*Journal of Statistical Software*

*December 2011, Volume 45, Issue 3.* *http://www.jstatsoft.org/*

# mice: Multivariate Imputation by Chained Equations in R

**Stef van Buuren**
TNO

**Karin Groothuis-Oudshoorn**
University of Twente

**Abstract**

The R package **mice** imputes incomplete multivariate data by chained equations. The software **mice** 1.0 appeared in the year 2000 as an S-PLUS library, and in 2001 as an R package. **mice** 1.0 introduced predictor selection, passive imputation and automatic pooling. This article documents **mice** 2.9, which extends the functionality of **mice** 1.0 in several ways. In **mice** 2.9, the analysis of imputed data is made completely general, whereas the range of models under which pooling works is substantially extended. **mice** 2.9 adds new functionality for imputing multilevel data, automatic predictor selection, data handling, post-processing imputed values, specialized pooling routines, model selection tools, and diagnostic graphs. Imputation of categorical data is improved in order to bypass problems caused by perfect prediction. Special attention is paid to transformations, sum scores, indices and interactions using passive imputation, and to the proper setup of the predictor matrix. **mice** 2.9 can be downloaded from the Comprehensive R Archive Network. This article provides a hands-on, stepwise approach to solve applied incomplete data problems.

*Keywords*: MICE, multiple imputation, chained equations, fully conditional specification, Gibbs sampler, predictor selection, passive imputation, R.

## 1. Introduction

Multiple imputation (Rubin 1987, 1996) is the method of choice for complex incomplete data problems. Missing data that occur in more than one variable presents a special challenge. Two general approaches for imputing multivariate data have emerged: joint modeling (JM) and fully conditional specification (FCS), also known as multivariate imputation by chained equations (MICE). Schafer (1997) developed various JM techniques for imputation under the multivariate normal, the log-linear, and the general location model. JM involves specifying a multivariate distribution for the missing data, and drawing imputation from their conditional

distributions by Markov chain Monte Carlo (MCMC) techniques. This methodology is attractive if the multivariate distribution is a reasonable description of the data. FCS specifies the multivariate imputation model on a variable-by-variable basis by a set of conditional densities, one for each incomplete variable. Starting from an initial imputation, FCS draws imputations by iterating over the conditional densities. A low number of iterations (say 10–20) is often sufficient. FCS is attractive as an alternative to JM in cases where no suitable multivariate distribution can be found. The basic idea of FCS is already quite old, and has been proposed using a variety of names: stochastic relaxation (Kennickell 1991), variable-by-variable imputation (Brand 1999), regression switching (van Buuren *et al.* 1999), sequential regressions (Raghunathan *et al.* 2001), ordered pseudo-Gibbs sampler (Heckerman *et al.* 2001), partially incompatible MCMC (Rubin 2003), iterated univariate imputation (Gelman 2004), MICE (van Buuren and Oudshoorn 2000; van Buuren and Groothuis-Oudshoorn 2011) and FCS (van Buuren 2007).

*Software implementations*

Several authors have implemented fully conditionally specified models for imputation. **mice** 1.0 (van Buuren and Oudshoorn 2000) was released as an S-PLUS library in 2000, and was converted by several users into R (R Development Core Team 2011). **IVEware** (Raghunathan *et al.* 2001) is a SAS-based procedure that was independently developed by Raghunathan and colleagues. The function `aRegImpute` in R and S-PLUS is part of the **Hmisc** package (Harrell 2001). The **ice** software (Royston 2004, 2005; Royston and White 2011) is a widely used implementation in Stata. **SOLAS** 3.0 (Statistical Solutions 2001) is also based on conditional specification, but does not iterate. **WinMICE** (Jacobusse 2005) is a Windows stand-alone program for generating imputations under the hierarchical linear model. A recent addition is the R package **mi** (Su *et al.* 2011). Furthermore, FCS is now widely available through the `multiple imputation` procedure part of the SPSS 17 Missing Values Analysis add-on module. See `http://www.multiple-imputation.com/` for an overview.

*Applications of chained equations*

Applications of imputation by chained equations have now appeared in quite diverse fields: addiction (Schnoll *et al.* 2006; MacLeod *et al.* 2008; Adamczyk and Palmer 2008; Caria *et al.* 2009; Morgenstern *et al.* 2009), arthritis and rheumatology (Wolfe *et al.* 2006; Rahman *et al.* 2008; van den Hout *et al.* 2009), atherosclerosis (Tiemeier *et al.* 2004; van Oijen *et al.* 2007; McClelland *et al.* 2008), cardiovascular system (Ambler *et al.* 2005; van Buuren *et al.* 2006a; Chase *et al.* 2008; Byrne *et al.* 2009; Klein *et al.* 2009), cancer (Clark *et al.* 2001, 2003; Clark and Altman 2003; Royston *et al.* 2004; Barosi *et al.* 2007; Fernandes *et al.* 2008; Sharma *et al.* 2008; McCaul *et al.* 2008; Huo *et al.* 2008; Gerestein *et al.* 2009), epidemiology (Cummings *et al.* 2006; Hindorff *et al.* 2008; Mueller *et al.* 2008; Ton *et al.* 2009), endocrinology (Rouxel *et al.* 2004; Prompers *et al.* 2008), infectious diseases (Cottrell *et al.* 2005; Walker *et al.* 2006; Cottrell *et al.* 2007; Kekitiinwa *et al.* 2008; Nash *et al.* 2008; Sabin *et al.* 2008; Thein *et al.* 2008; Garabed *et al.* 2008; Michel *et al.* 2009), genetics (Souverein *et al.* 2006), health economics (Briggs *et al.* 2003; Burton *et al.* 2007; Klein *et al.* 2008; Marshall *et al.* 2009), obesity and physical activity (Orsini *et al.* 2008a; Wiles *et al.* 2008; Orsini *et al.* 2008b; van Vlierberghe *et al.* 2009), pediatrics and child development (Hill *et al.* 2004; Mumtaz *et al.* 2007; Deave *et al.* 2008; Samant *et al.* 2008; Butler and Heron 2008; Ramchandani *et al.* 2008; van Wouwe *et al.* 2009), rehabilitation (van der Hulst *et al.* 2008), behavior (Veenstra

*et al.* 2005; Melhem *et al.* 2007; Horwood *et al.* 2008; Rubin *et al.* 2008), quality of care (Sisk *et al.* 2006; Roudsari *et al.* 2007; Ward and Franks 2007; Grote *et al.* 2007; Roudsari *et al.* 2008; Grote *et al.* 2008; Sommer *et al.* 2009), human reproduction (Smith *et al.* 2004a,b; Hille *et al.* 2005; Alati *et al.* 2006; O'Callaghan *et al.* 2006; Hille *et al.* 2007; Hartog *et al.* 2008), management sciences (Jensen and Roy 2008), occupational health (Heymans *et al.* 2007; Brunner *et al.* 2007; Chamberlain *et al.* 2008), politics (Tanasoiu and Colonescu 2008), psychology (Sundell *et al.* 2008) and sociology (Finke and Adamczyk 2008). All authors use some form of chained equations to handle the missing data, but the details vary considerably. The interested reader could check out articles from a familiar application area to see how multiple imputation is done and reported.

*Features*

This paper describes the R package **mice** 2.9 for multiple imputation: generating multiple imputation, analyzing imputed data, and for pooling analysis results. Specific features of the software are:

- Columnwise specification of the imputation model (Section 3.2).
- Arbitrary patterns of missing data (Section 6.2).
- Passive imputation (Section 3.4).
- Subset selection of predictors (Section 3.3).
- Support of arbitrary complete-data methods (Section 5.1).
- Support pooling various types of statistics (Section 5.3).
- Diagnostics of imputations (Section 4.5).
- Callable user-written imputation functions (Section 6.1).

Package **mice** 2.9 replaces version **mice** 1.21, but is compatible with previous versions. This document replaces the original manual (van Buuren and Oudshoorn 2000). The **mice** 2.9 package extends **mice** 1.0 in several ways. New features in **mice** 2.9 include:

- `quickpred()` for automatic generation of the predictor matrix (Section 3.3).
- `mice.impute.2L.norm()` for imputing multilevel data (Section 3.3).
- Stable imputation of categorical data (Section 4.4).
- Post-processing imputations through the `post` argument (Section 3.5).
- `with.mids()` for general data analysis on imputed data (Section 5.1).
- `pool.scalar()` and `pool.r.squared()` for specialized pooling (Section 5.3).
- `pool.compare()` for model testing on imputed data (Section 5.3).
- `cbind.mids()`, `rbind.mids()` and `ibind()` for combining imputed data (see help file of these functions).

Furthermore, this document introduces a new strategy to specify the predictor matrix in conjunction with passive imputation. The amount and scope of example code has been expanded considerably. All programming code used in this paper is available in the file `v45i03.R` along with the manuscript and as `doc/JSScode.R` in the **mice** package.

The intended audience of this paper consists of applied researchers who want to address problems caused by missing data by multiple imputation. The text assumes basic familiarity with R. The document contains hands-on analysis using the **mice** package. We do not discuss problems of incomplete data in general. We refer to the excellent books by Little and Rubin (2002) and Schafer (1997). Theory and applications of multiple imputation have been developed in Rubin (1987) and Rubin (1996). van Buuren (2012) introduces multiple imputation from an applied perspective.

Package **mice** 2.9 was written in pure R using old-style S3 classes and methods. **mice** 2.9 was written and tested in R 2.12.2. The package has a simple architecture, is highly modular, and allows easy access to all program code from within the R environment.

## 2. General framework

To the uninitiated, multiple imputation is a bewildering technique that differs substantially from conventional statistical approaches. As a result, the first-time user may get lost in a labyrinth of imputation models, missing data mechanisms, multiple versions of the data, pooling, and so on. This section describes a modular approach to multiple imputation that forms the basis of the architecture of **mice**. The philosophy behind the MICE methodology is that multiple imputation is best done as a sequence of small steps, each of which may require diagnostic checking. Our hope is that the framework will aid the user to map out the steps needed in practical applications.

### 2.1. Notation

Let $Y_j$ with $(j = 1, \ldots, p)$ be one of $p$ incomplete variables, where $Y = (Y_1, \ldots, Y_p)$. The observed and missing parts of $Y_j$ are denoted by $Y_j^{\text{obs}}$ and $Y_j^{\text{mis}}$, respectively, so $Y^{\text{obs}} = (Y_1^{\text{obs}}, \ldots, Y_p^{\text{obs}})$ and $Y^{\text{mis}} = (Y_1^{\text{mis}}, \ldots, Y_p^{\text{mis}})$ stand for the observed and missing data in $Y$. The number of imputation is equal to $m \geq 1$. The $h$th imputed data sets is denoted as $Y^{(h)}$ where $h = 1, \ldots, m$. Let $Y_{-j} = (Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_p)$ denote the collection of the $p - 1$ variables in $Y$ except $Y_j$. Let $Q$ denote the quantity of scientific interest (e.g., a regression coefficient). In practice, $Q$ is often a multivariate vector. More generally, $Q$ encompasses any model of scientific interest.

### 2.2. Modular approach to multiple imputation

Figure 1 illustrates the three main steps in multiple imputation: imputation, analysis and pooling. The software stores the results of each step in a specific class: `mids`, `mira` and `mipo`. We now explain each of these in more detail.

The leftmost side of the picture indicates that the analysis starts with an observed, incomplete data set $Y_{\text{obs}}$. In general, the problem is that we cannot estimate $Q$ from $Y_{\text{obs}}$ without making unrealistic assumptions about the unobserved data. Multiple imputation is a general framework that several imputed versions of the data by replacing the missing values by plau-

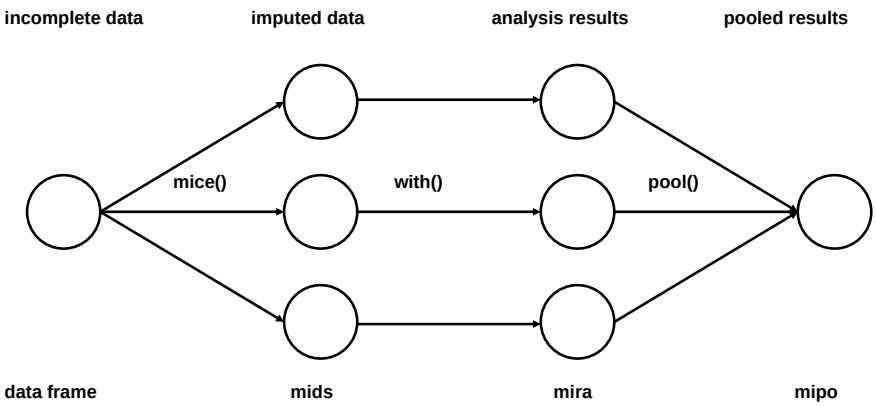


Figure 1: Main steps used in multiple imputation.

sible data values. These plausible values are drawn from a distribution specifically modeled for each missing entry. In **mice** this task is being done by the function `mice()`. Figure 1 portrays $m = 3$ imputed data sets $Y^{(1)}, \ldots, Y^{(3)}$. The three imputed sets are identical for the non-missing data entries, but differ in the imputed values. The magnitude of these difference reflects our uncertainty about what value to impute. The package has a special class for storing the imputed data: a *multiply imputed dataset* of class `mids`.

The second step is to estimate $Q$ on each imputed data set, typically by the method we would have used if the data had been complete. This is easy since all data are now complete. The model applied to $Y^{(1)}, \ldots, Y^{(m)}$ is the generally identical. **mice** 2.9 contains a function `with.mids()` that perform this analysis. This function supersedes the `lm.mids()` and `glm.mids()`. The estimates $\hat{Q}^{(1)}, \ldots, \hat{Q}^{(m)}$ will differ from each other because their input data differ. It is important to realize that these differences are caused because of our uncertainty about what value to impute. In **mice** the analysis results are collectively stored as a *multiply imputed repeated analysis* within an R object of class `mira`.

The last step is to pool the $m$ estimates $\hat{Q}^{(1)}, \ldots, \hat{Q}^{(m)}$ into one estimate $\bar{Q}$ and estimate its variance. For quantities $Q$ that are approximately normally distributed, we can calculate the mean over $\hat{Q}^{(1)}, \ldots, \hat{Q}^{(m)}$ and sum the within- and between-imputation variance according to the method outlined in Rubin (1987, pp. 76–77). The function `pool()` contains methods for pooling quantities by Rubin's rules. The results of the function is stored as a *multiple imputed pooled outcomes* object of class `mipo`.

## 2.3. MICE algorithm

The imputation model should

- Account for the process that created the missing data.
- Preserve the relations in the data.
- Preserve the uncertainty about these relations.

The hope is that adherence to these principles will yield imputations that are statistically

correct as in Rubin (1987, Chapter 4) for a wide range in $Q$. Typical problems that may surface while imputing multivariate missing data are

- For a given $Y_j$, predictors $Y_{-j}$ used in the imputation model may themselves be incomplete.
- Circular dependence can occur, where $Y_1$ depends on $Y_2$ and $Y_2$ depends on $Y_1$ because in general $Y_1$ and $Y_2$ are correlated, even given other variables.
- Especially with large $p$ and small $n$, collinearity and empty cells may occur.
- Rows or columns can be ordered, e.g., as with longitudinal data.
- Variables can be of different types (e.g., binary, unordered, ordered, continuous), thereby making the application of theoretically convenient models, such as the multivariate normal, theoretically inappropriate.
- The relation between $Y_j$ and $Y_{-j}$ could be complex, e.g., nonlinear, or subject to censoring processes.
- Imputation can create impossible combinations (e.g., pregnant fathers), or destroy deterministic relations in the data (e.g., sum scores).
- Imputations can be nonsensical (e.g., body temperature of the dead).
- Models for $Q$ that will be applied to the imputed data may not (yet) be known.

This list is by no means exhaustive, and other complexities may appear for particular data.

In order to address the issues posed by the real-life complexities of the data, it is convenient to specify the imputation model separately for each column in the data. This has led by to the development of the technique of *chained equations*. Specification occurs on at a level that is well understood by the user, i.e., at the variable level. Moreover, techniques for creating univariate imputations have been well developed.

Let the hypothetically complete data $Y$ be a partially observed random sample from the $p$-variate multivariate distribution $P(Y|\theta)$. We assume that the multivariate distribution of $Y$ is completely specified by $\theta$, a vector of unknown parameters. The problem is how to get the multivariate distribution of $\theta$, either explicitly or implicitly. The MICE algorithm obtains the posterior distribution of $\theta$ by sampling iteratively from conditional distributions of the form

$$\begin{aligned} &P(Y_1|Y_{-1},\theta_1)\\ &\vdots\\ &P(Y_p|Y_{-p},\theta_p). \end{aligned}$$

The parameters $\theta_1,\ldots,\theta_p$ are specific to the respective conditional densities and are not necessarily the product of a factorization of the 'true' joint distribution $P(Y|\theta)$. Starting from a simple draw from observed marginal distributions, the $t$th iteration of chained equations is a Gibbs sampler that successively draws

$$\theta_1^{*(t)} \sim P(\theta_1|Y_1^{\text{obs}},Y_2^{(t-1)},\ldots,Y_p^{(t-1)})$$

$$
\begin{aligned}
Y_1^{*(t)} &\sim P(Y_1|Y_1^{\text{obs}}, Y_2^{(t-1)}, \ldots, Y_p^{(t-1)}, \theta_1^{*(t)}) \\
&\vdots \\
\theta_p^{*(t)} &\sim P(\theta_p|Y_p^{\text{obs}}, Y_1^{(t)}, \ldots, Y_{p-1}^{(t)}) \\
Y_p^{*(t)} &\sim P(Y_p|Y_p^{\text{obs}}, Y_1^{(t)}, \ldots, Y_p^{(t)}, \theta_p^{*(t)})
\end{aligned}
$$

where $Y_j^{(t)} = (Y_j^{\text{obs}}, Y_j^{*(t)})$ is the $j$th imputed variable at iteration $t$. Observe that previous imputations $Y_j^{*(t-1)}$ only enter $Y_j^{*(t)}$ through its relation with other variables, and not directly. Convergence can therefore be quite fast, unlike many other MCMC methods. It is important to monitor convergence, but in our experience the number of iterations can often be a small number, say 10–20. The name chained equations refers to the fact that the MICE algorithm can be easily implemented as a concatenation of univariate procedures to fill out the missing data. The `mice()` function executes $m$ streams in parallel, each of which generates one imputed data set.

The MICE algorithm possesses a touch of magic. The method has been found to work well in a variety of simulation studies (Brand 1999; Horton and Lipsitz 2001; Moons *et al.* 2006; van Buuren *et al.* 2006b; Horton and Kleinman 2007; Yu *et al.* 2007; Schunk 2008; Drechsler and Rassler 2008; Giorgi *et al.* 2008). Note that it is possible to specify models for which no known joint distribution exits. Two linear regressions specify a joint multivariate normal given specific regularity condition (Arnold and Press 1989). However, the joint distribution of one linear and, say, one proportional odds regression model is unknown, yet very easy to specify with the MICE framework. The conditionally specified model may be incompatible in the sense that the joint distribution cannot exist. It is not yet clear what the consequences of incompatibility are on the quality of the imputations. The little simulation work that is available suggests that the problem is probably not serious in practice (van Buuren *et al.* 2006b; Drechsler and Rassler 2008). Compatible multivariate imputation models (Schafer 1997) have been found to work in a large variety of cases, but may lack flexibility to address specific features of the data. Gelman and Raghunathan (2001) remark that "separate regressions often make more sense than joint models". In order to bypass the limitations of joint models, Gelman (2004, pp. 541) concludes: "Thus we are suggesting the use of a new class of models—inconsistent conditional distributions—that were initially motivated by computational and analytical convenience." As a safeguard to evade potential problems by incompatibility, we suggest that the order in which variable are imputed should be sensible. This ordering can be specified in **mice** (cf. Section 3.6). Existence and uniqueness theorems for conditionally specified models have been derived (Arnold and Press 1989; Arnold *et al.* 1999; Ip and Wang 2009). More work along these lines would be useful in order to identify the boundaries at which the MICE algorithm breaks down. Barring this, the method seems to work well in many examples, is of great importance in practice, and is easily applied.

## 2.4. Simple example

The section presents a simple example incorporating all three steps. After installing the R package **mice** from the Comprehensive R Archive Network (CRAN), load the package.

```
R> library("mice")
```

This paper uses the features of **mice** 2.9. The data frame `nhanes` contains data from Schafer

(1997, p. 237). The data contains four variables: `age` (age group), `bmi` (body mass index), `hyp` (hypertension status) and `chl` (cholesterol level). The data are stored as a `data frame`. Missing values are represented as `NA`.

```
R> nhanes
```

```
   age  bmi hyp chl
1    1   NA  NA  NA
2    2 22.7   1 187
3    1   NA   1 187
4    3   NA  NA  NA
5    1 20.4   1 113
     ...
```

*Inspecting the missing data*

The number of the missing values can be counted and visualized as follows:

```
R> md.pattern(nhanes)
```

```
   age hyp bmi chl
13   1   1   1   1  0
 1   1   1   0   1  1
 3   1   1   1   0  1
 1   1   0   0   1  2
 7   1   0   0   0  3
     0   8   9  10 27
```

There are 13 (out of 25) rows that are complete. There is one row for which only `bmi` is missing, and there are seven rows for which only `age` is known. The total number of missing values is equal to $(7 \times 3) + (1 \times 2) + (3 \times 1) + (1 \times 1) = 27$. Most missing values (10) occur in `chl`.

Another way to study the pattern involves calculating the number of observations per patterns for all pairs of variables. A pair of variables can have exactly four missingness patterns: both variables are observed (pattern `rr`), the first variable is observed and the second variable is missing (pattern `rm`), the first variable is missing and the second variable is observed (pattern `mr`), and both are missing (pattern `mm`). We can use the `md.pairs()` function to calculate the frequency in each pattern for all variable pairs as

```
R> p <- md.pairs(nhanes)
R> p
```

```
$rr
    age bmi hyp chl
age  25  16  17  15
bmi  16  16  16  13
```

```
hyp  17  16  17  14
chl  15  13  14  15

$rm
    age bmi hyp chl
age   0   9   8  10
bmi   0   0   0   3
hyp   0   1   0   3
chl   0   2   1   0

$mr
    age bmi hyp chl
age   0   0   0   0
bmi   9   0   1   2
hyp   8   0   0   1
chl  10   3   3   0

$mm
    age bmi hyp chl
age   0   0   0   0
bmi   0   9   8   7
hyp   0   8   8   7
chl   0   7   7  10
```

Thus, for pair `(bmi,chl)` there are 13 completely observed pairs, 3 pairs for which `bmi` is observed but `hyp` not, 2 pairs for which `bmi` is missing but with `hyp` observed, and 7 pairs with both missing `bmi` and `hyp`. Note that these numbers add up to the total sample size.

