching (any)
- ▶ **norm**: Bayesian linear regression (numeric)

Imputation Methods

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

- ▶ **pmm**: predictive mean matching (any)
- ▶ **norm**: Bayesian linear regression (numeric)
- ▶ **logreg**: binary logistic regression (binary)

Imputation Methods

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

- ▶ **pmm**: predictive mean matching (any)
- ▶ **norm**: Bayesian linear regression (numeric)
- ▶ **logreg**: binary logistic regression (binary)
- ▶ **polr**: proportional odds model (ordered factors)

Imputation Methods

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

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

Imputation Methods

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

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

Imputation methods are chosen **automatically based on the type of variable**.

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

Imputation Methods: defaultMethod

The default setting for the argument `defaultMethod` is:

```
defaultMethod = c("pmm", "logreg", "polyreg", "polr")
```

i.e.,

- ▶ `pmm` for numerical variables
- ▶ `logreg` for binary variables
- ▶ `polr` for unordered factors with > 2 levels
- ▶ `polyreg` for ordered factors

Imputation Methods: defaultMethod

The default setting for the argument `defaultMethod` is:

```
defaultMethod = c("pmm", "logreg", "polyreg", "polr")
```

i.e.,

- ▶ `pmm` for numerical variables
- ▶ `logreg` for binary variables
- ▶ `polr` for unordered factors with > 2 levels
- ▶ `polyreg` for ordered factors

Example:

To use `norm` instead of `pmm` for all continuous incomplete variables:

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


Tip: Use a Setup Run

To obtain a default version for the arguments like `method` and `predictorMatrix` it is convenient to **do a setup run** of `mice()` without iterations:

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

Tip: Use a Setup Run

To obtain a default version for the arguments like `method` and `predictorMatrix` it is convenient to **do a setup run** of `mice()` without iterations:

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

The function `mice()` returns an object of class **mids** (multiply imputed data set).

Tip: Use a Setup Run

To obtain a default version for the arguments like `method` and `predictorMatrix` it is convenient to **do a setup run** of `mice()` without iterations:

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

The function `mice()` returns an object of class **mids** (multiply imputed data set).

The default versions can then be extracted and modified, for example:

```
meth <- imp0$method
```

Imputation Methods: `method`

`meth`

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

Imputation Methods: `method`

```
meth
```

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

Variables that do **not need to be imputed** (= do not have any missing values) are set to "".

To change the imputation method for single variables, the vector `meth` can then be adapted:

```
meth["albu"] <- "norm"
```

Imputation Methods: `method`

For variables that **should not be imputed** the imputation method can be set to `""`:

```
meth["HyperMed"] <- ""
```

For the actual imputation, `mice()` is called using the adapted argument(s):

```
imp1 <- mice(NHANES, method = meth)
```

Predictor Matrix

The `predictorMatrix` is a matrix that specifies **which variables are used as predictors** in which imputation model.

Each row represents the model for the variable given in the row name.

```
head(imp0$predictorMatrix)[, 1:11] # subset, to fit on the slide
```

##	age	gender	race	bili	chol	HDL	hypten	BMI	hypchol	DM	smoke
## 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 predictors.

Predictor Matrix

Important:

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

Predictor Matrix

Important:

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

Example:

- ▶ We set `meth["HyperMed"] = ""`
 - ➔ `HyperMed` will not be imputed
- ▶ `HyperMed` is still used as predictor variable (default setting of `predicorMatrix`)
- ▶ for cases with missing `HyperMed` none of the other variables can be imputed

Predictor Matrix

Important:

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

Example:

- ▶ We set `meth["HyperMed"] = ""`
 - ➔ `HyperMed` will not be imputed
- ▶ `HyperMed` is still used as predictor variable (default setting of `predicorMatrix`)
- ▶ for cases with missing `HyperMed` none of the other variables can be imputed

➔ We also have to set

```
pred <- imp0$predictorMatrix
pred[, 'HyperMed'] <- 0
```

A Quick Summary

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

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

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

# run imputation with adjusted settings
imp1 <- mice(NHANES, method = meth, predictorMatrix = pred)
```

Passive Imputation

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

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

Passive Imputation

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

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