The R package **VIM** (Templ *et al.* 2011) contains functions for plotting incomplete data. The *margin plot* of the pair `(bmi,chl)` can be plotted by

```
R> library("VIM")
R> marginplot(nhanes[, c("chl", "bmi")], col = mdc(1:2), cex = 1.2,
+    cex.lab = 1.2, cex.numbers = 1.3, pch = 19)
```

Figure 2 displays the result. The data area holds 13 blue points for which both `bmi` and `chl` were observed. The plot in Figure 2 requires a graphic device that supports transparent colors, e.g., `pdf()`. To create the plot in other devices, change the `col = mdc(1:2)` argument to `col = mdc(1:2, trans = FALSE)`. The three red dots in the left margin correspond to the records for which `bmi` is observed and `chl` is missing. The points are drawn at the known values of `bmi` at 24.9, 25.5 and 29.6. Likewise, the bottom margin contain two red points with observed `chl` and missing `bmi`. The red dot at the intersection of the bottom and left margin indicates that there are records for which both `bmi` and `chl` are missing. The three numbers at the lower left corner indicate the number of incomplete records for various combinations. There are 9 records in which `bmi` is missing, 10 records in which `chl` is missing, and 7 records in which both are missing. Furthermore, the left margin contain two box plots, a blue and a red one. The blue box plot in the left margin summarizes the marginal distribution of `bmi` of the 13 blue points. The red box plot summarizes the distribution of the three `bmi` values

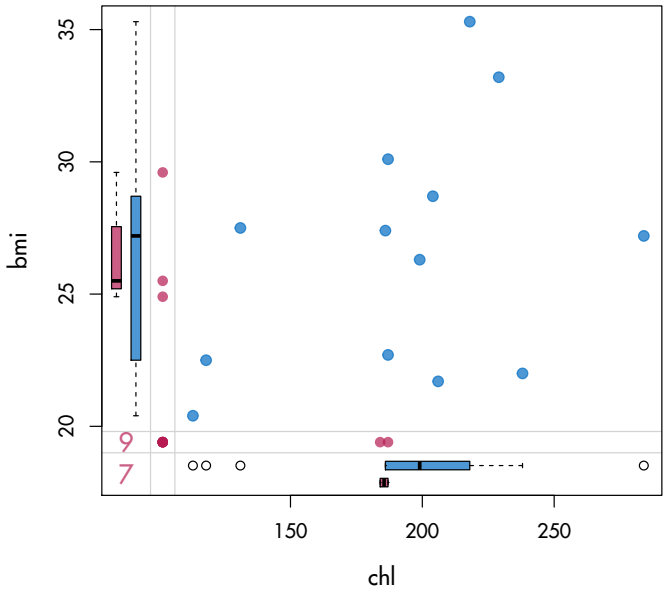


Figure 2: Margin plot of `bmi` versus `chl` as drawn by the `marginplot()` function in the **VIM** package. Observed data in blue, missing data in red.

with missing `chl`. Under MCAR, these distribution are expected to be identical. Likewise, the two colored box plots in the bottom margin summarize the respective distributions for `chl`.

*Creating imputations*

Creating imputations can be done with a call to `mice()` as follows:

```
R> imp <- mice(nhanes, seed = 23109)

 iter imp variable
  1   1  bmi  hyp  chl
  1   2  bmi  hyp  chl
  1   3  bmi  hyp  chl
  1   4  bmi  hyp  chl
  1   5  bmi  hyp  chl
  2   1  bmi  hyp  chl
  2   2  bmi  hyp  chl
     ...
```

where the multiply imputed data set is stored in the object `imp` of class `mids`. Inspect what the result looks like

```
R> print(imp)
```

```
Multiply imputed data set
Call:
mice(data = nhanes, seed = 23109)
Number of multiple imputations:  5
Missing cells per column:
age bmi hyp chl
  0   9   8  10
Imputation methods:
  age   bmi   hyp   chl
   "" "pmm" "pmm" "pmm"
VisitSequence:
bmi hyp chl
  2   3   4
PredictorMatrix:
    age bmi hyp chl
age   0   0   0   0
bmi   1   0   1   1
hyp   1   1   0   1
chl   1   1   1   0
Random generator seed value:  23109
```

Imputations are generated according to the default method, which is, for numerical data, predictive mean matching (`pmm`). The entries `imp$VisitSequence` and `imp$PredictorMatrix` are algorithmic options that will be discusses later. The default number of multiple imputations is equal to $m = 5$.

*Diagnostic checking*

An important step in multiple imputation is to assess whether imputations are plausible. Imputations should be values that could have been obtained had they not been missing. Imputations should be close to the data. Data values that are clearly impossible (e.g., negative counts, pregnant fathers) should not occur in the imputed data. Imputations should respect relations between variables, and reflect the appropriate amount of uncertainty about their 'true' values. Diagnostic checks on the imputed data provide a way to check the plausibility of the imputations. The imputations for `bmi` are stored as

```
R> imp$imp$bmi
```

```
      1    2    3    4    5
1  29.6 27.2 29.6 27.5 29.6
3  29.6 26.3 29.6 30.1 28.7
4  20.4 29.6 27.2 24.9 21.7
6  21.7 25.5 27.4 21.7 21.7
10 20.4 22.0 28.7 29.6 22.5
11 22.0 35.3 35.3 30.1 29.6
12 20.4 28.7 27.2 27.5 25.5
16 22.0 35.3 30.1 29.6 28.7
21 27.5 33.2 22.0 35.3 22.0
```

Each row corresponds to a missing entry in `bmi`. The columns contain the multiple imputations. The completed data set combines the observed and imputed values. The (first) completed data set can be obtained as

```
R> complete(imp)
```

```
   age  bmi hyp chl
1    1 29.6   1 238
2    2 22.7   1 187
3    1 29.6   1 187
4    3 20.4   1 186
5    1 20.4   1 113
  ...
```

The `complete()` function extracts the five imputed data sets from the `imp` object as a long (row-stacked) matrix with 125 records. The missing entries in `nhanes` have now been filled by the values from the first (of five) imputation. The second completed data set can be obtained by `complete(imp, 2)`. For the observed data, it is identical to the first completed data set, but it may differ in the imputed data.

It is often useful to inspect the distributions of original and the imputed data. One way of doing this is to use the function `stripplot()` in **mice** 2.9, an adapted version of the same function in the package **lattice** (Sarkar 2008). The stripplot in Figure 3 is created as

```
R> stripplot(imp, pch = 20, cex = 1.2)
```

The figure shows the distributions of the four variables as individual points. Blue points are observed, the red points are imputed. The panel for `age` contains blue points only because `age` is complete. Furthermore, note that the red points follow the blue points reasonably well, including the gaps in the distribution, e.g., for `chl`.

The scatterplot of `chl` and `bmi` for each imputed data set in Figure 4 is created by

```
R> xyplot(imp, bmi ~ chl | .imp, pch = 20, cex = 1.4)
```

The figure redraws figure 2, but now for the observed and imputed data. Imputations are plotted in red. The blue points are the same across different panels, but the red point vary. The red points have more or less the same shape as blue data, which indicates that they could have been plausible measurements if they had not been missing. The differences between the red points represents our uncertainty about the true (but unknown) values.

Under MCAR, univariate distributions of the observed and imputed data are expected to be identical. Under MAR, they can be different, both in location and spread, but their multivariate distribution is assumed to be identical. There are many other ways to look at the completed data, but we defer of a discussion of those to Section 4.5.

*Analysis of imputed data*

Suppose that the complete-data analysis of interest is a linear regression of `chl` on `age` and `bmi`. For this purpose, we can use the function `with.mids()`, a wrapper function that applies the complete data model to each of the imputed data sets:

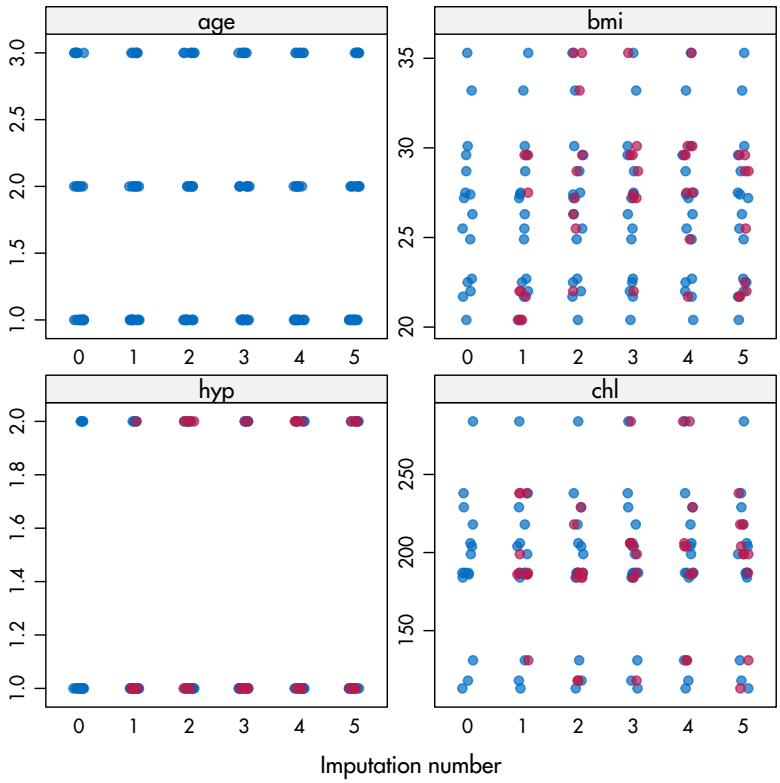


Figure 3: Stripplot of four variables in the original data and in the five imputed data sets. Points are slightly jittered. Observed data in blue, imputed data in red.

```
R> fit <- with(imp, lm(chl ~ age + bmi))
```

The `fit` object has class `mira` and contains the results of five complete-data analyses. These can be pooled as follows:

```
R> print(pool(fit))

Call: pool(object = fit)

Pooled coefficients:
(Intercept)         age         bmi
 -34.158914   34.330666    6.212025

Fraction of information about the coefficients missing due to nonresponse:
(Intercept)         age         bmi
  0.5747265   0.7501284   0.4795427
```

More detailed output can be obtained, as usual, with the `summary()` function, i.e.,

```
R> round(summary(pool(fit)), 2)
```

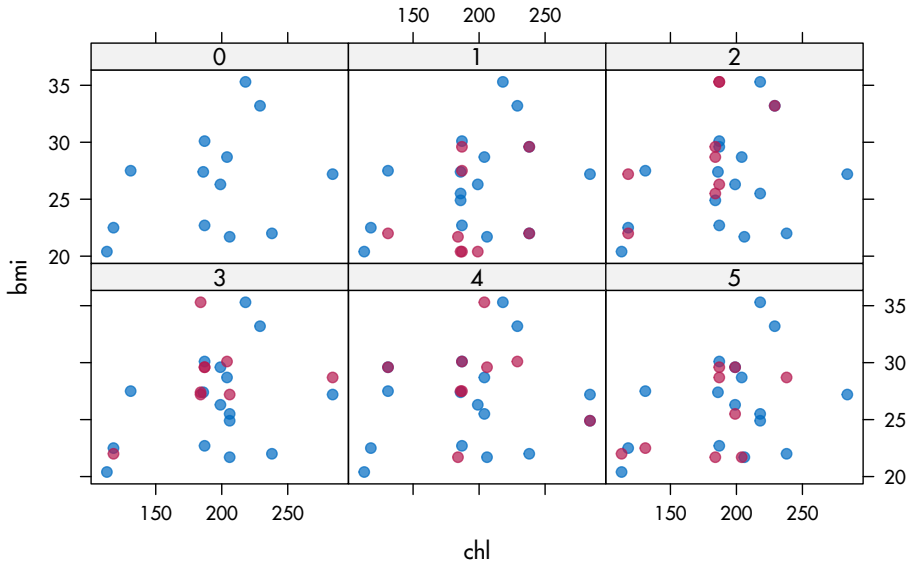


Figure 4: Scatterplot of cholesterol (`chl`) and body mass index (`bmi`) in the original data (panel 0), and five imputed data sets. Observed data in blue, imputed data in red.

```
               est    se     t   df Pr(>|t|)   lo 95  hi 95 nmis  fmi
(Intercept) -34.16 76.07 -0.45 6.81     0.67 -215.05 146.73   NA 0.57
age          34.33 14.86  2.31 4.04     0.08   -6.76  75.42    0 0.75
bmi           6.21  2.21  2.81 8.80     0.02    1.20  11.23    9 0.48
            lambda
(Intercept)   0.47
age           0.65
bmi           0.37
```

After multiple imputation, we find a significant effect `bmi`. The column `fmi` contains the *fraction of missing information* as defined in Rubin (1987), and the column `lambda` is the proportion of the total variance that is attributable to the missing data ($\lambda = (B + B/m)/T$).

The pooled results are subject to simulation error and therefore depend on the `seed` argument of the `mice()` function. In order to minimize simulation error, we can use a higher number of imputations, for example `m=50`. It is easy to do this as

```
R> imp50 <- mice(nhanes, m = 50, seed = 23109)
R> fit <- with(imp50, lm(chl ~ age + bmi))

R> round(summary(pool(fit)), 2)
```

```
               est    se     t    df Pr(>|t|)   lo 95  hi 95 nmis
(Intercept) -35.53 63.61 -0.56 14.46     0.58 -171.55 100.49   NA
age          35.90 10.48  3.42 12.76     0.00   13.21  58.58    0
bmi           6.15  1.97  3.13 15.13     0.01    1.96  10.35    9
```

```
             fmi lambda
(Intercept) 0.35   0.27
age         0.43   0.35
bmi         0.32   0.24
```

We find that actually both `age` and `chl` are significant effects. This is the result that can be reported.

# 3. Imputation models

## 3.1. Seven choices

The specification of the imputation model is the most challenging step in multiple imputation. What are the choices that we need to make, and in what order? There are seven main choices:

1. First, we should decide whether the missing at random (MAR) assumption (Rubin 1976) is plausible. The MAR assumption is a suitable starting point in many practical cases, but there are also cases where the assumption is suspect. Schafer (1997, pp. 20–23) provides a good set of practical examples. MICE can handle both MAR and missing not at random (MNAR). Multiple imputation under MNAR requires additional modeling assumptions that influence the generated imputations. There are many ways to do this. We refer to Section 6.2 for an example of how that could be realized.

2. The second choice refers to the form of the imputation model. The form encompasses both the structural part and the assumed error distribution. Within MICE the form needs to be specified for each incomplete column in the data. The choice will be steered by the scale of the dependent variable (i.e., the variable to be imputed), and preferably incorporates knowledge about the relation between the variables. Section 3.2 describes the possibilities within **mice** 2.9.

3. Our third choice concerns the set of variables to include as predictors into the imputation model. The general advice is to include as many relevant variables as possible including their interactions (Collins *et al.* 2001). This may however lead to unwieldy model specifications that could easily get out of hand. Section 3.3 describes the facilities within **mice** 2.9 for selecting the predictor set.

4. The fourth choice is whether we should impute variables that are functions of other (incomplete) variables. Many data sets contain transformed variables, sum scores, interaction variables, ratio's, and so on. It can be useful to incorporate the transformed variables into the multiple imputation algorithm. Section 3.4 describes how **mice** 2.9 deals with this situation using passive imputation.

5. The fifth choice concerns the order in which variables should be imputed. Several strategies are possible, each with their respective pro's and cons. Section 3.6 shows how the visitation scheme of the MICE algorithm within **mice** 2.9 is under control of the user.

| Method | Description | Scale type | Default |
|---|---|---|---|
| `pmm` | Predictive mean matching | numeric | Y |
| `norm` | Bayesian linear regression | numeric | |
| `norm.nob` | Linear regression, non-Bayesian | numeric | |
| `mean` | Unconditional mean imputation | numeric | |
| `2L.norm` | Two-level linear model | numeric | |
| `logreg` | Logistic regression | factor, 2 levels | Y |
| `polyreg` | Multinomial logit model | factor, >2 levels | Y |
| `polr` | Ordered logit model | ordered, >2 levels | Y |
| `lda` | Linear discriminant analysis | factor | |
| `sample` | Random sample from the observed data | any | |

Table 1: Built-in univariate imputation techniques. The techniques are coded as functions named `mice.impute.pmm()`, and so on.

6. The sixth choice concerns the setup of the starting imputations and the number of iterations. The convergence of the MICE algorithm can be monitored in many ways. Section 4.3 outlines some techniques in **mice** 2.9 that assist in this task.

7. The seventh choice is $m$, the number of multiply imputed data sets. Setting $m$ too low may result in large simulation error, especially if the fraction of missing information is high.

Please realize that these choices are always needed. The analysis in Section 2.4 imputed the `nhanes` data using just a minimum of specifications and relied on **mice** defaults. However, these default choices are not necessarily the best for your data. There is no magical setting that produces appropriate imputations in every problem. Real problems need tailoring. It is our hope that the software will invite you to go beyond the default settings.

### 3.2. Univariate imputation methods

In MICE one specifies a univariate imputation model of each incomplete variable. Both the structural part of the imputation model and the error distribution need to be specified. The choice will depend on, amongst others, the scale of the variable to be imputed. The univariate imputation method takes a set of (at that moment) complete predictors, and returns a single imputation for each missing entry in the incomplete target column. The **mice** 2.9 package supplies a number of built-in univariate imputation models. These all have names `mice.impute.name`, where `name` identifies the univariate imputation method.

Table 1 contains the list of built-in imputation functions. The default methods are indicated. The `method` argument of `mice()` specifies the imputation method per column and overrides the default. If `method` is specified as one string, then all visited data columns (cf. Section 3.6) will be imputed by the univariate function indicated by this string. So

```
R> imp <- mice(nhanes, method = "norm")
```

specifies that the function `mice.impute.norm()` is called for all columns. Alternatively, `method` can be a vector of strings of length `ncol(data)` specifying the function that is applied to each column. Columns that need not be imputed have method `""`. For example,

```
R> imp <- mice(nhanes, meth = c("", "norm", "pmm", "mean"))
```

applies different methods for different columns. The `nhanes2` data frame contains one polytomous, one binary and two numeric variables.

```
R> str(nhanes2)
```

```
'data.frame':        25 obs. of  4 variables:
 $ age: Factor w/ 3 levels "20-39","40-59",..: 1 2 1 3 1 3 1 1 2 2 ...
 $ bmi: num  NA 22.7 NA NA 20.4 NA 22.5 30.1 22 NA ...
 $ hyp: Factor w/ 2 levels "no","yes": NA 1 1 NA 1 NA 1 1 1 NA ...
 $ chl: num  NA 187 187 NA 113 184 118 187 238 NA ...
```

Imputations can be created as

```
R> imp <- mice(nhanes2, me = c("polyreg", "pmm", "logreg", "norm"))
```

where function `mice.impute.polyreg()` is used to impute the first (categorical) variable `age`, `mice.impute.ppm()` for the second numeric variable `bmi`, function `mice.impute.logreg()` for the third binary variable `hyp` and function `mice.impute.norm()` for the numeric variable `chl`. The `me` parameter is a legal abbreviation of the `method` argument.

*Empty imputation method*

The `mice()` function will automatically skip imputation of variables that are complete. One of the problems in previous versions this function was that all incomplete data needed to be imputed. In **mice** 2.9 it is possible to skip imputation of selected incomplete variables by specifying the empty method `""`. This works as long as the incomplete variable that is skipped is not being used as a predictor for imputing other variables. The `mice()` function will detect this case, and automatically remove the variable from the predictor list. For example, suppose that we do not want to impute `bmi`, but still want to retain in it the imputed data. We can run the following

```
R> imp <- mice(nhanes2, meth = c("", "", "logreg", "norm"))
```

This statement runs because `bmi` is removed from the predictor list. When removal is not possible, the program aborts with an error message like

```
Error in check.predictorMatrix(predictorMatrix, method, varnames,
nmis, : Variable bmi is used, has missing values, but is not imputed
```

Section 3.3 explains how to solve this problem.

*Perfect prediction*

Previous versions produced warnings like `fitted probabilities numerically 0 or 1 occurred` and `algorithm did not converge` on these data. These warnings are caused by the sample size of 25 relative to the number of parameters. **mice** 2.9 implements more stable

algorithms into `mice.impute.logreg()` and `mice.impute.polyreg()` based on augmenting the rows prior to imputation (White *et al.* 2010).

*Default imputation method*

The **mice** package distinguishes between four types of variables: numeric, binary (factor with 2 levels), and unordered (factor with more than 2 levels) and ordered (ordered factor with more than 2 levels). Each type has a default imputation method, which are indicated in Table 1. These defaults can be changed by the `defaultMethod` argument to the `mice()` function. For example

```
R> mice(nhanes2, defaultMethod = c("norm", "logreg", "polyreg", "polr"))
```

applies the function `mice.impute.norm()` to each numeric variable in nhanes instead of `mice.impute.pmm()`. It leaves the defaults for binary and categorical data unchanged. The `mice()` function checks the type of the variable against the specified imputation method, and produces a warning if a type mismatch is found.

*Overview of imputation methods*

The function `mice.impute.pmm()` implements predictive mean matching (Little 1988), a general purpose semi-parametric imputation method. Its main virtues are that imputations are restricted to the observed values and that it can preserve non-linear relations even if the structural part of the imputation model is wrong. It is a good overall imputation method. The functions `mice.impute.norm()` and `mice.impute.norm.nob()` impute according to a linear imputation model, and are fast and efficient if the model residuals are close to normal. The second model ignores any sampling uncertainty of the imputation model, so it is only appropriate for very large samples. The method `mice.impute.mean()` simply imputes the mean of the observed data. Mean imputation is known to be a bad strategy, and the user should be aware of the implications.

The function `mice.impute.2L.norm()` imputes according to the heteroscedastic linear two-level model by a Gibbs sampler (Note: Interpret '2L' as 'two levels', not as 'twenty-one'). It is new in **mice** 2.9. The method considerably improves upon standard methods that ignore the clustering structure, or that model the clustering as fixed effects (van Buuren 2010). See multilevel imputation in Section 3.3 for an example.

The function `mice.impute.polyreg()` imputes factor with two or more levels by the multinomial model using the `multinom()` function in **nnet** (Venables and Ripley 2002) for the hard work. The function `mice.impute.polr()` implements the ordered logit model, also known as the proportional odds model. It calls `polr` from **MASS** (Venables and Ripley 2002). The function `mice.impute.lda()` uses the **MASS** function `lda()` for linear discriminant analysis to compute posterior probabilities for each incomplete case, and subsequently draws imputations from these posteriors. This statistical properties of this method are not as good as `mice.impute.polyreg()`(Brand 1999), but it is a bit faster and uses fewer resources. The maximum number of categories these function handle is set to 50. Finally, the function `mice.impute.sample()` just takes a random draw from the observed data, and imputes these into missing cells. This function does not condition on any other variable. `mice()` calls `mice.impute.sample()` for initialization.

The univariate imputation functions are designed to be called from the main function `mice()`, and this is by far the easiest way to invoke them. It is however possible to call them directly, assuming that the arguments are all properly specified. See the documentation for more details.

### 3.3. Predictor selection

One of the most useful features of the MICE algorithm is the ability to specify the set of predictors to be used for each incomplete variable. The basic specification is made through the `predictorMatrix` argument, which is a square matrix of size `ncol(data)` containing 0/1 data. Each row in `predictorMatrix` identifies which predictors are to be used for the variable in the row name. If `diagnostics = TRUE` (the default), then `mice()` returns a `mids` object containing a `predictorMatrix` entry. For example, type

```
R> imp <- mice(nhanes, print = FALSE)
R> imp$predictorMatrix
```

```
    age bmi hyp chl
age   0   0   0   0
bmi   1   0   1   1
hyp   1   1   0   1
chl   1   1   1   0
```

The row correspond to incomplete target variables, in the sequence as they appear in data. Row and column names of the `predictorMatrix` are ignored on input, and overwritten by `mice()` on output. A value of `1` indicates that the column variable is used as a predictor to impute the target (row) variable, and a `0` means that it is not used. Thus, in the above example, `bmi` is predicted from `age`, `hyp` and `chl`. Note that the diagonal is `0` since a variable cannot predict itself. Since `age` contains no missing data, `mice()` silently sets all values in the row to `0`. The default setting of the `predictorMatrix` specifies that all variables predict all others.

*Removing a predictor*

The user can specify a custom `predictorMatrix`, thereby effectively regulating the number of predictors per variable. For example, suppose that `bmi` is considered irrelevant as a predictor. Setting all entries within the `bmi` column to zero effectively removes it from the predictor set, e.g.,

```
R> pred <- imp$predictorMatrix
R> pred[, "bmi"] <- 0
R> pred
```

```
    age bmi hyp chl
age   0   0   0   0
bmi   1   0   1   1
hyp   1   0   0   1
chl   1   0   1   0
```

will not use `bmi` as a predictor, but still impute it. Using this new specification, we create imputations as

```
R> imp <- mice(nhanes, pred = pred, pri = FALSE)
```

This imputes the incomplete variables `hyp` and `chl` without using `bmi` as a predictor.

*Skipping imputation*

Suppose that we want to skip imputation of `bmi`, and leave it as it is. This can be achieved by 1) eliminating `bmi` from the predictor set, and 2) setting the imputation method to `""`. More specifically

```
R> ini <- mice(nhanes2, maxit = 0, pri = FALSE)
R> pred <- ini$pred
R> pred[, "bmi"] <- 0
R> meth <- ini$meth
R> meth["bmi"] <- ""
R> imp <- mice(nhanes2, meth = meth, pred = pred, pri = FALSE)
R> imp$imp$bmi
```

```
   1  2  3  4  5
1  NA NA NA NA NA
3  NA NA NA NA NA
4  NA NA NA NA NA
6  NA NA NA NA NA
10 NA NA NA NA NA
11 NA NA NA NA NA
12 NA NA NA NA NA
16 NA NA NA NA NA
21 NA NA NA NA NA
```

The first statement calls `mice()` with the maximum number of iterations `maxit` set to zero. This is a fast way to create the `mids` object called `ini` containing the default settings. It is then easy to copy and edit the `predictorMatrix` and `method` arguments of the `mice()` function. Now `mice()` will impute `NA` into the missing values of `bmi`. In effect, the original `bmi` (with the missing values included) is copied into the multiply imputed data set. This technique is not only useful for 'keeping all the data together', but also opens up ways to performs imputation by nested blocks of variables. For examples where this could be useful, see Shen (2000) and Rubin (2003).

*Intercept imputation*

Eliminating all predictors for `bmi` can be done by

```
R> pred <- ini$pred
R> pred["bmi", ] <- 0
R> imp <- mice(nhanes2, pred = pred, pri = FALSE, seed = 51162)
R> imp$imp$bmi
```

```
      1    2    3    4    5
1  20.4 27.2 22.0 25.5 27.4
3  27.4 22.5 24.9 22.7 33.2
4  20.4 20.4 24.9 27.2 27.5
6  22.5 27.5 26.3 20.4 24.9
10 27.2 20.4 27.2 26.3 22.7
11 22.7 22.5 22.7 29.6 25.5
12 29.6 28.7 22.5 33.2 27.4
16 27.4 22.5 35.3 22.7 20.4
21 30.1 27.4 24.9 20.4 27.2
```

Imputations for `bmi` are now sampled (by `mice.impute.pmm()`) under the intercept-only model. Note that these imputations are appropriate only under the MCAR assumption.

*Multilevel imputation*

Imputation of multilevel data poses special problems. Most techniques have been developed under the joint modeling perspective (Schafer and Yucel 2002; Yucel 2008; Goldstein *et al.* 2009). Some work within the context of FCS has been done (Jacobusse 2005), but this is still an open research area. The **mice** 2.9 package include the `mice.impute.2L.norm()` function, which can be used to impute missing data under a linear multilevel model. The function was contributed by Roel de Jong, and implements the Gibbs sampler for the linear multilevel model where the within-class error variance is allowed to vary (Kasim and Raudenbush 1998). Heterogeneity in the variances is essential for getting good imputations in multilevel data. The method is an improvement over simpler methods like flat-file imputation or per-group imputation (van Buuren 2010).

Using `mice.impute.2L.norm()` (or equivalently `mice.impute.2l.norm()`) deviates from other univariate imputation functions in **mice** 2.9 in two respects. It requires the specification of the fixed effects, the random effects and the class variable. Furthermore, it assumes that the predictors contain a column of ones representing the intercept. Random effects are coded in the predictor matrix as a '2'. The class variable (only one is allowed) is coded by a '-2'. The example below uses the popularity data of (Hox 2002). The dependent variable is pupil `popularity`, which contains 848 missing values. There are two random effects: `const` (intercept) and `sex` (slope), one fixed effect, teacher experience (`texp`), and one class variable (`school`). Imputations can be generated as

```
R> popmis[1:3, ]

  pupil school popular sex texp const teachpop
1     1      1      NA   1   24     1        7
2     2      1      NA   0   24     1        7
3     3      1       7   1   24     1        6

R> ini <- mice(popmis, maxit = 0)
R> pred <- ini$pred
R> pred["popular", ] <- c(0, -2, 0, 2, 1, 2, 0)
R> imp <- mice(popmis, meth = c("", "", "2l.norm", "", "",
+      "", ""), pred = pred, maxit = 1, seed = 71152)
```

```
iter imp variable
 1   1  popular
 1   2  popular
 1   3  popular
 1   4  popular
 1   5  popular
```

The extension to the multivariate case will be obvious, but relatively little is known about the statistical properties.

*Advice on predictor selection*

The `predictorMatrix` argument is especially useful when dealing with data sets with a large number of variables. We now provide some advice regarding the selection of predictors for large data, especially with many incomplete data.

As a general rule, using every bit of available information yields multiple imputations that have minimal bias and maximal certainty (Meng 1995; Collins *et al.* 2001). This principle implies that the number of predictors should be chosen as large as possible. Including as many predictors as possible tends to make the MAR assumption more plausible, thus reducing the need to make special adjustments for NMAR mechanisms (Schafer 1997).

However, data sets often contain several hundreds of variables, all of which can potentially be used to generate imputations. It is not feasible (because of multicollinearity and computational problems) to include all these variables. It is also not necessary. In our experience, the increase in explained variance in linear regression is typically negligible after the best, say, 15 variables have been included. For imputation purposes, it is expedient to select a suitable subset of data that contains no more than 15 to 25 variables. van Buuren *et al.* (1999) provide the following strategy for selecting predictor variables from a large data base:

1. Include all variables that appear in the complete-data model, i.e., the model that will be applied to the data after imputation. Failure to do so may bias the complete-data analysis, especially if the complete-data model contains strong predictive relations. Note that this step is somewhat counter-intuitive, as it may seem that imputation would artificially strengthen the relations of the complete-data model, which is clearly undesirable. If done properly however, this is not the case. On the contrary, not including the complete-data model variables will tend to bias the results towards zero. Note that interactions of scientific interest also need to be included into the imputation model.

2. In addition, include the variables that are related to the nonresponse. Factors that are known to have influenced the occurrence of missing data (stratification, reasons for nonresponse) are to be included on substantive grounds. Others variables of interest are those for which the distributions differ between the response and nonresponse groups. These can be found by inspecting their correlations with the response indicator of the variable to be imputed. If the magnitude of this correlation exceeds a certain level, then the variable is included.

3. In addition, include variables that explain a considerable amount of variance. Such

predictors help to reduce the uncertainty of the imputations. They are crudely identified by their correlation with the target variable.

4. Remove from the variables selected in steps 2 and 3 those variables that have too many missing values within the subgroup of incomplete cases. A simple indicator is the percentage of observed cases within this subgroup, the percentage of usable cases.

Most predictors used for imputation are incomplete themselves. In principle, one could apply the above modeling steps for each incomplete predictor in turn, but this may lead to a cascade of auxiliary imputation problems. In doing so, one runs the risk that every variable needs to be included after all. In practice, there is often a small set of key variables, for which imputations are needed, which suggests that steps 1 through 4 are to be performed for key variables only. This was the approach taken in van Buuren *et al.* (1999), but it may miss important predictors of predictors. A safer and more efficient, though more laborious, strategy is to perform the modeling steps also for the predictors of predictors of key variables. This is done in Oudshoorn *et al.* (1999). We expect that it is rarely necessary to go beyond predictors of predictors. At the terminal node, we can apply a simply method like `mice.impute.sample()` that does not need any predictors for itself.

*Quick predictor selection*

Correlations for the strategy outlined above can be calculated with the standard function `cor()`. For example,

```
R> round(cor(nhanes, use = "pair"), 3)
```

```
       age    bmi   hyp   chl
age  1.000 -0.372 0.506 0.507
bmi -0.372  1.000 0.051 0.373
hyp  0.506  0.051 1.000 0.429
chl  0.507  0.373 0.429 1.000
```

calculates Pearson correlations using all available cases in each pair of variables. Similarly,

```
R> round(cor(y = nhanes, x = !is.na(nhanes), use = "pair"),
+      3)
```

```
       age    bmi    hyp   chl
age     NA     NA     NA    NA
bmi  0.086     NA  0.139 0.053
hyp  0.008     NA     NA 0.045
chl -0.040 -0.012 -0.107    NA
```

calculates the mutual correlations between the data and the response indicators. The warning can be safely ignored and is caused by the fact that `age` contains no missing data.

The *proportion of usable cases* measures how many cases with missing data on the target variable actually have observed values on the predictor. The proportion will be low if both

target and predictor are missing on the same cases. If so, the predictor contains only little information to impute the target variable, and could be dropped from the model, especially if the bivariate relation is not primary scientific interest. The proportion of usable cases can be calculated as

```
R> p <- md.pairs(nhanes)
R> round(p$mr/(p$mr + p$mm), 3)
```

```
    age bmi   hyp   chl
age NaN NaN   NaN   NaN
bmi   1 0.0 0.111 0.222
hyp   1 0.0 0.000 0.125
chl   1 0.3 0.300 0.000
```

For imputing `hyp` only 1 out of 8 cases was observed in predictor `chl`. Thus, predictor `chl` does not contain much information to impute `hyp`, despite the substantial correlation of 0.42. If the relation is of no further scientific interest, omitting predictor `chl` from the model to impute `hyp` will only have a small effect. Note that proportion of usable cases is asymmetric.

**mice** 2.9 contains a new function `quickpred()` that calculates these quantities, and combines them automatically in a `predictorMatrix` that can be used to call `mice()`. The `quickpred()` function assumes that the correlation is a sensible measure for the data at hand (e.g., order of factor levels should be reasonable). For example,

```
R> quickpred(nhanes)
```

```
    age bmi hyp chl
age   0   0   0   0
bmi   1   0   1   1
hyp   1   0   0   1
chl   1   1   1   0
```

yields a `predictorMatrix` for a model that includes all predictors with an absolute correlation with the target or with the response indicator of at least 0.1 (the default value of the `mincor` argument). Observe that the predictor matrix need not always be symmetric. In particular, `bmi` is not a predictor of `hyp`, but `hyp` is a predictor of `bmi` here. This can occur because the correlation of `hyp` with the response indicator of `bmi` (0.139) exceeds the threshold.

The `quickpred()` function has arguments that change the minimum correlation, that allow to select predictor based on their proportion of usable cases, and that can specify variables that should always be included or excluded. It is also possible to specify thresholds per target variable, or even per target-predictor combination. See the help files for more details.

It is easy to use the function in conjunction with `mice()`. For example,

```
R> imp <- mice(nhanes, pred = quickpred(nhanes, minpuc = 0.25,
+    include = "age"))
```

imputes the data from a model where the minimum proportion of usable cases is at least 0.25 and that always includes `age` as a predictor.

Any interactions of interest need to be appended to the data before using `quickpred()`. For large data, the user can experiment with the `mincor`, `minpuc`, `include` and `exclude` arguments to trim the imputation problem to a reasonable size. The application of `quickpred()` can substantially cut down the time needed to specify the imputation model for data with many variables.

### 3.4. Passive imputation

There is often a need for transformed, combined or recoded versions of the data. In the case of incomplete data, one could impute the original, and transform the completed original afterwards, or transform the incomplete original and impute the transformed version. If, however, both the original and the transform are needed within the imputation algorithm, neither of these approaches work because one cannot be sure that the transformation holds between the imputed values of the original and transformed versions.

**mice** implements a special mechanism, called *passive imputation*, to deal with such situations. Passive imputation maintains the consistency among different transformations of the same data. The method can be used to ensure that the transform always depends on the most recently generated imputations in the original untransformed data. Passive imputation is invoked by specifying a `~` (tilde) as the first character of the imputation method. The entire string, including the `~` is interpreted as the `formula` argument in a call to `model.frame(formula, data[!r[,j],])`. This provides a simple method for specifying a large variety of dependencies among the variables, such as transformed variables, recodes, interactions, sum scores, and so on, that may themselves be needed in other parts of the algorithm.

*Preserving a transformation*

As an example, suppose that previous research suggested that `bmi` is better imputed from `log(chl)` than from `chl`. We may thus want to add an extra column to the data with `log(chl)`, and impute `bmi` from `log(chl)`. Any missing values in `chl` will also be present in `log(chl)`. The problem is to keep imputations in `chl` and `log(chl)` consistent with each other, i.e., the imputations should respect their relationship. The following code will take care of this:

```
R> nhanes2.ext <- cbind(nhanes2, lchl = log(nhanes2$chl))
R> ini <- mice(nhanes2.ext, max = 0, print = FALSE)
R> meth <- ini$meth
R> meth["lchl"] <- "~log(chl)"
R> pred <- ini$pred
R> pred[c("hyp", "chl"), "lchl"] <- 0
R> pred["bmi", "chl"] <- 0
R> pred
```

```
     age bmi hyp chl lchl
age    0   0   0   0    0
bmi    1   0   1   0    1
hyp    1   1   0   1    0
```

```
chl    1   1   1   0    0
lchl   1   1   1   1    0
```

```
R> imp <- mice(nhanes2.ext, meth = meth, pred = pred, seed = 38788,
+    print = FALSE)
R> head(complete(imp))
```

```
    age  bmi hyp chl     lchl
1 20-39 35.3  no 218 5.384495
2 40-59 22.7  no 187 5.231109
3 20-39 30.1  no 187 5.231109
4 60-99 22.5 yes 218 5.384495
5 20-39 20.4  no 113 4.727388
6 60-99 22.7  no 184 5.214936
```

We defined the predictor matrix such that either `chl` or `log(chl)` is a predictor, but not both at the same time, primarily to avoid collinearity. Moreover, we do not want to predict `chl` from `lchl`. Doing so would immobilize the MICE algorithm at the starting imputation. It is thus important to set the entry `pred["chl", "lchl"]` equal to zero to avoid this. After running `mice()` we find imputations for both `chl` and `lchl` that respect the relation.

Note: A slightly easier way to create `nhanes2.ext` is to specify

```
R> nhanes2.ext <- cbind(nhanes2, lchl = NA)
```

followed by the same commands. This has the advantage that the transform needs to be specified only once. Since all values in `lchl` are now treated as missing, the size of `imp` will generally become (much) larger however. The first method is generally more efficient, but the second is easier.

*Index of two variables*

The idea can be extended to two or more columns. This is useful to create derived variables that should remain synchronized. As an example, we consider imputation of body mass index (bmi), which is defined as weight divided by height*height. It is impossible to calculate bmi if either weight or height is missing. Consider the data `boys` in **mice**.

```
R> md.pattern(boys[, c("hgt", "wgt", "bmi")])
```

```
    wgt hgt bmi
727   1   1   1  0
 17   1   0   0  2
  1   0   1   0  2
  3   0   0   0  3
      4  20  21 45
```

Data on weight and height are missing for 4 and 20 cases, respectively, resulting in 21 cases for which `bmi` could not be calculated. Using passive imputation, we can impute `bmi` from height and weight by means of the `I()` operator.

```
R> ini <- mice(boys, max = 0, print = FALSE)
R> meth <- ini$meth
R> meth["bmi"] <- "~I(wgt/(hgt/100)^2)"
R> pred <- ini$pred
R> pred[c("wgt", "hgt", "hc", "reg"), "bmi"] <- 0
R> pred[c("gen", "phb", "tv"), c("hgt", "wgt", "hc")] <- 0
R> pred
```

```
    age hgt wgt bmi hc gen phb tv reg
age   0   0   0   0  0   0   0  0   0
hgt   1   0   1   0  1   1   1  1   1
wgt   1   1   0   0  1   1   1  1   1
bmi   1   1   1   0  1   1   1  1   1
hc    1   1   1   0  0   1   1  1   1
gen   1   0   0   1  0   0   1  1   1
phb   1   0   0   1  0   1   0  1   1
tv    1   0   0   1  0   1   1  0   1
reg   1   1   1   0  1   1   1  1   0
```

The predictor matrix prevents that `hgt` or `wgt` are imputed from `bmi`, and takes care that there are no cases where `hgt`, `wgt` and `bmi` are simultaneous predictors. Passive imputation overrules the selection of variables specified in the `predictorMatrix` argument. Thus, in the above case, we might have well set `pred["bmi",] <- 0` and obtain identical results. Imputations can now be created by

```
R> imp.idx <- mice(boys, pred = pred, meth = meth, maxit = 20,
+    seed = 9212, print = FALSE)
R> head(complete(imp.idx)[is.na(boys$bmi), ], 3)
```

```
       age  hgt   wgt      bmi   hc gen phb tv   reg
103  0.087 60.0  4.54 12.61111 39.0  G1  P1  3  west
366  0.177 57.5  4.20 12.70321 40.4  G1  P1  1  west
1617 1.481 85.5 12.04 16.47002 47.5  G1  P1  1 north
```

Observe than the imputed values for `bmi` are consistent with (imputed) values of `hgt` and `wgt`.

Note: The values of `bmi` in the original data have been rounded to two decimals. If desired, one can get that also in the imputed values by setting

```
R> meth["bmi"] <- "~round(wgt/(hgt/100)^2,dig=2)"
```

*Sum scores*

The sum score is undefined if one of the variables to be added is missing. We can use sum scores of imputed variables within the MICE algorithm to economize on the number of predictors. For example, suppose we create a summary maturation score of the pubertal measurements `gen`, `phb` and `tv`, and use that score to impute the other variables instead of the three original pubertal measurements. We can achieve that by

```
R> ini <- mice(cbind(boys, mat = NA), max = 0, print = FALSE)
R> meth <- ini$meth
R> meth["mat"] <- "~I(as.integer(gen) + as.integer(phb) +\n
+    + as.integer(cut(tv,breaks=c(0,3,6,10,15,20,25))))"
R> meth["bmi"] <- "~I(wgt/(hgt/100)^2)"
R> pred <- ini$pred
R> pred[c("bmi", "gen", "phb", "tv"), "mat"] <- 0
R> pred[c("hgt", "wgt", "hc", "reg"), "mat"] <- 1
R> pred[c("hgt", "wgt", "hc", "reg"), c("gen", "phb", "tv")] <- 0
R> pred[c("wgt", "hgt", "hc", "reg"), "bmi"] <- 0
R> pred[c("gen", "phb", "tv"), c("hgt", "wgt", "hc")] <- 0
R> pred

    age hgt wgt bmi hc gen phb tv reg mat
age   0   0   0   0  0   0   0  0   0   0
hgt   1   0   1   0  1   0   0  0   1   1
wgt   1   1   0   0  1   0   0  0   1   1
bmi   1   1   1   0  1   1   1  1   1   0
hc    1   1   1   0  0   0   0  0   1   1
gen   1   0   0   1  0   0   1  1   1   0
phb   1   0   0   1  0   1   0  1   1   0
tv    1   0   0   1  0   1   1  0   1   0
reg   1   1   1   0  1   0   0  0   0   1
mat   0   0   0   0  0   0   0  0   0   0
```

The maturation score `mat` is composed of the sum of `gen`, `phb` and `tv`. Since the first two are factors, we need the `as.integer()` function to get the internal numerical codes. Furthermore, we recoded `tv` into 6 ordered categories by calling the `cut()` function, and use the category number to calculate the sum score. The predictor matrix is set up so that either the set of `(gen,phb,tv)` or `mat` are predictors, but never at the same time. The number of predictors for say, `hgt`, has now dropped from 8 to 5, but imputation still incorporates the main relations of interest. Imputations can now be generated and plotted by

```
R> imp.sum <- mice(cbind(boys, mat = NA), pred = pred, meth = meth,
+    maxit = 20, seed = 10948, print = FALSE)
R> xyplot(imp.sum, mat ~ age | .imp, na = gen | phb | tv,
+    subset = .imp == 1, ylab = "Maturation score", xlab = "Age (years)")
```

Figure 5 plots the derived maturation scores against age. Since no measurements were made before the age of 8 years, all scores on the left side are sums of three imputed values for `gen`, `phb` and `tv`. Note that imputation relies on extreme extrapolation outside the range of the data. Though quite a few anomalies are present (many babies score a '4' or higher), the overall pattern is as expected. Section 3.5 discusses ways to improve the imputations.