```
DF1 <- complete(imp1, 1) # select the first imputed dataset

round(cbind(BMI = DF1$BMI,                                     # imputed
            "wgt/hgt^2" = DF1$weight/DF1$height^2           # calculated
), 2)[which(is.na(NHANES$BMI))[1:3], ]
```

Passive Imputation

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

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

```
DF1 <- complete(imp1, 1) # select the first imputed dataset

round(cbind(BMI = DF1$BMI,                                     # imputed
            "wgt/hgt^2" = DF1$weight/DF1$height^2           # calculated
), 2)[which(is.na(NHANES$BMI))[1:3], ]
```

```
##      BMI wgt/hgt^2
## [1,] 24.24    22.60
## [2,] 23.99    24.96
## [3,] 27.53    27.60
```

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

Passive Imputation

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

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

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

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

Passive Imputation

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

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

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

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

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

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***:

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)"
```

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)"
```

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

```
pred[c("weight", "height"), "BMI"] <- 0
```

Passive Imputation

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

Which one to use may differ between imputation models.

Passive Imputation

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

Which one to use may differ between imputation models.

Passive imputation can also be useful in settings where

- ▶ imputation models include **interaction terms** between incomplete variables (see Van Buuren (2012), p. 133 for an example), or when

Passive Imputation

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

Which one to use may differ between imputation models.

Passive imputation can also be useful in settings where

- ▶ imputation models include **interaction terms** between incomplete variables (see Van Buuren (2012), 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.

Post Processing

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

Post Processing

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

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.  
## -0.1254  0.7000  0.8300  0.8862  0.9900  9.5100
```


Post Processing

A helpful function:

- ▶ `mice::squeeze()` to censor variables at given boundaries

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 <- imp1$post
post["creat"] <- "imp[[j]][,i] <- squeeze(imp[[j]][,i], c(0, 100))"
```

Post Processing

A helpful function:

- ▶ `mice::squeeze()` to censor variables at given boundaries

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 <- imp1$post
post["creat"] <- "imp[[j]][,i] <- squeeze(imp[[j]][,i], c(0, 100))"
```

```
imp2 <- update(imp1, post = post)
```

Post Processing

A helpful function:

- ▶ `mice::squeeze()` to censor variables at given boundaries

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 <- imp1$post
post["creat"] <- "imp[[j]][,i] <- squeeze(imp[[j]][,i], c(0, 100))"
```

```
imp2 <- update(imp1, post = post)
```

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 complete data distribution**.

Post Processing

The **post-processing** functionality allows for many **more data manipulations** and is not restricted to `squeeze()`.

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

Post Processing

The **post-processing** functionality allows for many **more data manipulations** and is not restricted to `squeeze()`.

Any string 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]][data[where[, j], 'SBP'] > 140, i] <- 'yes'"
```

Post Processing

The **post-processing** functionality allows for many **more data manipulations** and is not restricted to `squeeze()`.

Any string 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]][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.

Visit Sequence

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

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

Visit Sequence

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

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

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.

Visit Sequence

Example:

Currently, BMI is imputed before height and weight:

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

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

To get consistent values, we need to change the visitSequence:

```
visitSeq <- c(visitSeq[-which(visitSeq == "BMI")], # everything else  
              "BMI")                               # BMI
```

Visit Sequence

- ▶ By default `mice()` imputes in the **order of the columns in the** data.
- ▶ The `visitSequence` may specify that a column is **visited multiple times** during one iteration.
- ▶ All incomplete variables must be **visited at least once**.

Automated Changes

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

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

Automated Changes

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

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

In each iteration

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

Why?

To avoid problems in the imputation models.

Automated Changes

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.

Automated Changes

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

A note

*“Please realize that these choices are always needed. Imputation software needs to make default choices. These choices are intended to be useful across a wide range of applications. However, the **default choices are not necessarily the best for the data at hand. There is simply no magical setting that always works,** so often some tailoring is needed.” (Van Buuren 2012, p.124)*

Your Turn!

Practical

Imputation with mice [html](#)

References

Van Buuren, Stef. 2012. *Flexible Imputation of Missing Data*. Chapman & Hall/Crc Interdisciplinary Statistics. Taylor & Francis.
<https://stefvanbuuren.name/fimd/>.