*Interaction terms*

In some cases scientific interest focusses on interactions terms. For example, in experimental studies we may be interested in assessing whether the rate of change differs between two

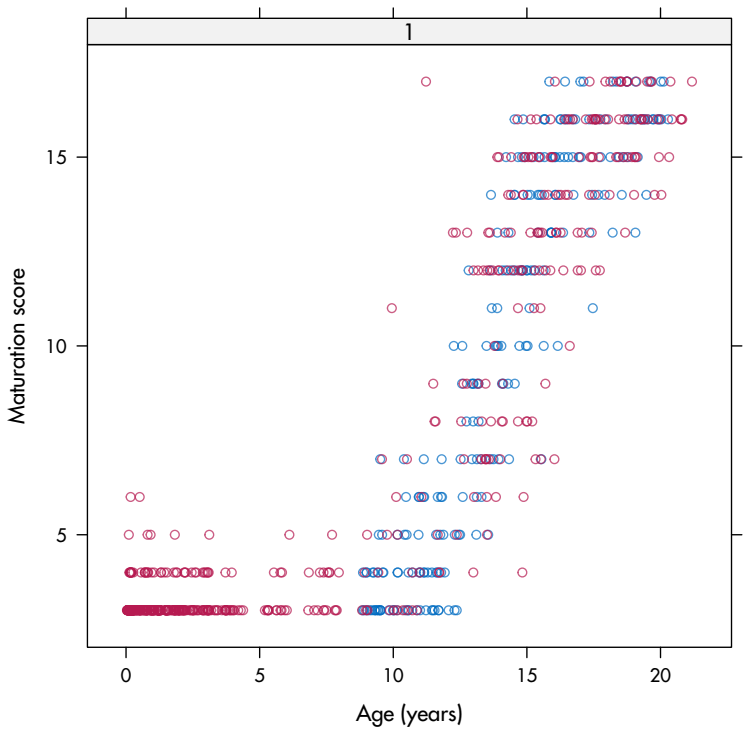


Figure 5: Observed (blue) and (partially) imputed (red) maturation scores plotted against age.

treatment groups. In such cases, the primary goal is to get an unbiased estimate of the time by group interaction. In general imputations should be conditional upon the interactions of interest. However, interaction terms will be incomplete if the variables that make up the interaction are incomplete. It is straightforward to solve this problem using passive imputation.

Interactions between two continuous variables are often defined by subtracting the mean and taking the product. In **mice** 2.9 we may say

```
R> nhanes2.ext <- cbind(nhanes2, bmi.chl = NA)
R> ini <- mice(nhanes2.ext, max = 0, print = FALSE)
R> meth <- ini$meth
R> meth["bmi.chl"] <- "~I((bmi-25)*(chl-200))"
R> pred <- ini$pred
R> pred[c("bmi", "chl"), "bmi.chl"] <- 0
R> imp <- mice(nhanes2.ext, meth = meth, pred = pred, seed = 51600,
+    print = FALSE)
```

Imputations created in this way preserve the interaction of `bmi` with `chl`. This would be useful if the complete-data model is to predict, for example, `hyp` from `bmi` and `chl` and their interaction.

Interactions involving categorical variables need a representation using dummy variables. The

`mice()` function internally creates dummy variables for any factor that are being used as a predictor. The data and names of these dummy variables can be accessed from `imp$pad$data`. In the above example, we find

```
R> head(ini$pad$data, 3)

    age  bmi  hyp chl bmi.chl age.1 age.2 hyp.1
1 20-39   NA <NA>  NA      NA     0     0    NA
2 40-59 22.7   no 187      NA     1     0     0
3 20-39   NA   no 187      NA     0     0     0
```

The factors `age` and `hyp` are internally represented by dummy variables `age.1`, `age.2` and `hyp.1`. The interaction between `age` and `bmi` can be added as

```
R> nhanes2.ext <- cbind(nhanes2, age.1.bmi = NA, age.2.bmi = NA)
R> ini <- mice(nhanes2.ext, max = 0, print = FALSE)
R> meth <- ini$meth
R> meth["age.1.bmi"] <- "~I(age.1*(bmi-25))"
R> meth["age.2.bmi"] <- "~I(age.2*(bmi-25))"
R> pred <- ini$pred
R> pred[c("age", "bmi"), c("age.1.bmi", "age.2.bmi")] <- 0
R> imp <- mice(nhanes2.ext, meth = meth, pred = pred, maxit = 10)
```

Imputation of `hyp` and `chl` will now respect the interaction between `age` and `bmi`.

*Squeeze*

Imputed values that are implausible or impossible should not be accepted. For example, `mice.impute.norm()` can generate values outside the data range. Positive-valued variables could occasionally receive negative values. For example, the following code produces a crash:

```
R> nhanes2.ext <- cbind(nhanes2, lchl = NA)
R> ini <- mice(nhanes2.ext, max = 0, pri = FALSE)
R> meth <- ini$meth
R> meth[c("lchl", "chl")] <- c("~log(chl)", "norm")
R> pred <- ini$pred
R> pred[c("hyp", "chl"), "lchl"] <- 0
R> pred["bmi", "chl"] <- 0
R> imp <- mice(nhanes2.ext, meth = meth, pred = pred, seed = 1,
+    maxit = 100)

  ...
27   3  bmi  hyp  chl  lchl
27   4  bmi  hyp  chl  lchl
Error in `[<-.data.frame`(`*tmp*`, , i, value = list
(`log(chl)` = c(4.09912613113127,  :
  replacement element 1 has 24 rows, need 25
In addition: Warning message:
In log(chl) : NaNs produced
```

The problem here is that one of the imputed values in `chl` is negative. Negative values can occur when imputing under the normal model, but leads here to a fatal error. One way to prevent this error is to *squeeze* the imputations into an allowable range. The `squeeze()` function in **mice** 2.9 recodes any outlying values in the tail of the distribution to the nearest allowed value. Using

```
R> meth["lchl"] <- "~log(squeeze(chl, bounds=c(100,300)))"
R> imp <- mice(nhanes2.ext, meth = meth, pred = pred, seed = 1, maxit = 100)
```

will squeeze all imputed values into the range 100–300 before taking the `log`. This trick will solve the problem, but does not store any squeezed values in `chl`, so `lchl` and `chl` become inconsistent. Depending on the situation, this may or may not be a problem. One way to ensure consistency is to create an intermediate variable `schl` by passive imputation. Thus, `schl` contains the squeezed values, and takes over the role of `chl` within the algorithm. We will see an alternative in Section 3.5.

*Cautious remarks*

There are some specific points that need attention when using passive imputation through the `~` mechanism. Deterministic relations between columns remain only synchronized if the passively imputed variable is updated immediately after any of its predictors are imputed. So in the last example variables `age.1.bmi` and `age.2.bmi` should be updated each time after `age` or `bmi` is imputed in order to stay synchronized. This can be done by changing the sequence in which the algorithm visits the columns. The `mice()` function does not automatically change the visiting sequence if passive variables are added. Section 3.6 provides techniques for setting the visiting sequence. Whether synchronization is really worthwhile will depend on the specific data at hand, but it is a healthy general strategy to pursue.

The `~` mechanism may easily lead to highly correlated variables or linear dependencies among predictors. Sometimes we want this behavior on purpose, for example if we want to impute using both $X$ and $X^2$. However, linear dependencies among predictors will produce a fatal error during imputation. In this section, our strategy has been to avoid this by requiring that either the original or the passive variable can be a predictor. This strategy may not always be desired or feasible however.

Another point is that passive imputation may easily lock up the algorithm when it is not done properly. Suppose that we make a copy `bmi2` of `bmi` by passive imputation, and subsequently use `bmi2` to impute missing data in `bmi`. Re-imputing `bmi` from `bmi2` will fix the imputations to the starting imputations. This situation is easy to diagnose and correct (cf. Section 4.3).

The `mice` algorithm internally uses passive imputation to create dummy variables of factors. These dummy variables are created automatically and discarded within the main algorithm, and are always kept in sync with the original by passive imputation. The relevant data and settings are stored within the list `imp$pad`. Normally, the user will not have to deal with this, but in case of running problems it could be useful to be aware of this. Section 4 provides more details.

## 3.5. Post-processing imputations

It can be useful to post-process imputations generated by univariate methods. For example, we may require imputation to be bounded within a certain range, or we may wish to exclude

implausible or impossible combinations. The `mice()` function has an argument `post` that takes a vector of strings of R commands. These commands are parsed and evaluated just after the univariate imputation function returns, and thus provide a way to post-process the imputed values. For example, a way to ensure positive imputations for `chl` under normal imputation (cf. Section 3.4 on the `squeeze()` function) is:

```
R> nhanes2.ext <- cbind(nhanes2, lchl = NA)
R> ini <- mice(nhanes2.ext, max = 0, print = FALSE)
R> meth <- ini$meth
R> meth[c("lchl", "chl")] <- c("~log(chl)", "norm")
R> pred <- ini$pred
R> pred[c("hyp", "chl"), "lchl"] <- 0
R> pred["bmi", "chl"] <- 0
R> post <- ini$post
R> post["chl"] <- "imp[[j]][,i] <- squeeze(imp[[j]][,i],c(100,300))"
R> imp <- mice(nhanes2.ext, meth = meth, pred = pred, post = post,
+    seed = 30031, maxit = 10, print = FALSE)
R> imp$imp$chl
```

```
          1        2        3        4        5
1  141.9775 100.0000 215.5944 167.4770 132.2074
4  175.9424 197.7750 252.6138 270.0522 162.9603
10 203.5916 138.0413 190.6631 188.8077 160.8818
11 232.7851 100.0000 100.0000 171.7716 151.0727
12 210.0065 139.1416 199.3788 238.3034 128.7681
15 215.4298 160.8284 148.6562 197.7282 178.2769
16 172.3431 168.2527 161.2586 207.1777 140.7590
20 212.0838 226.3953 258.8964 213.4481 160.4901
21 203.1200 184.1227 123.5391 175.1849 126.5748
24 231.2208 255.7083 137.7759 260.2423 222.0734
```

The expression `imp[[j]][,i]` in the definition of `post["chl"]` refers to a vector that is used to store the $i$-th imputation ($i = 1, \ldots, m$) for the $j$-th column in `p$data`, a padded version of the input data, here `nhanes2.ext`. Expression(s) are evaluated within the `sampler()` function. Any expressions that are valid within that context can be executed, but be careful not the introduce any `NA`'s if the variable is to be used as a predictor for another variable. The output shows that several imputed values have been constrained to lie within the specified range.

Another example refers to Figure 5. Puberty can already start at the age of 3 years in clinical populations of American girls (Herman-Giddens *et al.* 1997). For our data of healthy Dutch boys we assume that puberty will not start before the age of 5 years. We thus want to restrict any imputations of `gen`, `phb` and `tv` to the lowest possible category for children younger than 5 years. This can be achieved by using the `post` argument. The code below first repeats the setting of `meth` and `pred` from Section 3.4.

```
R> ini <- mice(cbind(boys, mat = NA), max = 0, print = FALSE)
R> meth <- ini$meth
```

```
R> meth["mat"] <- "~I(as.integer(gen) + as.integer(phb) +\n
+    + as.integer(cut(tv,breaks=c(0,3,6,10,15,20,25))))"
R> meth["bmi"] <- "~I(wgt/(hgt/100)^2)"
R> pred <- ini$pred
R> pred[c("bmi", "gen", "phb", "tv"), "mat"] <- 0
R> pred[c("hgt", "wgt", "hc", "reg"), "mat"] <- 1
R> pred[c("hgt", "wgt", "hc", "reg"), c("gen", "phb", "tv")] <- 0
R> pred[c("wgt", "hgt", "hc", "reg"), "bmi"] <- 0
R> pred[c("gen", "phb", "tv"), c("hgt", "wgt", "hc")] <- 0
R> pred
```

```
    age hgt wgt bmi hc gen phb tv reg mat
age   0   0   0   0  0   0   0  0   0   0
hgt   1   0   1   0  1   0   0  0   1   1
wgt   1   1   0   0  1   0   0  0   1   1
bmi   1   1   1   0  1   1   1  1   1   0
hc    1   1   1   0  0   0   0  0   1   1
gen   1   0   0   1  0   0   1  1   1   0
phb   1   0   0   1  0   1   0  1   1   0
tv    1   0   0   1  0   1   1  0   1   0
reg   1   1   1   0  1   0   0  0   0   1
mat   0   0   0   0  0   0   0  0   0   0
```

```
R> post <- ini$post
R> post["gen"] <- "imp[[j]][p$data$age[!r[,j]]<5,i] <- levels(boys$gen)[1]"
R> post["phb"] <- "imp[[j]][p$data$age[!r[,j]]<5,i] <- levels(boys$phb)[1]"
R> post["tv"] <- "imp[[j]][p$data$age[!r[,j]]<5,i] <- 1"
R> imp <- mice(cbind(boys, mat = NA), pred = pred, meth = meth, post = post,
+    maxit = 10, print = FALSE)
```

The expression `p$data$age[!r[,j]]<5` will find the ages of the children for whom the current ($j$-th) variable was imputed. The maturation score `mat` now always takes it lowest value before the age of 5 years (cf. Figure 6). Section 6.2 contains another application of post-processing. You can write your own post-processing functions, and call these from within the MICE algorithm.

### 3.6. Visiting scheme

The default MICE algorithm imputes incomplete columns in the data from left to right. Theoretically, the visiting scheme is irrelevant as long as each column is visited often enough, but some schemes are more efficient than others. In particular, for monotonically missing data, convergence is immediate if variables are ordered according to their number of missing cases. Rather than reordering the data itself, it is more convenient to change the visiting scheme of the algorithm by the `visitSequence` argument. In its basic form, the `visitSequence` argument is a vector of integers in the range `1:ncol(data)` of arbitrary length, specifying the sequence of column numbers for one iteration of the algorithm. Any given column may be visited more than once within the same iteration, which can be useful to ensure proper

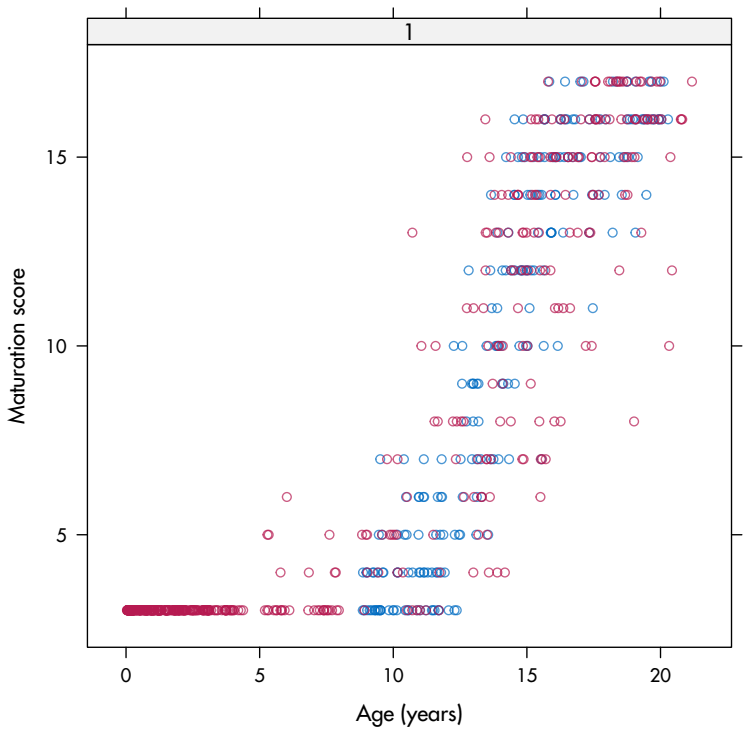


Figure 6: Observed (blue) and (partially) imputed (red) maturation scores plotted against age, where the imputed values for `gen`, `phb` and `tv` are constrained before the age of 5 years.

synchronization among variables. It is mandatory that all columns with missing data that are being used as predictors are visited, or else the function will stop with an error.

As an example, rerun the code of the Section 3.4, to obtain imputed data `imp` that allow for the interaction `bmi.chl`. The visiting scheme is

```
R> imp$vis

    bmi     hyp     chl bmi.chl
      2       3       4       5
```

If `visitSequence` is not specified, the `mice()` function imputes the data from left to right. In this case, `bmi.chl` is calculated after `chl` is imputed, so at point `bmi.chl` is synchronized with both `bmi` and `chl`. Note however that `bmi.chl` is not synchronized with `bmi` when imputing `hyp`, so `bmi.chl` is not representing the current interaction effect. This could result in wrong imputations. We can correct this by including an extra visit to `bmi.chl` after `bmi` has been imputed:

```
R> vis <- imp$vis
R> vis <- append(vis, vis[4], 1)
R> vis
R> imp <- mice(nhanes2.ext, meth = meth, pred = pred, vis = vis)
```

```
 iter imp variable
  1   1  bmi  bmi.chl  hyp  chl  bmi.chl
  1   2  bmi  bmi.chl  hyp  chl  bmi.chl
  1   3  bmi  bmi.chl  hyp  chl  bmi.chl
 ...
```

The effect is that `bmi.chl` is now properly updated. By the way, a more efficient ordering of the variables is

```
R> imp <- mice(nhanes2.ext, meth = meth, pred = pred, vis = c(2, 4, 5, 3))
```

```
 iter imp variable
  1   1  bmi  chl  bmi.chl  hyp
  1   2  bmi  chl  bmi.chl  hyp
  1   3  bmi  chl  bmi.chl  hyp
...
```

When the missing data pattern is close to monotone, convergence may be speeded by visiting the columns in increasing order of the number of missing data. We can specify this order by the `"monotone"` keyword as

```
R> imp <- mice(nhanes2.ext, meth = meth, pred = pred, vis = "monotone")
```

```
 iter imp variable
  1   1  hyp  bmi  chl  bmi.chl
  1   2  hyp  bmi  chl  bmi.chl
  1   3  hyp  bmi  chl  bmi.chl
 ...
```

# 4. Running MICE

## 4.1. Dry run

A *dry run* is a call to `mice()` with the maximum number of iterations `maxit` set to zero by

```
R> ini <- mice(nhanes2, maxit = 0)
```

A dry run is a fast way to create the `mids` object `ini` containing the default settings. The default settings of the attributes of this `mids` object, like `ini$method`, `ini$predictorMatrix`, `ini$post`, and `ini$visitSequence` can be used to define user-specific settings. Especially for datasets with many variables this is easier than defining these settings from scratch. This technique was already used in many examples in Section 3.

A `mids` object obtained with a dry run can also be used to include manually imputations obtained from other software. It is essential that external imputations are stored in the proper format of the `mids` object. For example, the matrix `import` contains two imputations for nine missing values in `bmi` generated by other software. It can be assigned to the `mids` object `imp` by

```
R> import <- matrix(c(30, 30, 30, 29, 25, 21, 25, 25, 22, 33, 27, 22, 27,
+    35, 27, 20, 27, 30), byrow = TRUE, nr = 9)
R> imp <- mice(nhanes, print = FALSE, seed = 77172)
R> imp$imp$bmi[, 1:2] <- import
R> imp$imp$bmi

    1  2    3    4    5
1  30 30 22.0 33.2 29.6
3  30 29 28.7 22.0 27.4
4  25 21 22.5 24.9 22.0
6  25 25 24.9 24.9 24.9
10 22 33 27.2 28.7 24.9
11 27 22 35.3 29.6 22.5
12 27 35 27.4 26.3 30.1
16 27 20 35.3 22.0 25.5
21 27 30 33.2 29.6 20.4
```

It is important to realize that this technique assumes that the order of imputed values across software systems is identical. Section 7 shows some alternative ways to interact with other software.

The altered `mids` object can now be used as input for the MICE algorithm by calling the `mice.mids` function (see Section 4.2). Note that this requires all empty cells to be imputed, otherwise the sampler will get stuck on empty cells. Also, the altered `mids` object can be used for repeated complete data analyses by calling the `with.mids` function. For this use, empty cells in the imputation can be encoded by `NA`. The complete data method should then be able to handle the missing data correctly.

## 4.2. Step by step

The function `mice.mids()` takes a `mids` object as input, iterates `maxit` iterations and produces another `mids` object as output. This function enables the user to split up the computations of the MICE algorithm into smaller parts by providing a stopping point after every full iteration. There are various circumstances in which this might be useful:

- For large data, RAM memory may become exhausted if the number of iterations is large. Returning to prompt/session level may alleviate these problems.
- The user wants to compute special convergence statistics at intermediate points, e.g., after each iteration, for monitoring convergence.
- For computing a 'few extra iterations'.

The imputation model itself is specified in the `mice()` function and cannot be changed with `mice.mids` (Well actually you can by tweaking the `mids` object directly, but this is not recommended). The state of the random generator is saved with the `mids` object.

As a simple example, calculate two `mids` objects `imp2` and `imp` as

```
R> imp <- mice(nhanes, maxit = 4, seed = 44612, print = FALSE)
R> imp1 <- mice(nhanes, maxit = 1, seed = 44612, print = FALSE)
R> a <- runif(10)
R> imp2 <- mice.mids(imp1, maxit = 3, print = FALSE)
```

Imputations in `imp` and `imp2` are identical so

```
R> all(imp$imp$bmi == imp2$imp$bmi)
```

```
[1] TRUE
```

should yield `TRUE`.

### 4.3. Assessing convergence

There is no clear-cut method for determining whether the MICE algorithm has converged. What is often done is to plot one or more parameters against the iteration number. The `mice()` function produces $m$ parallel imputation streams. The `mids` object contains components `chainMean` and `chainVar` with the mean and variance of the imputations per stream, respectively. These can be plotted by the `plot.mids` object. On convergence, the different streams should be freely intermingled with each other, without showing any definite trends. Convergence is diagnosed when the variance between different sequences is no larger than the variance with each individual sequence.

Inspection of the stream may reveal particular problems of the imputation model. Section 3.4 shows that passive imputation should carefully set the predictor matrix. The code below is a pathological example where the MICE algorithm is stuck at the initial imputation.

```
R> ini <- mice(boys, max = 0, print = FALSE)
R> meth <- ini$meth
R> meth["bmi"] <- "~I(wgt/(hgt/100)^2)"
R> meth["wgt"] <- "~I(bmi*(hgt/100)^2)"
R> meth["hgt"] <- "~I(100*sqrt(wgt/bmi))"
R> imp1 <- mice(boys, meth = meth, maxit = 20, print = FALSE, seed = 9212)
```

Variables `hgt`, `wgt` and `bmi` have an exact nonlinear relationships. The above code simultaneously specifies all relationships between them. The net effect is that the sampler locks up. Figure 7 is created by

```
R> plot(imp1, c("hgt", "wgt", "bmi"))
```

and indicates that the mean (on the left) and the standard deviation (on the right) of the imputation for the three variables remain constant.

A less extreme but still pathological case of non-convergence occurs with the following code:

```
R> ini <- mice(boys, max = 0, print = FALSE)
R> meth <- ini$meth
R> meth["bmi"] <- "~I(wgt/(hgt/100)^2)"
R> imp2 <- mice(boys, meth = meth, maxit = 20, print = FALSE, seed = 9212)
```

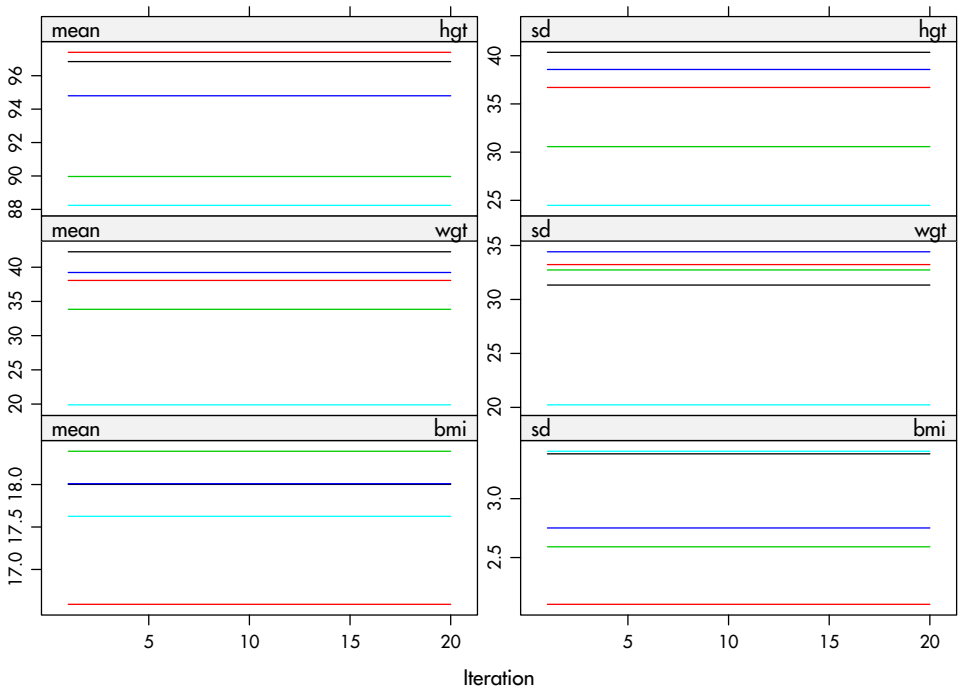


Figure 7: Pathological example of non-convergence of the MICE algorithm. Plotted are the means and standard deviations per iteration of the imputed values of `hgt`, `wgt` and `bmi`. The values are locked to the starting imputation.

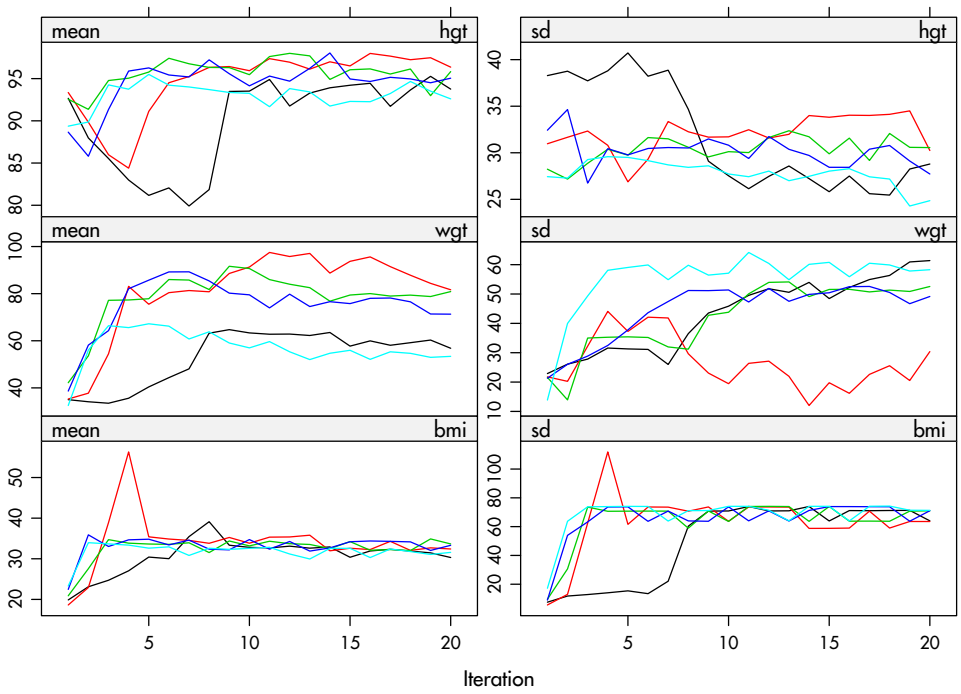


Figure 8: Non-convergence of the MICE algorithm. Imputations for `hgt`, `wgt` and `bmi` hardly mix and resolve into a steady state.

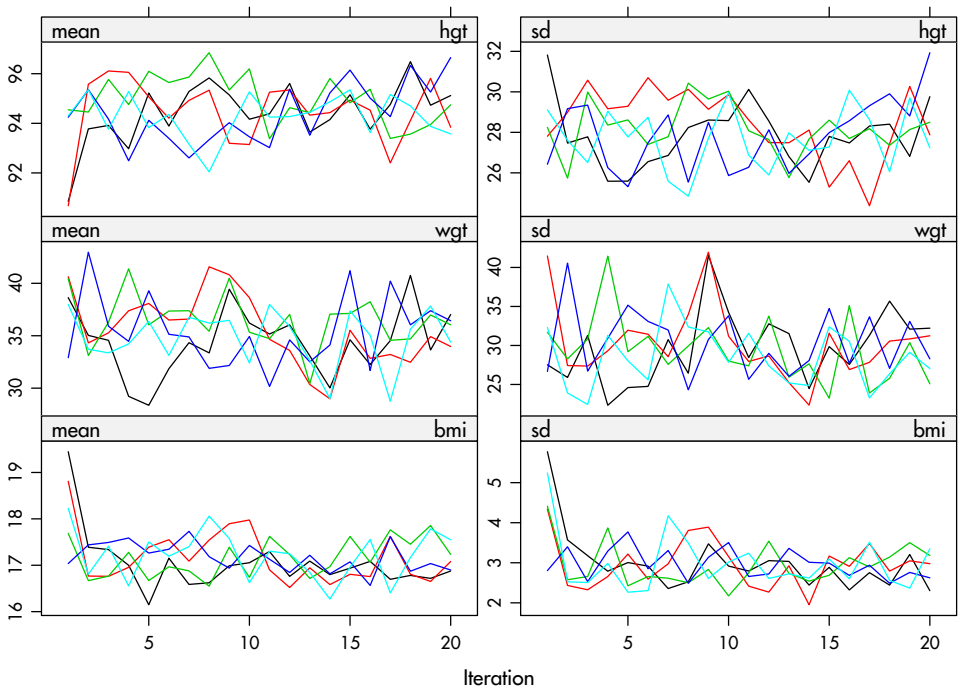


Figure 9: Healthy convergence of the MICE algorithm for `hgt`, `wgt` and `bmi`, where feedback loop of `bmi` into `hgt` and `wgt` is broken (solution `imp.idx`).

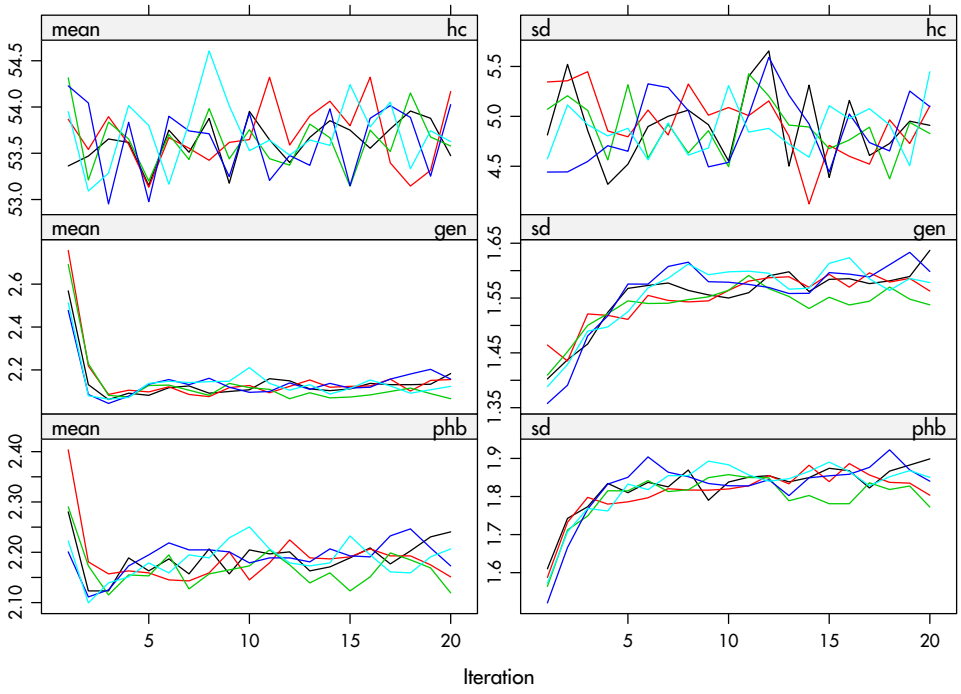


Figure 10: Healthy convergence of the MICE algorithm for `hc`, `gen` and `phb` showing a strong initial trend for the latter two (solution `imp.idx`).

Convergence is bad because imputations of `bmi` feed back into `hgt` and `wgt`. Figure 8 shows that the streams hardly mix and slowly resolve into a steady state.

By comparison, Figure 9 is an example of healthy convergence of the same three variables. The model fitted here is the `imp.idx` solution from Section 3.4, where `bmi` is passively imputed without feedback into `hgt` and `wgt`. There is very little trend and the streams mingle very well right from the start.

Figure 10 shows the convergence plot for `hc`, `gen` and `phb` from the same solution. There is a strong initial trend for `gen` and `phb`). The MICE algorithm initializes the solution by taking a random draw from the observed data. For `gen` and `phb` these values are far too high. The algorithm quickly picks up the strong relation between `age`, `gen` and `phb`, and has more or less reached the appropriate level within about five iterations. This demonstrates that convergence can be very fast, even if the starting imputations are clearly off-target. Plotting somewhat longer iteration sequences will generally convey a good idea whether the between-imputation variability has stabilized and whether the estimates are free of trend.

Note that one iteration of the MICE algorithm involves a lot of work. For each variable and each repeated imputation a statistical model is fitted, imputations are drawn and the data are updated. Fortunately, the needed number of main iterations is typically much lower than is common in modern MCMC techniques, which often require thousands of iterations. The key to fast convergence is to achieve independence in the imputations themselves. Univariate imputation procedures create imputations that are already statistically independent for a given value of the regression parameters. The number of iterations should be large enough to stabilize the distributions of these parameters. Simulation work using moderate amounts of missing data yields satisfactory performance with just 5 or 10 iterations (Brand 1999; van Buuren *et al.* 2006b). In many cases, we can obtain good results with as few iterations, but it does not hurt to calculate some extra iterations to assess convergence over longer stretches. For large amounts of missing data or for remotely connected data (e.g., file matching) convergence can be slower.

The default plot of the `mids` object plots the mean and variance of the imputations. This may not correspond to the parameter of most interest. To check convergence of an arbitrary parameter of interest, one could write a function that loops over `mice.mids`, that extracts imputed data with the complete function, and that recomputes the parameter after each iteration using the current imputations. More in particular, one could set maxit to 1, generate imputations, compute the statistic of interest on the completed data, save the result, compute the second iteration using `mice.mids`, and so on.

As a example, suppose that we are interested in the association between `gen` and `phb` as measured by Kendall's $\tau$. We can monitor convergence of the MICE algorithm with respect to Kendall's $\tau$ by the following lines:

```
R> m <- 5
R> T <- 20
R> imp.kendall <- mice(boys, m = m, meth = imp.idx$meth,
+    pred = imp.idx$pred, maxit = 0, print = FALSE)
R> tau <- matrix(NA, nrow = T, ncol = m)
R> for (i in 1:T) {
+    if (i == 1) set.seed(9212)
+    imp.kendall <- mice.mids(imp.kendall, maxit = 1, print = FALSE)
```

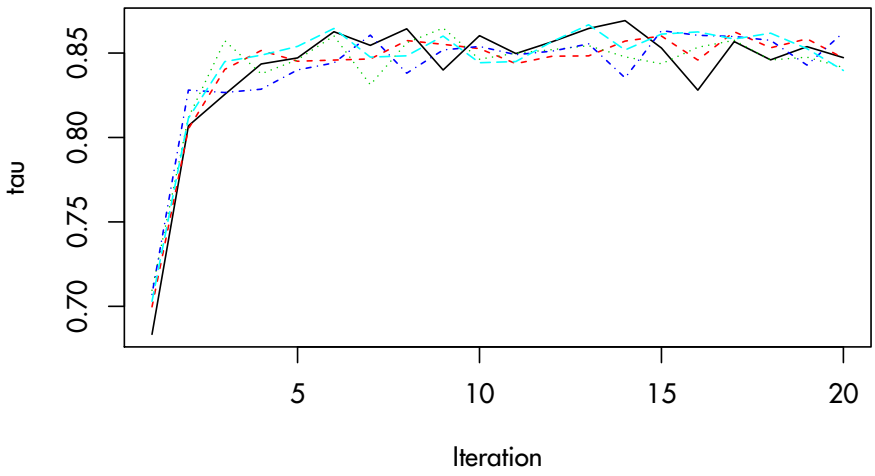


Figure 11: Convergence of the MICE algorithm with respect to Kendall's $\tau$ between `gen` and `phb`.

```
+    x <- complete(imp.kendall, "repeated")[, paste("gen", 1:m, sep = ".")]
+    y <- complete(imp.kendall, "repeated")[, paste("phb", 1:m, sep = ".")]
+    xn <- as.data.frame(lapply(x, as.numeric))
+    yn <- as.data.frame(lapply(y, as.numeric))
+    tau[i, ] <- diag(cor(xn, yn, method = "kendall"))
+  }
```

The development of $\tau$ can now be plotted by:

```
R> matplot(x = 1:T, y = tau, xlab = "Iteration", type = "l")
```

Figure 11 shows that diagnostic convergence plot for the association between `gen` and `phb`. In general, the pattern is comparable to that in Figure 10 where 5–10 iterations are needed to be free of trend.

### 4.4. Solving problems with the data

The MICE algorithm performs checks on the data before: missing data should be present, the data consists of at least two columns, and constant predictors are not allowed. Version V2.0 implements more stable algorithms for imputing categorical data, which caused a lot of warnings in previous versions.

One source of confusion is that the `mice()` function chooses the imputation method according to the type of the variable. Categorical variables should be coded as factors in order to invoke `polyreg`, otherwise `pmm` will be selected. Variables with many categories ($> 20$) are often problematic. Internally, the `mice()` function will create a dummy variable for each category, which can produce very large matrices, time consuming calculations, introduce empty cell

problems and cause instability problems of the algorithm. For variable with many categories, we therefore recommend `pmm`.

The `mice()` function was programmed for flexibility, and not to minimize the use of time or resources. Combined with the greedy nature of R in general and the fact that the method does not use compiled functions, this poses some limits to the size of the data sets and the type of data that can be analyzed. The excution time of `mice()` depends primarily on the choice of the univariate imputation methods. Methods like `pmm` and `norm` are fast, while others, `logreg` and `polyreg`, can be slow. If there are many categorical variables, each with many categories, one can speed up the algorithm considerably by imputing (some of the) variables as numerical variables with `pmm` instead of `polyreg`. Since `pmm` imputes only values that are observed, the original categories of the variable are preserved.

A frequent source of problems is caused by predictors that are (almost) linearly related. For example, if one predictor can be written as a linear combination of some others, then this results in messages like `Error in Solve.default(t(xobs) %*% xobs) : system is computationally singular: reciprocal condition number = 2.87491e-20` or `Error in solve.default(t(xobs) %*% xobs) : Lapack routine dgesv: system is exactly singular.` The solution to collinearity is to eliminate duplicate information, for example by specifying a reduced set of predictors via the `predictorMatrix` argument. Sometimes the source of the problem is obvious, for example if there are sum scores. However, finding the sets of nearly dependent variables can prove to be difficult and laborious. One trick that we use in practice is to study the last eigenvector of the covariance matrix of the incomplete data (after listwise deletion). Variables with high values on that factor often cause the problems. Alternatively, one could revert to collinearity diagnostics like the VIF (Kleinbaum *et al.* 1988) or graphical displays (Cook and Weisberg 1999). Pedhazur (1973) provided a good discussion on multicollinearity. Version 2.5 incorporated new functions for trapping multicollinearity, so these problems should be things of the past.

If all fails, use the debugger. Functions in **mice** are all full R. This allows you to trace every detail of your calculations, and eventually track down the source of the problem. This is your last resort, and requires you to understand our code, which can be difficult at times, even for us. Use `traceback()` to find out what R was doing at the time the error occurred. In order to use the debugger set `options(error = dump.frames)` before the error occurs, and then type `debugger()`. This will get you to choose the local frame where the error, and inspect the current value of local variables. If you have suggestions for improvement, please let us know.

## 4.5. Checking your imputations

Once the algorithm has converged, it is important to inspect the imputations. In general, a good imputed value is a value that could have been observed had it not been missing. In this phase, it is often useful to report results to the investigator who understands the science behind the data. Since there are some many aspects that we could look at, we have refrained from implemented specific tools. It is however not difficult with the powerful graphical functions in **lattice** to make diagnostic plots for checking the imputations. We will give some examples below, following the ideas of Raghunathan and Bondarenko (2007).

The MAR assumption can never be tested from the observed data. One can however check whether the imputations created by MICE algorithm are plausible. As a first step one can

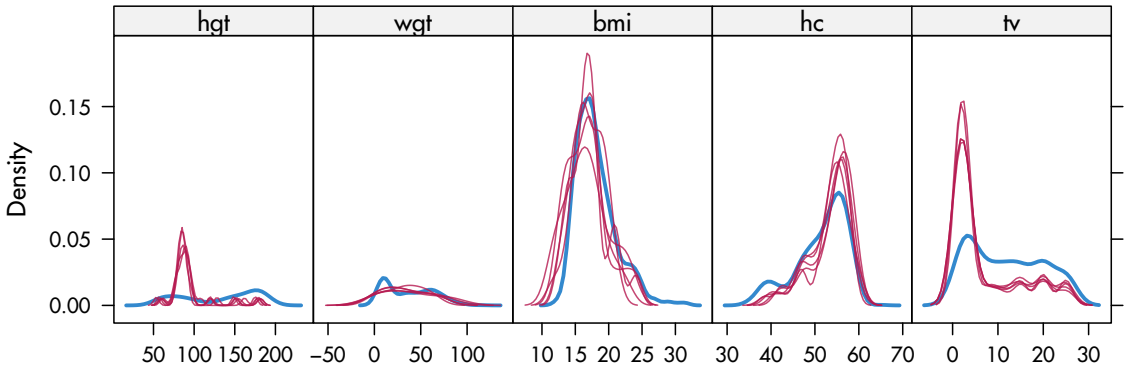


Figure 12: Kernel density estimates for the marginal distributions of the observed data (blue) and the $m = 5$ densities per variable calcuted from the imputed data (thin red lines).

plot densities of both the observed and imputed values of all variables to see whether the imputations are reasonable. Differences in the densities between the observed and imputed values may suggest a problem that needs to be further checked. The `densityplot()` function from the **lattice** package can be used on `mids` objects to produce Figure 12 as

```
R> densityplot(imp.kendall, scales = list(x = list(relation = "free")),
+    layout = c(5, 1))
```

Figure 12 shows that the imputed values can be quite different from the observed data. For example, the imputed heights are around 90 cm, which is due to the fact the some of the values of the two-year olds were missing. The same holds for testicular volume (`tv`), which was not measured below the age of 8 years. Reversely, the imputed values for head circumference (`hc`) are higher since `hc` was not measured in the older boys. In general, plots like this are useful to detect interesting differences between the observed and imputed data.

Another diagnostic tool involves comparing the distributions of observed and imputed data conditional on the propensity score (Raghunathan and Bondarenko 2007). The idea is that the conditional distributions should be similar if the assumed model for creating multiple imputations is a good fit. The following statements create the missing data indicator `hc.na` of `hc` in the global environment, and calculate propensity scores:

```
R> hc.na <- is.na(boys$hc)
R> fit.hc <- with(imp.kendall, glm(hc.na ~ age + wgt + hgt + reg,
+    family = binomial))
R> ps <- rep(rowMeans(sapply(fit.hc$analyses, fitted.values)), 6)
```

Figure 13 is plotted by

```
R> xyplot(imp.kendall, hc ~ ps | .imp, pch = c(1, 20), cex = c(0.8, 1.2),
+    xlab = "Probability that head circumference is missing",
+    ylab = "Head circumference (cm)", scales = list(tick.number = 3))
```

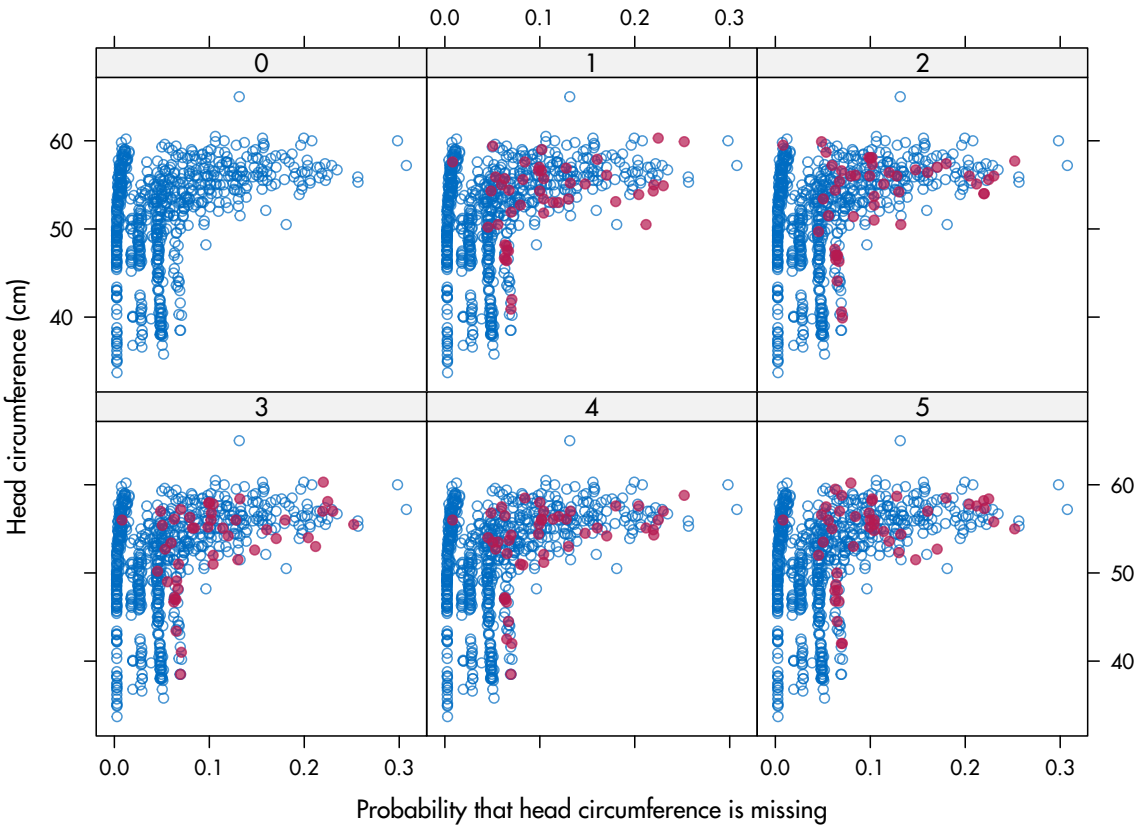


Figure 13: Head circumference against its (pooled) propensity score for observed and imputed values.

The figure plots head circumference (observed and imputed) against the propensity score, where the propensity score is equal to the average over the imputations. By definition, there are more imputed values on the right hand side of the display than on the left hand side. The striped pattern in the distribution of the propensity score is related to region (variable `reg`). More missing data appear in the city and in the western part of the country. If the imputation model fits well, we expect that for a given propensity score, the distribution of the observed and imputed data conform well. This appears to be the case here, so this plot provides evidence that the imputations are reasonable. Realize however that the comparison is as good as the propensity score. If important predictors are omitted from the nonresponse model, then we cannot see the potential misfit related to these.

It is also useful to study the residual of the relation depicted in Figure 13. Under MAR we expect that the spread of the residuals will be similar (but not identical) for observed and imputed data, so their distributions should overlap. The relation between the propensity score and head circumference is clearly not linear, so we added polynomial terms to account for the nonlinearity.

```
R> hc <- complete(imp.kendall, "long", TRUE)$hc
R> fit <- lm(hc ~ poly(ps, 4))
```

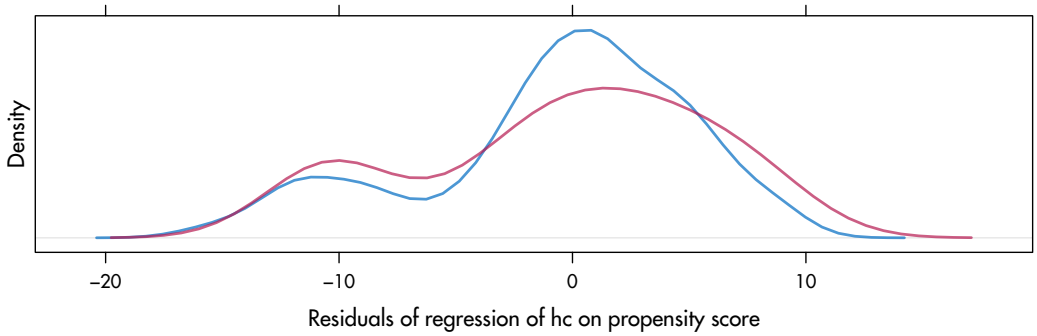


Figure 14: Distributions of the residuals of the missing data model.

Figure 14 is created by

```
R> densityplot(~residuals(fit), group = hc.na, plot.points = FALSE,
+     ref = TRUE, scales = list(y = list(draw = FALSE)),
+     par.settings = simpleTheme(col.line = rep(mdc(1:2))),
+     xlab = "Residuals of regression of hc on propensity score", lwd = 2)
```

The figure displays the distributions of the residuals in the observed and imputed data. The amount of overlap is large, lending credit to the notion that the spread of imputations is appropriate.

More methods for diagnostic checking can be found in Section 2.4. Also consult methods for checking multiply imputed values by Abayomi *et al.* (2008). These are also easily implemented using standard tools.

# 5. After MICE

Imputations are stored in an object of class `mids`. The next step is to analyze the multiply imputed data by the model of scientific interest. The function `with.mids()` takes imputed data, fits the model to each of the $m$ imputed data sets, and returns an object of type `mira`. Section 5.1 deals with this step. The result is $m$ analysis instead of one, so the last task is to combine these analyses into one final final result. The function `pool()` takes an object of class `mira` and produces a combined result in an object of class `mipo`. Section 5.3 provides the details.

## 5.1. Repeated data analysis

Performing the desired analysis repeatedly for each imputed copy of the data can be done with the function `with.mids`. This function evaluates an expression in each multiply imputed dataset and creates an object of class `mira`. We impute `nhanes2` and apply a linear regression on the imputed data to predict `chl` as follows:

```
R> imp <- mice(nhanes2, seed = 99210, print = FALSE)
```

```
R> fit <- with(imp, lm(chl ~ age + bmi))
R> summary(pool(fit))

                   est        se          t       df    Pr(>|t|)
(Intercept) -19.671875 54.092442 -0.3636714 16.14412 0.720822014
age2         50.381076 17.928842  2.8100575 16.83937 0.012127888
age3         68.702368 19.868766  3.4578076 13.61617 0.003988504
bmi           6.777794  1.927845  3.5157357 14.26303 0.003340391
                 lo 95     hi 95 nmis       fmi     lambda
(Intercept) -134.25959  94.91584   NA 0.2070037 0.11449428
age2          12.52702  88.23513   NA 0.1840057 0.09252328
age3          25.97513 111.42961   NA 0.2889235 0.19162353
bmi            2.65012  10.90547    9 0.2677970 0.17185238
```

We can access the results of the third imputation by `fit$ana[[3]]`. The `with.mids()` function is called by just `with()`. The function accepts a valid R expression as its second argument, and applies it to each completed data set. For example, to calculate the contingency table of overweight by age per imputation, we can use the following:

```
R> expr <- expression(ov <- cut(bmi, c(10, 25, 50)), table(age, ov))
R> fit <- with(imp, eval(expr))
```

The contingency tables corresponding to the imputed data sets 2 and 5 are obtained as:

```
R> fit$an[c(2, 5)]

[[1]]
       ov
age     (10,25] (25,50]
  20-39       4       8
  40-59       2       5
  60-99       2       4

[[2]]
       ov
age     (10,25] (25,50]
  20-39       3       9
  40-59       3       4
  60-99       2       4
```

The function `with.mids()` is completely general, and replaces `lm.mids()` and `glm.mids()`. For compatibility reasons, the latter two functions remain available.

## 5.2. Extracting imputed data

An alternative is to export the imputed data into a conventional rectangular form, followed by the analysis of the imputed data. The `complete()` function extracts imputed data sets from a `mids` object, and returns the completed data as a data frame. For example,

```
R> com <- complete(imp, 3)
```

extracts the third complete data set from the multiply imputed data in `imp`. Specifying

```
R> com <- complete(imp, "long")
```

produces a long matrix `com` where the $m$ completed data matrices are vertically stacked and padded with the imputation number in a column called `.imp`. This form is convenient for making point estimates and for exporting multiply imputed data to other software. Other options are `broad` and `repeated`, which produce complete data in formats that are convenient for investigating between-imputation patterns. One could also optionally include the original incomplete data.

An alternative way of creating the above contingency tables using `complete()` is:

```
R> com <- complete(imp, "long", include = TRUE)
R> by(cbind(age = com$age, ov = cut(com$bmi, c(10, 25, 50))),
+    com$.imp, table)

INDICES: 0
   ov
age 1 2
  1 2 5
  2 2 3
  3 2 2
------------------------------------------------------
INDICES: 1
   ov
age 1 2
  1 3 9
  2 4 3
  3 2 4
------------------------------------------------------
INDICES: 2
   ov
age 1 2
  1 4 8
  2 2 5
  3 2 4
...
```

The first contingency table labeled `INDICES: 0` corresponds to the original data, the table labeled `INDICES: 1` to the first imputed data set, and so on.

### 5.3. Pooling

Rubin (1987) developed a set of rules for combining the separate estimates and standard errors from each of the $m$ imputed datasets into an overall estimate with standard error,

confidence intervals and $p$ values. These rules are based on asymptotic theory on the normal distribution, and are implemented in the functions `pool()` and `pool.scalar()`.

`mira` *objects*

The function `pool()` take an object of class `mira` and creates an object of class `mipo` (multiple imputed pooled outcomes). For example,

```
R> fit <- with(imp, lm(chl ~ age + bmi))
R> est <- pool(fit)

Call: pool(object = fit)

Pooled coefficients:
(Intercept)        age2        age3         bmi
 -19.671875   50.381076   68.702368    6.777794

Fraction of information about the coefficients missing due to nonresponse:
(Intercept)        age2        age3         bmi
  0.2070037   0.1840057   0.2889235   0.2677970
```

More detailed output can be obtained by `summary(pool(fit))`.

The function `pool()` works for any object having both `coef()` and `vcov()` methods. The function tests for this, and aborts if it fails to find an appropriate method. A list of methods for which `coef()` and `vcov()` exists can be obtained by

```
R> methods(coef)

 [1] coef.Arima*          coef.aov*            coef.default*
 [4] coef.fitdistr*       coef.lda*            coef.listof*
 [7] coef.loglm*          coef.multinom*       coef.nls*
[10] coef.nnet*           coef.powerTransform* coef.ridgelm*

   Non-visible functions are asterisked

R> methods(vcov)

 [1] vcov.Arima*          vcov.cch*            vcov.coxph*
 [4] vcov.fitdistr*       vcov.glm*            vcov.glmrob*
 [7] vcov.lm*             vcov.lmrob*          vcov.mlm*
[10] vcov.multinom*       vcov.negbin*         vcov.nls*
[13] vcov.polr*           vcov.powerTransform* vcov.rlm*
[16] vcov.survreg*

   Non-visible functions are asterisked
```

In addition, the `pool()` function will also work for objects of class `lme` defined in the package **nlme**. It is possible to pool the fixed coefficients from a linear mixed model according to Rubin's rules. By default the number of degrees of freedom is calculated using the method of Barnard and Rubin (1999).

*Scalars*

The function `pool.scalar` pools univariate estimates of $m$ repeated complete data analysis according to Rubin's rules. The arguments of the function are two vectors containing the $m$ repeated complete data estimates and their corresponding $m$ variances. The function returns a list containing the pooled estimate, the between, within and total variance, the relative increase in variance due to nonresponse, the degrees of freedom for the $t$ reference distribution and the fraction of missing information due to nonresponse. This function is useful when the estimated parameters are not obtained through one of the regular R modeling functions.

*Explained variance $R^2$*

For combining estimates of $R^2$ and adjusted $R^2$ one can use the function `pool.r.squared`. The method is based on the familiar Fisher $z$ transformation for correlations. Its properties in the context of multiple imputation were studied by Harel (2009).

*Model testing*

The function `pool.compare()` compares two nested models fitted on multiply imputed data. The function implements the Wald test and the likelihood ratio test. The function can be used to test whether one or more variables should be present in the complete-data model. Variable selection in the context of multiple imputation is somewhat different. Several strategies have been proposed that count the number of times that variable is in the model (Brand 1999; Heymans *et al.* 2007; Wood *et al.* 2008). The `pool.compare()` function provides an alternative that takes the between-imputation variability into account.

The Wald test can be used when the completed-data estimates and their covariance matrices are known (e.g., for estimates obtained with `lm`), and when the dimensionality of the estimates is not too high. The `pool.compare()` function with the argument `method = "Wald"` pools the $p$ values for comparing the nested models using the method of Li *et al.* (1991).

```
R> imp <- mice(nhanes2, print = FALSE, m = 50, seed = 219)
R> fit0 <- with(data = imp, expr = lm(bmi ~ age + hyp))
R> fit1 <- with(data = imp, expr = lm(bmi ~ age + hyp + chl))
R> stat <- pool.compare(fit1, fit0, method = "Wald")
R> stat$p

             [,1]
[1,] 0.005015501
```

Meng and Rubin (1992) provide a procedure for testing nested hypotheses by likelihood ratio tests from multiple imputed data. The likelihood function needs to be fully specified in order to calculate the likelihood ratio statistics at the average over the imputations of the parameter estimates under both the null and alternative hypotheses. The current version of

`pool.compare()` implements the likelihood function for logistic regression, i.e., complete-data models obtained with `glm(family = "binomial")`. The example below illustrates the use of the method in the boys data. The question is whether the factor `reg` (region, with five levels) should be included in the logistic regression model for onset of puberty.

```
R> imp <- mice(boys, print = FALSE, seed = 60019)
R> fit0 <- with(data = imp,
+    expr = glm(I(gen > levels(gen)[1]) ~ hgt + hc, family = binomial))
R> fit1 <- with(data = imp,
+    expr = glm(I(gen > levels(gen)[1]) ~ hgt + hc + reg, family = binomial))
R> stat <- pool.compare(fit1, fit0, method = "likelihood", data = imp)
R> stat$p

[1] 0.2022771
```

The difference as measured by the likelihood ratio statistic is not significant. The number of imputation is a bit low for this problem. Rerunning the imputations with $m = 50$ yields a $p$ value of 0.1293, so the contribution of `reg` is not statistically significant.

# 6. Miscellaneous topics

## 6.1. Adding your own imputation functions

Some organizations have made considerable investments to develop procedures for imputing key variables, like income or family size, whose values are subject to all kinds of subtle constraints. Using one of the built-in imputation methods could be a waste of this investment, and may fail to produce what is needed.

It is possible to write your own univariate imputation function, and call this function from within the MICE algorithm. The easiest way to write such a function is to copy and modify an existing `mice.impute.xxx()` function, for example `mice.impute.norm()`. Most univariate imputation functions have just three arguments: the variable to be imputed `y`, the response indicator `ry` and the matrix of predictors `x` (without intercept). The function should return a vector of `length(y)-sum(ry)` imputations of the correct type. Consult `mice.impute.norm()` or `mice.impute.polyreg()` for inspiration. Your new function can be called from within the MICE algorithm by the `method` argument. For example, calling your function `mice.impute.myfunc()` for each column can be done by typing

```
R> mice(nhanes, method = "myfunc")
```

Using this procedure enables you to speed up the algorithm by including pre-compiled Fortran or C code, or to dump imputation models for closer inspection.

## 6.2. Sensitivity analysis under MNAR

A common misunderstanding about multiple imputation is that it is restricted to MAR. While it is certainly true that imputation techniques commonly assume MAR, the theory of multiple imputation is completely general and also applies to MNAR. Under MNAR, the model fitted to the complete cases is incorrect for the missing cases, and thus cannot be used

for imputation. Unless we have external data, there is no way of estimating the amount of error.

A sensible alternative is to set up a number of plausible scenarios, and investigate the consequences of each of them on the final inferences. Chapter 6 in Rubin (1987) contains a number of basic techniques. Up-to-date overviews of specialized MNAR models can be found in Little (2009b) and Albert and Follmann (2009). If the influence under these scenarios is small, then the analysis is said to be robust against the investigated violations of the MAR mechanism.

Suppose that we have reason to believe that our imputations made under the MAR assumption are too low, even after accounting for the predictive information in the available data. One simple trick is to multiply the imputations by a factor. With `mice()`, this is easily achieved by post-processing imputations through the `post` argument. The scenario involves increasing imputations for `chl` by 0% (MAR), 10%, 20%, 30%, 40% and 50% (MNAR).

```
R> ini <- mice(nhanes2, maxit = 0, print = FALSE)
R> post <- ini$post
R> k <- seq(1, 1.5, 0.1)
R> est <- vector("list", length(k))
R> for (i in 1:length(k)) {
+    post["chl"] <- paste("imp[[j]][,i] <-", k[i], "* imp[[j]][,i]")
+    imp <- mice(nhanes2, post = post, seed = 10, print = FALSE, maxit = 20)
+    fit <- with(imp, lm(bmi ~ age + chl))
+    est[[i]] <- summary(pool(fit)) }
```

The parameters of interest under these six scenarios are stored in the list `est`. Inspection of these solutions reveals that the size of $k$ primarily increases the intercept term. There is a slightly trend that moves the model estimates towards zero if $k$ goes up. Note that we used fixed `seed` value, so all differences between scenarios are strictly due to $k$.

Carefully monitor convergence if variables are highly correlated. A higher imputed value in `chl` will produce a higher than average imputation in `bmi` and `hyp`. If relations are strong, these higher average will feedback into a higher imputations for `chl`, which is then multiplied by $k$, and so on. Such feedback could result in explosive behavior of the MICE algorithm that should be diagnosed and prevented. A remedy is to remove the strongest predictors from the imputation model.

Little (2009a) stresses that all MNAR models are subject to a fundamental lack of identification. More fancy models do not make this problem go away. He therefore advocates the use of simple models, like multiplying by a factor (Rubin 1987, pp. 203) or adding offsets (van Buuren *et al.* 1999). Both are basic forms of pattern-mixture models. Little (2009a, pp. 49) writes: "The idea of adding offsets is simple, transparent, and can be readily accomplished with existing software."

# 7. Interacting with other software

## 7.1. Microsoft Excel

Microsoft's **Excel** is the most widely tool to store and manipulate data. **Excel** has strong facilities for interactive data editing. We use the software in combination with **mice** to specify

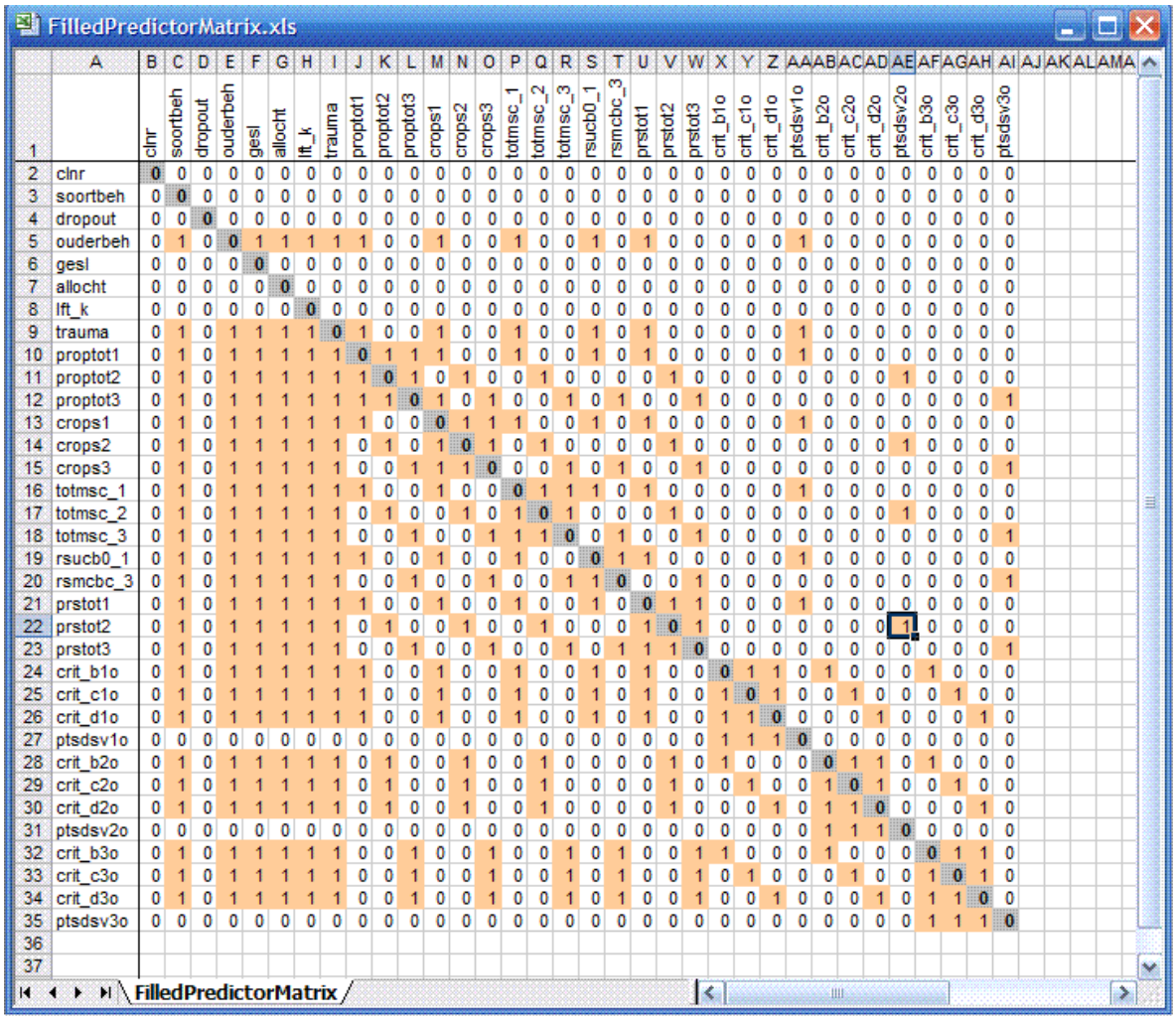


Figure 15: A predictor matrix in **Excel** for an imputation model of longitudinal data.

to predictor matrix for imputation problems that involve many variables. Figure 15 shows an example for an imputation problem for longitudinal data, where we have used to Excel's conditional formatting option for automatic coloring. It is straightforward to transfer the predictor matrix between **Excel** and R by simple tab-delimited files.

## 7.2. SPSS

SPSS 17 (now called PASW Statistics 17) implements the FCS approach to multiple imputation. See the documentation on the new `MULTIPLE IMPUTATION` procedure (SPSS Inc. 2008b,a) for details. The procedure is largely based on FCS, and the functionality of **mice** and the `MULTIPLE IMPUTATION` procedure has many overlaps. A difference is that SPSS has no support for passive imputation or post-processing, though it is possible to specify bounds.

The procedure in SPSS produces a long data set that is recognized by other procedures in SPSS as a multiply imputed data set. Application of general regression models like linear, logistic, GEE analysis, mixed model, Cox regression and general linear models automatically

gives pooled estimates of the coefficients, standard errors and $p$ values. Diagnostics like $R^2$, comparison of models using the likelihood ratio, are not pooled. It is also possible to import a multiply imputed data set into SPSS so that SPSS recognizes it as such. The key to importing data is to create a variable named `Imputation_`, and use this as split variable. The function `mids2spss()` in **mice** 2.9 converts a `mids` object into a format recognized by SPSS, and writes the data and the SPSS syntax files. Multiply imputed data can exported from SPSS to R. Converting this data set into a `mids` object has not yet been automated.

SPSS and R form a strong combination. SPSS has superior data handling and documentation functions, whereas R is free, extensible and flexible. Starting from SPSS 16.0, it is possible to integrate both software packages by the SPSS Integration Plug-In. A small program to impute data stored in SPSS with `mice()` looks like this (run this from the SPSS syntax window).

```
BEGIN PROGRAM R.
dict <- spssdictionary.GetDictionaryFromSPSS()
data <- spssdata.GetDataFromSPSS()

library("mice")
imp <- mice(data, maxit = 10)
com <- complete(imp, "long", inc = TRUE)
com <- cbind(com, Imputation_ = as.integer(com$.imp) - 1)

spssdictionary.SetDictionaryToSPSS("com", dict)
spssdata.SetDataToSPSS("com", com)
spssdictionary.EndDataStep()
END PROGRAM.
```

The code instructs R to import the data and the dictionary from SPSS. Subsequently **mice** is loaded, the data are imputed. Finally, the stacked imputed data and its dictionary is exported back from R into SPSS.

### 7.3. mitools

**mitools** 2.0.1 is a small R package available from CRAN written by Lumley (2010) containing tools for analyzing and pooling multiply imputed data. The multiply imputed data should be of class `ImputationList`. A `mids` object (`imp`) can be transformed into an object of class `ImputationList` object as follows:

```
R> library("mitools")
R> mydata <- imputationList(lapply(1:5, complete, x = imp))
```

Statistical analyses on these data can be done by `with()`:

```
R> fit <- with(mydata, expr = lm(chl ~ age + bmi))
```

The pooled outcomes are obtained with `summary(MIcombine(fit))`. `MIcombine()` can be used for all `fit` objects for which a `vcov()` function exists, like our function `pool()`. Since this function does not exists for objects of class `lme`, pooling of mixed models is not possible with **mitools**. The package itself has no functions for creating imputations.

### 7.4. Zelig

The package **Zelig** 3.4-6 (Imai *et al.* 2008) offers a framework with a simple structure and syntax that encompasses a number of existing statistical methods. One of the functions of **Zelig** is `mi()` which makes a list of imputed data sets of class `mi`. **Zelig** does not contain functions for generating multiple imputations. When a data set of class `mi` is passed to the general modeling function `zelig()`, it is recognized as an imputed data set. For a number of modeling functions in **Zelig** the outcomes are then automatically pooled when the data is such a list of imputed data sets. It is however not clear for which modeling function this works (`ls.mixed()` does not seem to be supported). The code for estimating the propensity score model for `hc` (cf. Section 4.5) with `zelig()` is as follows:

```
R> library("Zelig")
R> imp <- cbind.mids(imp.idx, data.frame(r.hc = is.na(boys$hc)))
R> mydata2 <- mi(complete(imp, 1), complete(imp, 2), complete(imp, 3),
+    complete(imp, 4), complete(imp, 5))
R> fit <- zelig(r.hc ~ age + wgt + hgt + bmi + gen + phb + tv + reg,
+    model = "logit", data = mydata2)
R> summary(fit)


  Model: logit
  Number of multiply imputed data sets: 5

Combined results:

Call:
zelig(formula = r.hc ~ age + wgt + hgt + bmi + gen + phb + tv +
    reg, model = "logit", data = mydata2)

Coefficients:
                    Value Std. Error      t-stat    p-value
(Intercept) -4.166824779 3.61385156 -1.15301492 0.24971471
age          0.075074606 0.15218856  0.49329992 0.62512416
wgt         -0.007125900 0.04740758 -0.15031140 0.88064148
....
```

The result of `zelig()` is not as detailed as `pool()`. The confidence intervals of the estimates and the fraction of missing information are not printed.

### 7.5. mi

**mi** 0.09-14 is an R package for multiple imputation (Su *et al.* 2011). The imputation method used in **mi** is also based on the FCS principle, and as in **Zelig** is called `mi()`. The package contains modeling functions that pull together the estimates (point estimates and their standard errors) from multiply imputed data sets for several models like `lm()`, `glm()` and `lmer()`. The package is quite extensive, and complements our package in some aspects, e.g., other facilities for diagnostic checking. The internal structure of class `mi` and class `mids` is different, but it

would certainly be useful to have conversion functions `mi2mids()` and `mids2mi()` in order to be able to combine the strengths of both packages.

## 8. Conclusion

The principle of fully conditional specification (FCS) has now gained wide acceptance. Good software is now available. Many applications using FCS have appeared, and many more will follow. This paper documents a major update of MICE. FCS has recently been adopted and implemented by SPSS, and was advertised by SPSS as the major methodological improvement of SPSS Statistics 17.

In the years to come, attention will shift from computational issues to the question how we can apply the methodology in a responsible way. We need guidelines on how to report MI, we need a better understanding of the dangers and limitations of the technique, we need pooling methods for special distributions, and we need entry-level texts that explain the idea and that demonstrate how to use the techniques in practice. Assuming that this all happens, multiple imputation using FCS will prove to be a great addition to our statistical tool chest.

## Acknowledgements

Many people have been helpful in converting the original S-PLUS library into the R package **mice**. We thank Jason Turner, Peter Malewski, Frank E. Harrell, John Fox and Roel de Jong for their valuable contributions.

## References

Abayomi K, Gelman A, Levy M (2008). "Diagnostics for Multivariate Imputations." *Journal of the Royal Statistical Society C*, **57**(3), 273–291.

Adamczyk A, Palmer I (2008). "Religion and Initiation Into Marijuana Use: The Deterring Role of Religious Friends." *Journal of Drug Issues*, **38**(3), 717–741.

Alati R, Mamun AA, Williams GM, O'callaghan M, Najman JM, Bor W (2006). "In Utero Alcohol Exposure and Prediction of Alcohol Disorders in Early Adulthood: A Birth Cohort Study." *Archives of General Psychiatry*, **63**(9), 1009–1016.

Albert PS, Follmann D (2009). "Shared-Parameter Models." In G Fitzmaurice, M Davidian, G Verbeke, G Molenberghs (eds.), *Longitudinal Data Analysis*, chapter 18, pp. 433–452. CRC Press, Boca Raton, FL.

Ambler G, Omar RZ, Royston P, Kinsman R, Keogh BE, Taylor KM (2005). "Generic, Simple Risk Stratification Model for Heart Valve Surgery." *Circulation*, **112**(2), 224–231.

Arnold BC, Castillo E, Sarabia JM (1999). *Conditional Specification of Statistical Models*. Springer-Verlag, New York.

Arnold BC, Press SJ (1989). "Compatible Conditional Distributions." *Journal of the American Statistical Association*, **84**, 152–156.

Barnard J, Rubin DB (1999). "Small Sample Degrees of Freedom with Multiple Imputation." *Biometrika*, **86**, 948–955.

Barosi G, Bergamaschi G, Marchetti M, Vannucchi AM, Guglielmelli P, Antonioli E, Massa M, Rosti V, Campanelli R, Villani L, Viarengo G, Gattoni E, Gerli G, Specchia G, Tinelli C, Rambaldi A, Barbui T (2007). "JAK2 V617F Mutational Status Predicts Progression to Large Splenomegaly and Leukemic Transformation in Primary Myelofibrosis." *Blood*, **110**(12), 4030–4036.

Brand JPL (1999). *Development, Implementation and Evaluation of Multiple Imputation Strategies for the Statistical Analysis of Incomplete Data Sets*. Ph.D. thesis, Erasmus University, Rotterdam.

Briggs A, Clark T, Wolstenholme J, Clarke P (2003). "Missing... Presumed at Random: Cost-Analysis of Incomplete Data." *Health Economics*, **12**(5), 377–392.

Brunner EJ, Chandola T, Marmot MG (2007). "Prospective Effect of Job Strain on General and Central Obesity in the Whitehall II Study." *American Journal of Epidemiology*, **165**(7), 828–837.

Burton A, Billingham LJ, Bryan S (2007). "Cost-Effectiveness in Clinical Trials: Using Multiple Imputation to Deal with Incomplete Cost Data." *Clinical Trials*, **4**(2), 154–161.

Butler RJ, Heron J (2008). "The Prevalence of Infrequent Bedwetting and Nocturnal Enuresis in Childhood." *Scandinavian Journal of Urology and Nephrology*, **42**(3), 257–264.

Byrne J, Spence MS, Fretz E, Mildenberger R, Chase A, Berry B, Pi D, Janssen C, Klinke P, Hilton D (2009). "Body Mass Index, Periprocedural Bleeding, and Outcome Following Percutaneous Coronary Intervention (from the British Columbia Cardiac Registry)." *American Journal of Cardiology*, **103**(4), 507–511.

Caria MP, Bellocco R, Zambon A, Horton NJ, Galanti MR (2009). "Overweight and Perception of Overweight as Predictors of Smokeless Tobacco Use and of Cigarette Smoking in a Cohort of Swedish Adolescents." *Addiction*, **104**(4), 661–668.

Chamberlain LJ, Crowley M, Tope D, Hodson R (2008). "Sexual Harassment in Organizational Context." *Work and Occupations*, **35**(3), 262–295.

Chase AJ, Fretz EB, Warburton WP, Klinke WP, Carere RG, Pi D, Berry B, Hilton JD (2008). "Association of the Arterial Access Site at Angioplasty with Transfusion and Mortality: The M.O.R.T.A.L Study (Mortality Benefit of Reduced Transfusion after Percutaneous Coronary Intervention via the Arm or Leg)." *Heart*, **94**(8), 1019–1025.

Clark TG, Altman DG (2003). "Developing a Prognostic Model in the Presence of Missing Data: An Ovarian Cancer Case Study." *Journal of Clinical Epidemiology*, **56**(1), 28–37.

Clark TG, Bradburn MJ, Love SB, Altman DG (2003). "Survival Analysis Part IV: Further Concepts and Methods in Survival Analysis." *British Journal of Cancer*, **89**(5), 781–786.

Clark TG, Stewart ME, Altman DG, Gabra H, Smyth JF (2001). "A Prognostic Model for Ovarian Cancer." *British Journal of Cancer*, **85**(7), 944–952.

Collins LM, Schafer JL, Kam CM (2001). "A Comparison of Inclusive and Restrictive Strategies in Modern Missing Data Procedures." *Psychological Methods*, **6**(3), 330–351.

Cook RD, Weisberg S (1999). *Applied Regression Including Computing and Graphics*. John Wiley & Sons, New York.

Cottrell G, Mary JY, Barro D, Cot M (2005). "Is Malarial Placental Infection Related to Peripheral Infection at Any Time of Pregnancy?" *American Journal of Tropical Medicine and Hygiene*, **73**(6), 1112–1118.

Cottrell G, Mary JY, Barro D, Cot M (2007). "The Importance of the Period of Malarial Infection During Pregnancy on Birth Weight in Tropical Africa." *American Journal of Tropical Medicine and Hygiene*, **76**(5), 849–854.

Cummings P, Rivara FP, Olson CM, Smith KM (2006). "Changes in Traffic Crash Mortality Rates Attributed to Use of Alcohol, or Lack of a Seat Belt, Air Bag, Motorcycle Helmet, or Bicycle Helmet, United States, 1982-2001." *Injury Prevention*, **12**(3), 148–154.

Deave T, Heron J, Evans J, Emond A (2008). "The Impact of Maternal Depression in Pregnancy on Early Child Development." *BJOG: An International Journal of Obstetrics and Gynaecology*, **115**(8), 1043–1051.

Drechsler J, Rassler S (2008). "Does Convergence Really Matter?" In Shalabh, C Heumann (eds.), *Recent Advances in Linear Models and Related Areas – Essays in Honour of Helge Toutenburg*, pp. 341–355. Springer-Verlag, Berlin.

Fernandes AS, Jarman IH, Etchells TA, Fonseca JM, Biganzoli E, Bajdik C, Lisboa PJG (2008). "Missing Data Imputation in Longitudinal Cohort Studies – Application of PlannArd in Breast Cancer Survival." In *Proceedings – 7th International Conference on Machine Learning and Applications, ICMLA 2008*, pp. 644–649.

Finke R, Adamczyk A (2008). "Cross-National Moral Beliefs: The Influence of National Religious Context." *Sociological Quarterly*, **49**(4), 617–652.

Garabed RB, Johnson WO, Gill J, Perez AM, Thurmond MC (2008). "Exploration of Associations Between Governance and Economics and Country Level Foot-and-Mouth Disease Status by Using Bayesian Model Averaging." *Journal of the Royal Statistical Society A*, **171**(3), 699–722.

Gelman A (2004). "Parameterization and Bayesian Modeling." *Journal of the American Statistical Association*, **99**(466), 537–545.

Gelman A, Raghunathan TE (2001). "Discussion of Arnold et al. 'Conditionally Specified Distributions'." *Statistical Science*, **16**, 249–274.

Gerestein CG, Eijkemans MJC, Jong DD, van der Burg MEL, Dykgraaf RHM, Kooi GS, Baalbergen A, Burger CW, Ansink, C A (2009). "The Prediction of Progression-Free and Overall Survival in Women with an Advanced Stage of Epithelial Ovarian Carcinoma." *BJOG: An International Journal of Obstetrics and Gynaecology*, **116**(3), 372–380.

Giorgi R, Belot A, Gaudart J, Launoy G, the French Network of Cancer Registries FRANCIM (2008). "The Performance of Multiple Imputation for Missing Covariate Data within the Context of Regression Relative Survival Analysis." *Statistics in Medicine*, **27**(30), 6310–6331.

Goldstein H, Carpenter JR, Kenward MG, Levin KA (2009). "Multilevel Models with Multivariate Mixed Response Types." *Statistical Modelling*, **9**(3), 173–179.

Grote FK, Oostdijk W, Keizer-Schrama SMM, Dekker FW, van Dommelen P, van Buuren S, van der Kooij AML, Verkerk, H P, Wit JM (2007). "Referral Patterns of Children with Poor Growth in Primary Health Care." *BMC Public Health*, **7**, 77.

Grote FK, van Dommelen P, Oostdijk W, Keizer-Schrama SMM, Verkerk PH, Wit JM, van Buuren S (2008). "Developing Evidence-Based Guidelines for Referral for Short Stature." *Archives of Disease in Childhood*, **93**(3), 212–217.

Harel O (2009). "The Estimation of $R^2$ and Adjusted $R^2$ in Incomplete Data Sets Using Multiple Imputation." *Journal of Applied Statistics*, **36**(10), 1109–1118.

Harrell FE (2001). *Regression Modeling Strategies, with Applications to Linear Models, Logistic Regression, and Survival Analysis*. Springer-Verlag, New York.

Hartog JED, Lardenoije CMJG, Severens JL, Land JA, Evers JLH, Kessels AGH (2008). "Screening Strategies for Tubal Factor Subfertility." *Human Reproduction*, **23**(8), 1840–1848.

Heckerman D, Chickering DM, Meek C, Rounthwaite R, Kadie C (2001). "Dependency Networks for Inference, Collaborative Filtering, and Data Visualisation." *Journal of Machine Learning Research*, **1**, 49–75.

Herman-Giddens ME, Slora EJ, Wasserman RC, Bourdony CJ, Bhapkar MV, Koch CG, Hasemeier CM (1997). "Secondary Sexual Characteristics and Menses in Young Girls Seen in Office Practice: A Study from the Pediatric Research in Office Settings Network." *Pediatrics*, **99**(4), 505–512.

Heymans MW, van Buuren S, Knol DL, van Mechelen W, Vet HCWD (2007). "Variable Selection under Multiple Imputation Using the Bootstrap in a Prognostic Study." *BMC Medical Research Methodology*, **7**, 33.

Hill JL, Reiter JP, Zanutto EL (2004). "A Comparison of Experimental and Observational Data Analyses." In A Gelman, XL Meng (eds.), *Applied Bayesian Modeling and Causal Inference from Incomplete-Data Perspectives*, chapter 5, pp. 49–60. John Wiley & Sons, New York.

Hille ETM, Elbertse L, Gravenhorst JN, Brand R, Verloove-Vanhorick SP (2005). "Nonresponse Bias in a Follow-Up Study of 19-Year-Old Adolescents Born as Preterm Infants." *Pediatrics*, **116**(5).

Hille ETM, Weisglas-Kuperus N, van Goudoever JB, Jacobusse GW, Ens-Dokkum MH, Groot LD, Wit JM, Geven WB, Kok JH, Kleine MJKD, KollÉE LAA, Mulder ALM, van Straaten

HLM, Vries LSD, van Weissenbruch MM, Verloove-Vanhorick SP (2007). "Functional Outcomes and Participation in Young Adulthood for Very Preterm and Very Low Birth Weight Infants: The Dutch Project on Preterm and Small for Gestational Age Infants at 19 Years of Age." *Pediatrics*, **120**(3).

Hindorff LA, Rice KM, Lange LA, Diehr P, Halder I, Walston J, Kwok P, Ziv E, Nievergelt C, Cummings SR, Newman AB, Tracy RP, Psaty BM, Reiner AP (2008). "Common Variants in the CRP Gene in Relation to Longevity and Cause-Specific Mortality in Older Adults: The Cardiovascular Health Study." *Atherosclerosis*, **197**(2), 922–930.

Horton NJ, Kleinman KP (2007). "Much Ado About Nothing: A Comparison of Missing Data Methods and Software to Fit Incomplete Data Regression Models." *The American Statistician*, **61**(1), 79–90.

Horton NJ, Lipsitz SR (2001). "Multiple Imputation in Practice: Comparison of Software Packages for Regression Models with Missing Variables." *The American Statistician*, **55**, 244–254.

Horwood J, Salvi G, Thomas K, Duffy L, Gunnell D, Hollis C, Lewis G, Menezes P, Thompson A, Wolke D, Zammit S, Harrison G (2008). "Iq and Non-Clinical Psychotic Symptoms in 12-Year-Olds: Results from the Alspac Birth Cohort." *British Journal of Psychiatry*, **193**(3), 185–191.

Hox JJ (2002). *Multilevel Analysis. Techniques and Applications*. Lawrence Erlbaum Associates, Mahwah, NJ.

Huo D, Adebamowo CA, Ogundiran TO, Akang EE, Campbell O, Adenipekun A, Cummings S, Fackenthal J, Ademuyiwa F, Ahsan H, Olopade OI (2008). "Parity and Breastfeeding Are Protective Against Breast Cancer in Nigerian Women." *British Journal of Cancer*, **98**(5), 992–996.

Imai K, King G, Lau O (2008). "Toward a Common Framework for Statistical Analysis and Development." *Journal of Computational and Graphical Statistics*, **17**(4), 1–22.

Ip EH, Wang YJ (2009). "Canonical Representation of Conditionally Specified Multivariate Discrete Distributions." *Journal of Multivariate Analysis*, **100**(6), 1282–1290.

Jacobusse GW (2005). ***WinMICE*** *User's Manual*. TNO Quality of Life, Leiden. URL http://www.multiple-imputation.com/.

Jensen M, Roy A (2008). "Staging Exchange Partner Choices: When Do Status and Reputation Matter?" *Academy of Management Journal*, **51**(3), 495–516.

Kasim RM, Raudenbush SW (1998). "Application of Gibbs Sampling to Nested Variance Components Models with Heterogeneous Within-Group Variance." *Journal of Educational and Behavioral Statistics*, **23**(2), 93–116.

Kekitiinwa A, Lee KJ, Walker AS, Maganda A, Doerholt K, Kitaka SB, Asiimwe A, Judd A, Musoke P, Gibb DM (2008). "Differences in Factors Associated with Initial Growth, Cd4, and Viral Load Responses to Art in HIV-Infected Children in Kampala, Uganda, and the United Kingdom/Ireland." *Journal of Acquired Immune Deficiency Syndromes*, **49**(4), 384–392.

Kennickell AB (1991). "Imputation of the 1989 Survey of Consumer Finances: Stochastic Relaxation and Multiple Imputation." *ASA 1991 Proceedings of the Section on Survey Research Methods*, pp. 1–10.

Klein LW, Kolm P, Xu X, Krone RJ, Anderson HV, Rumsfeld JS, Brindis RG, Weintraub WS (2009). "A Longitudinal Assessment of Coronary Interventional Program Quality – A Report from the American College of Cardiology-National Cardiovascular Data Registry." *JACC: Cardiovascular Interventions*, **2**(2), 136–143.

Klein MB, Mack CD, Kramer CB, Heimbach DM, Gibran NS, Rivara FP (2008). "Influence of Injury Characteristics and Payer Status on Burn Treatment Location in Washington State." *Journal of Burn Care and Research*, **29**(3), 435–440.

Kleinbaum DG, Kupper LL, Muller KE (1988). *Applied Regression Analysis and other Multivariate Methods*. 2nd edition. PWS-Kent, Boston.

Li KH, Meng XL, Raghunathan TE, Rubin DB (1991). "Significance Levels from Repeated $p$-values with Multiply-Imputed Data." *Statistica Sinica*, **1**, 65–92.

Little RJA (1988). "Missing Data Adjustments in Large Surveys." *Journal of Business Economics and Statistics*, **6**, 287–301.

Little RJA (2009a). "Comments on: Missing Data Methods in Longitudinal Studies: A Review." *Test*, **18**, 47–50.

Little RJA (2009b). "Selection and Pattern-Mixture Models." In G Fitzmaurice, M Davidian, G Verbeke, G Molenberghs (eds.), *Longitudinal Data Analysis*, chapter 18, pp. 409–431. CRC Press, Boca Raton, Fl.

Little RJA, Rubin DB (2002). *Statistical Analysis with Missing Data*. 2nd edition. John Wiley & Sons, New York.

Lumley T (2010). ***mitools**: Tools for Multiple Imputation of Missing Data*. R package version 2.0.1, URL http://CRAN.R-project.org/package=mitools.

MacLeod J, Hickman M, Bowen E, Alati R, Tilling K, Smith GD (2008). "Parental Drug Use, Early Adversities, Later Childhood Problems and Children's Use of Tobacco and Alcohol at Age 10: Birth Cohort Study." *Addiction*, **103**(10), 1731–1743.

Marshall A, Billingham LJ, Bryan S (2009). "Can We Afford to Ignore Missing Data in Cost-Effectiveness Analyses?" *European Journal of Health Economics*, **10**(1), 1–3.

McCaul KA, Fritschi L, Baade P, Coory M (2008). "The Incidence of Second Primary Invasive Melanoma in Queensland, 1982–2003." *Cancer Causes and Control*, **19**(5), 451–458.

McClelland RL, Kronmal RA, Haessler J, Blumenthal RS, Goff J (2008). "Estimation of Risk Factor Associations when the Response Is Influenced by Medication Use: An Imputation Approach." *Statistics in Medicine*, **27**(24), 5039–5053.

Melhem NM, Brent DA, Ziegler M, Iyengar S, Kolko D, Oquendo M, Birmaher B, Burke A, Zelazny J, Stanley B, Mann, J J (2007). "Familial Pathways to Early-Onset Suicidal Behavior: Familial and Individual Antecedents of Suicidal Behavior." *American Journal of Psychiatry*, **164**(9), 1364–1370.

Meng XL (1995). "Multiple Imputation with Uncongenial Sources of Input." *Statistical Science*, **10**, 538–573.

Meng XL, Rubin DB (1992). "Performing Likelihood Ratio Tests with Multiple-Imputed Data Sets." *Biometrika*, **79**(1), 103–111.

Michel L, Giorgi R, Villes V, Poizot-Martin I, Dellamonica P, Spire B, Protopopescu C, Carrieri MP (2009). "Withdrawal Symptoms as a Predictor of Mortality in Patients HIV-Infected Through Drug Use and Receiving Highly Active Antiretroviral Therapy (Haart)." *Drug and Alcohol Dependence*, **99**(1-3), 96–104.

Moons KGM, Donders R, Stijnen T, Harrell FJ (2006). "Using the Outcome for Imputation of Missing Predictor Values was Preferred." *Journal of Clinical Epidemiology*, **59**(10), 1092–1101.

Morgenstern M, Wiborg G, Isensee B, Hanewinkel R (2009). "School-Based Alcohol Education: Results of a Cluster-Randomized Controlled Trial." *Addiction*, **104**(3), 402–412.

Mueller BA, Cummings P, Rivara FP, Brooks MA, Terasaki RD (2008). "Injuries of the Head, Face, and Neck in Relation to Ski Helmet Use." *Epidemiology*, **19**(2), 270–276.

Mumtaz G, Tamim H, Kanaan M, Khawaja M, Khogali M, Wakim G, Yunis KA (2007). "Effect of Consanguinity on Birth Weight for Gestational Age in a Developing Country." *American Journal of Epidemiology*, **165**(7), 742–752.

Nash D, Katyal M, Brinkhof MWG, Keiser O, May M, Hughes R, Dabis F, Wood R, Sprinz E, Schechter M, Egger M (2008). "Long-Term Immunologic Response to Antiretroviral Therapy in Low-Income Countries: A Collaborative Analysis of Prospective Studies." *Aids*, **22**(17), 2291–2302.

O'Callaghan FV, O'Callaghan M, Najman JM, Williams GM, Bor W, Alati R (2006). "Prediction of Adolescent Smoking from Family and Social Risk Factors at 5 Years, and Maternal Smoking in Pregnancy and at 5 and 14 Years." *Addiction*, **101**(2), 282–290.

Orsini N, Bellocco R, Bottai M, Pagano M, Michaelsson K, Wolk A (2008a). "Combined Effects of Obesity and Physical Activity in Predicting Mortality among Men." *Journal of Internal Medicine*, **264**(5), 442–451.

Orsini N, Mantzoros CS, Wolk A (2008b). "Association of Physical Activity with Cancer Incidence, Mortality, and Survival: A Population-Based Study of Men." *British Journal of Cancer*, **98**(11), 1864–1869.

Oudshoorn K, van Buuren S, van Rijckevorsel JLA (1999). *Flexible Multiple Imputation by Chained Equations of the AVO-95 Survey*, volume PG/VGZ/00.045. TNO Prevention and Health, Leiden. URL http://www.stefvanbuuren.nl/publications/Flexible%20multiple%20-%20TNO99045%201999.pdf.

Pedhazur EJ (1973). *Multiple Regression in Behavioral Research*. 2nd edition. Holt, Rinehart and Winston, New York.

Prompers L, Schaper N, Apelqvist J, Edmonds M, Jude E, Mauricio D, Uccioli L, Urbancic V, Bakker K, Holstein P, Jirkovska A, Piaggesi A, Ragnarson-Tennvall G, Reike H, Spraul M, van Acker K, van Baal J, van Merode F, Ferreira I, Huijberts M (2008). "Prediction of Outcome in Individuals with Diabetic Foot Ulcers: Focus on the Differences Between Individuals with and without Peripheral Arterial Disease. the Eurodiale Study." *Diabetologia*, **51**(5), 747–755.

Raghunathan T, Bondarenko I (2007). "Diagnostics for Multiple Imputations." *SSRN*. URL http://ssrn.com/abstract=1031750.

Raghunathan TE, Lepkowski JM, van Hoewyk J, Solenberger P (2001). "A Multivariate Technique for Multiply Imputing Missing Values Using a Sequence of Regression Models." *Survey Methodology*, **27**, 85–95.

Rahman A, Reed E, Underwood M, Shipley, E M, Omar RZ (2008). "Factors Affecting Self-Efficacy and Pain Intensity in Patients with Chronic Musculoskeletal Pain Seen in a Specialist Rheumatology Pain Clinic." *Rheumatology*, **47**(12), 1803–1808.

Ramchandani PG, Stein A, O'Connor TG, Heron J, Murray L, Evans J (2008). "Depression in Men in the Postnatal Period and Later Child Psychopathology: A Population Cohort Study." *Journal of the American Academy of Child and Adolescent Psychiatry*, **47**(4), 390–398.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Roudsari B, Field C, Caetano R (2008). "Clustered and Missing Data in the Us National Trauma Data Bank: Implications for Analysis." *Injury Prevention*, **14**(2), 96–100.

Roudsari BS, Nathens AB, Cameron P, Civil I, Gruen RL, Koepsell TD, Lecky FE, Lefering RL, Liberman M, Mock CN, Oestern HJ, Schildhauer TA, Waydhas C, Rivara FP (2007). "International Comparison of Prehospital Trauma Care Systems." *Injury*, **38**(9), 993–1000.

Rouxel A, Hejblum G, Bernier MO, Boëlle PY, Ménégaux F, Mansour G, Hoang C, Aurengo A, Leenhardt L (2004). "Prognostic Factors Associated with the Survival of Patients Developing Loco-Regional Recurrences of Differentiated Thyroid Carcinomas." *Journal of Clinical Endocrinology and Metabolism*, **89**(11), 5362–5368.

Royston P (2004). "Multiple Imputation of Missing Values." *The Stata Journal*, **4**, 227–241.

Royston P (2005). "Multiple Imputation of Missing Values: Update." *The Stata Journal*, **5**(2), 188–201.

Royston P, Parmar MKB, Sylvester R (2004). "Construction and Validation of a Prognostic Model Across Several Studies, with an Application in Superficial Bladder Cancer." *Statistics in Medicine*, **23**(6), 907–926.

Royston P, White IR (2011). "Multiple Imputation by Chained Equations (MICE): Implementation in Stata." *Journal of Statistical Software*, **45**(4), 1–20. URL http://www.jstatsoft.org/v45/i04/.

Rubin DB (1976). "Inference and Missing Data." *Biometrika*, **63**, 581–590.

Rubin DB (1987). *Multiple Imputation for Nonresponse in Surveys*. John Wiley & Sons, New York.

Rubin DB (1996). "Multiple Imputation after 18+ Years." *Journal of the American Statistical Association*, **91**(434), 473–489.

Rubin DB (2003). "Nested Multiple Imputation of Nmes via Partially Incompatible MCMC." *Statistica Neerlandica*, **57**(1), 3–18.

Rubin DM, Downes KJ, O'reilly ALR, Mekonnen R, Luan X, Localio R (2008). "Impact of Kinship Care on Behavioral Well-Being for Children in Out-of-Home Care." *Archives of Pediatrics and Adolescent Medicine*, **162**(6), 550–556.

Sabin CA, Lee KJ, Dunn D, Porter K, Bansi RGL, Hill T, Phillips AN, Schwenk A, Leen C, Delpech V, Anderson J, Gazzard B, Johnson M, Easterbrook P, Walsh J, Fisher M, Orkin C (2008). "Treatment Switches after Viral Rebound in HIV-Infected Adults Starting Antiretroviral Therapy: Multicentre Cohort Study: The United Kingdom Collaborative HIV Cohort (CHIC) Study." *Aids*, **22**(15), 1943–1950.

Samant UB, Mack CD, Koepsell T, Rivara FP, Vavilala MS (2008). "Time of Hypotension and Discharge Outcome in Children with Severe Traumatic Brain Injury." *Journal of Neurotrauma*, **25**(5), 495–502.

Sarkar D (2008). ***lattice**: Multivariate Data Visualization with R*. Springer-Verlag, New York.

Schafer JL (1997). *Analysis of Incomplete Multivariate Data*. Chapman & Hall, London.

Schafer JL, Yucel RM (2002). "Computational Strategies for Multivariate Linear Mixed-Effects Models with Missing Values." *Journal of Computational and Graphical Statistics*, **11**(2), 437–457.

Schnoll RA, Rukstalis M, Wileyto EP, Shields AE (2006). "Smoking Cessation Treatment by Primary Care Physicians. an Update and Call for Training." *American Journal of Preventive Medicine*, **31**(3), 233–239.

Schunk D (2008). "A Markov Chain Monte Carlo Algorithm for Multiple Imputation in Large Surveys." *Advances in Statistical Analysis*, **92**(1), 101–114.

Sharma R, Zucknick M, London R, Kacevska M, Liddle C, Clarke SJ (2008). "Systemic Inflammatory Response Predicts Prognosis in Patients with Advanced-Stage Colorectal Cancer." *Clinical Colorectal Cancer*, **7**(5), 331–337.

Shen Z (2000). *Nested Multiple Imputation*. Ph.D. thesis, Harvard University, Department of Statistics.

Sisk JE, Hebert PL, Horowitz CR, McLaughlin MA, Wang JJ, Chassin MR (2006). "Effects of Nurse Management on the Quality of Heart Failure Care in Minority Communities: A Randomized Trial." *Annals of Internal Medicine*, **145**(4), 273–283.

Smith GCS, Crossley JA, Aitken DA, Pell JP, Cameron AD, Connor JM, Dobbie R (2004a). "First-Trimester Placentation and the Risk of Antepartum Stillbirth." *Journal of the American Medical Association*, **292**(18), 2249–2254.

Smith GCS, Wood AM, Pell JP, White IR, Crossley JA, Dobbie R (2004b). "Second-Trimester Maternal Serum Levels of Alpha-Fetoprotein and the Subsequent Risk of Sudden Infant Death Syndrome." *New England Journal of Medicine*, **351**(10), 978–986.

Sommer M, Geurts JWJM, Stessel B, Kessels AGH, Peters ML, Patijn J, van Kleef M, Kremer B, Marcus MAE (2009). "Prevalence and Predictors of Postoperative Pain after Ear, Nose, and Throat Surgery." *Archives of Otolaryngology – Head and Neck Surgery*, **135**(2), 124–130.

Souverein OW, Zwinderman AH, Tanck, T MW (2006). "Multiple Imputation of Missing Genotype Data for Unrelated Individuals." *Annals of Human Genetics*, **70**(3), 372–381.

SPSS Inc (2008a). *SPSS 17 Algorithms*. SPSS Inc., Chicago, IL.

SPSS Inc (2008b). *SPSS 17 Missing Values*. SPSS Inc., Chicago, IL.

Statistical Solutions (2001). ***SOLAS*** *for Missing Data Analysis*. Statistical Solutions, Cork, Ireland. URL http://www.solasmissingdata.com/.

Su YS, Gelman A, Hill J, Yajima M (2011). "Multiple Imputation with Diagnostics (**mi**) in R: Opening Windows into the Black Box." *Journal of Statistical Software*, **45**(2), 1–31. URL http://www.jstatsoft.org/v45/i02/.

Sundell K, Hansson K, Löfholm CA, Olsson T, Gustle LH, Kadesjö C (2008). "The Transportability of Multisystemic Therapy to Sweden: Short-Term Results from a Randomized Trial of Conduct-Disordered Youths." *Journal of Family Psychology*, **22**(4), 550–560.

Tanasoiu C, Colonescu C (2008). "Determinants of Support for European Integration: The Case of Bulgaria." *European Union Politics*, **9**(3), 363–377.

Templ M, Alfons A, Kowarik A (2011). ***VIM****: Visualization of Imputation of Missing Values*. R package version 2.0.3, URL http://CRAN.R-project.org/package=VIM.

Thein HH, Yi Q, Dore GJ, Krahn MD (2008). "Natural History of Hepatitis C Virus Infection in HIV-Infected Individuals and the Impact of HIV in the Era of Highly Active Antiretroviral Therapy: A Meta-Analysis." *Aids*, **22**(15), 1979–1991.

Tiemeier H, van Dijck W, Hofman A, Witteman JCM, Stijnen T, Breteler MMB (2004). "Relationship Between Atherosclerosis and Late-Life Depression: The Rotterdam Study." *Archives of General Psychiatry*, **61**(4), 369–376.

Ton TGN, Longstreth J, Koepsell T (2009). "Active and Passive Smoking and Risk of Narcolepsy in People with Hla Dqb1*0602: A Population-Based Case-Control Study." *Neuroepidemiology*, **32**(2), 114–121.

van Buuren S (2007). "Multiple Imputation of Discrete and Continuous Data by Fully Conditional Specification." *Statistical Methods in Medical Research*, **16**(3), 219–242.

van Buuren S (2010). "Multiple Imputation of Multilevel Data." In JJ Hox, K Roberts (eds.), *The Handbook of Advanced Multilevel Analysis*, chapter 10, pp. 173–196. Routledge, Milton Park, UK.

van Buuren S (2012). *Flexible Imputation of Missing Data*. Chapman & Hall/CRC, Boca Raton, FL.

van Buuren S, Boshuizen HC, Knook DL (1999). "Multiple Imputation of Missing Blood Pressure Covariates in Survival Analysis." *Statistics in Medicine*, **18**(6), 681–694.

van Buuren S, Boshuizen HC, Reijneveld, A S (2006a). "Toward Targeted Hypertension Screening Guidelines." *Medical Decision Making*, **26**(2), 145–153.

van Buuren S, Brand JPL, Groothuis-Oudshoorn CGM, Rubin DB (2006b). "Fully Conditional Specification in Multivariate Imputation." *Journal of Statistical Computation and Simulation*, **76**(12), 1049–1064.

van Buuren S, Groothuis-Oudshoorn K (2011). ***mice**: Multivariate Imputation by Chained Equations*. R package version 2.9, URL http://CRAN.R-project.org/package=mice.

van Buuren S, Oudshoorn K (2000). *Multivariate Imputation by Chained Equations: MICE V1.0 User's Manual*, volume Pg/Vgz/00.038. TNO Prevention and Health, Leiden. URL http://www.stefvanbuuren.nl/publications/mice%20v1.0%20manual%20tno00038%202000.pdf.

van den Hout WB, Goekoop-Ruiterman YPM, Allaart CF, Vries-Bouwstra JKD, Hazes, M JM, Kerstens PJSM, van Zeben D, Hulsmans HMJ, Jonge-Bok JMD, Sonnaville PBJD, Dijkmans BAC, Breedveld FC (2009). "Cost-Utility Analysis of Treatment Strategies in Patients with Recent-Onset Rheumatoid Arthritis." *Arthritis Care and Research*, **61**(3), 291–299.

van der Hulst M, Vollenbroek-Hutten MMR, Groothuis-Oudshoorn KG, Hermens HJ (2008). "Multidisciplinary Rehabilitation Treatment of Patients with Chronic Low Back Pain: A Prognostic Model for Its Outcome." *Clinical Journal of Pain*, **24**(5), 421–430.

van Oijen M, Jong FJD, Witteman JCM, Hofman A, Koudstaal PJ, Breteler, B MM (2007). "Atherosclerosis and Risk for Dementia." *Annals of Neurology*, **61**(5), 403–410.

van Vlierberghe L, Braet C, Goossens L, Rosseel Y, Mels S (2009). "Psychological Disorder, Symptom Severity and Weight Loss in Inpatient Adolescent Obesity Treatment." *International Journal of Pediatric Obesity*, **4**(1), 36–44.

van Wouwe JP, Lanting CI, van Dommelen P, Treffers PE, van Buuren S (2009). "Breastfeeding Duration Related to Practised Contraception in The Netherlands." *Acta Paediatrica, International Journal of Paediatrics*, **98**(1), 86–90.

Veenstra R, Lindenberg S, Winter AFD, Oldehinkel AJ, Verhulst FC, Ormel J (2005). "Bullying and Victimization in Elementary Schools: A Comparison of Bullies, Victims, Bully/Victims, and Uninvolved Preadolescents." *Developmental Psychology*, **41**(4), 672–682.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. 4th edition. Springer-Verlag, New York.

Walker AS, Mulenga V, Sinyinza F, Lishimpi K, Nunn A, Chintu C, Gibb DM (2006). "Determinants of Survival without Antiretroviral Therapy after Infancy in HIV-1-Infected Zambian Children in the Chap Trial." *Journal of Acquired Immune Deficiency Syndromes*, **42**(5), 637–645.

Ward L, Franks P (2007). "Changes in Health Care Expenditure Associated with Gaining or Losing Health Insurance." *Annals of Internal Medicine*, **146**(11), 768–774.

White IR, Daniel R, Royston P (2010). "Avoiding Bias Due to Perfect Prediction in Multiple Imputation of Incomplete Categorical Variables." *Computational Statistics & Data Analysis*, **54**, 2267–2275.

Wiles NJ, Jones GT, Haase AM, Lawlor DA, Macfarlane GJ, Lewis G (2008). "Physical Activity and Emotional Problems Amongst Adolescents." *Social Psychiatry and Psychiatric Epidemiology*, **43**(10), 765–772.

Wolfe F, Caplan L, Michaud K (2006). "Treatment for Rheumatoid Arthritis and the Risk of Hospitalization for Pneumonia: Associations with Prednisone, Disease-Modifying Antirheumatic Drugs, and Anti-Tumor Necrosis Factor Therapy." *Arthritis and Rheumatism*, **54**(2), 628–634.

Wood AM, White IR, Royston P (2008). "How Should Variable Selection Be Performed with Multiply Imputed Data?" *Statistics in Medicine*, **27**(17), 3227–3246.

Yu LM, Burton A, Rivero-Arias O (2007). "Evaluation of Software for Multiple Imputation of Semi-Continuous Data." *Statistical Methods in Medical Research*, **16**, 243–258.

Yucel RM (2008). "Multiple Imputation Inference for Multivariate Multilevel Continuous Data with Ignorable Non-Response." *Philosophical Transactions of the Royal Society A*, **366**, 2389–2403.

## Affiliation:

Stef van Buuren
TNO
P.O. Box 2215
2301 CE Leiden, The Netherlands
*and*
Department of Methodology and Statistics, FSS
University of Utrecht
E-mail: stef.vanbuuren@tno.nl
URL: http://www.stefvanbuuren.nl

Karin Groothuis-Oudshoorn
MB-HTSR, University of Twente
P.O. Box 217
7500 AE Enschede, The Netherlands
E-mail: c.g.m.oudshoorn@utwente.nl
URL: http://www.twentestat.nl

*Journal of Statistical Software* http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 45, Issue 3 *Submitted:* 2009-09-01
December 2011 *Accepted:* 2011-05-30