NEPS
National Educational Panel Study

NEPS Working Papers

Sabine Zinn

An Imputation Model For Multilevel Binary Data

NEPS Working Paper No. 31

Bamberg, November 2013

SPONSORED BY THE

Federal Ministry
of Education
and Research

An Imputation Model For Multilevel Binary Data

Sabine Zinn, University of Bamberg

E-mail address of lead author:
sabine.zinn@uni-bamberg.de

Bibliographic data:

Zinn, S. (2013). *An imputation model for multilevel binary data* (NEPS Working Paper No. 31). Bamberg: University of Bamberg, National Educational Panel Study.

An Imputation Model For Multilevel Binary Data

Abstract

Missing data are a ubiquitous problem of almost all large-scale surveys, and it is also an issue to be addressed in the National Educational Panel Study (NEPS). Analyzing survey data without regarding missing data might cause invalid statistical inference. This is especially true if the process that creates the missing data is not a completely random one, i.e., is non-ignorable. If the probability of an observation being missing depends on observed measurements, the method of multiple imputation provides a remedy for adequately dealing with such situation. Its underlying idea is to replace missing values several times with plausible values. The resulting data sets are then analyzed separately, and the statistical results of the distinct analyses are subsequently combined into an overall result. A technique that has proven its value in this context is the method of multivariate imputation by chained equations. This technique demands a definition of a separate regression model for each incompletely observed variable. On the basis of the regression models thus defined, missing values are replaced by predicted ones. The main requisite for the feasibility of the method of multivariate imputation by chained equations is that the regression models applied be in accordance with the relationships prevalent in the data. The R package **mice** offers a comprehensive collection of relevant imputation models, for example, for continuous data. However, it is currently lacking an imputation model for multilevel binary data. This paper presents an accordant add-on function to enrich the toolbox of **mice**. The validity of this novel imputation function is shown using Monte Carlo simulations.

Keywords

multiple imputation, multivariate imputation by chained equations, imputation model, multilevel binary data

1 Introduction

In the majority of cases, most large scale surveys suffer from problems of missing data. Simply ignoring missing data in statistical analysis and conducting complete-case analyses might however result in invalid inference. This is especially the case if the process that creates the missing data is not a completely random one, i.e., is non-ignorable. If, additionally, the percentage of missing values is higher than about five percent of the data, statistical inferences are highly likely to be biased. The method of multiple imputation has been proven to be effective when dealing with incompletely observed data where the probability of a value being missing depends on the values of the observed study variables. The general idea of this approach is to replace missing values several times with equally plausible values. The resulting data sets are then analyzed separately, and the statistical results of the distinct analyses are subsequently combined into an overall result. For this purpose, Rubin developed a set of rules (Rubin, 1987): The combined parameter estimate is the mean of the obtained estimates and the combined standard error incorporates both between and within imputation variability. The practicability of the method of multiple imputation depends on whether the variable with missing values depends on observed variables or whether it can be explained by external factors. If this is not the case, there is a high risk of multiple imputation producing unfeasible outcomes.[1] One popular way to conduct multiple imputation is by using a sequence of regression models (Raghunathan et al., 2001; van Buuren et al., 2006; van Buuren & Groothuis-Oudshoorn, 2011). The basic idea of this approach is to specify separate imputation models for each variable with missing values and to impute data on a variable-to-variable basis. This imputation technique is also described as multivariate imputation by chained equations. Generating feasible imputed data sets requires regression models that are in accordance with the analysis model in mind, i.e., if the analysis model comprises an interaction effect the imputation model should do as well. Furthermore, relationships prevalent in the data have to be preserved, for example, it is crucial to factor in nonlinear relationships. The regression models used have also to account for the character of the considered variables. Variables can be, for example, binary, nominal, ordinal, or continuous. That is, statistical software aiming to provide reliable imputed data sets in very general contexts should offer a wide range of functionality. To the author's knowledge, the R package **mice** currently offers the most comprehensive collection of relevant instruments (van Buuren & Groothuis-Oudshoorn, 2011). Among other things, it comprises imputation routines for predictive mean matching, Bayesian linear regression, logistic regression, and hot-deck imputation. In addition, the package provides some functions to impute different kinds of multilevel data, for example, it facilitates imputing two-level normal data. Due to its intuitive structure the package can easily be extended by new functionalities. For example, researchers have recently implemented an add-on function to impute multilevel count data (Kleinke & Reinecke, 2013). This paper aims to further enrich the toolbox of **mice** by introducing an add-on function that will make possible to impute multilevel binary data.

Such an imputation routine might be of special relevance in large-scale survey concerned with students in institutions, such as the National Educational Panel Study (NEPS). For example, in the NEPS fall survey of 2010, Grade 9 students were asked, among other things, about their realistic aspirations concerning graduation. The corresponding data could be used to find out whether the migration background of a student would influence his/her aspiration to graduate from secondary school qualifying for university admission (i.e., graduating with *Abitur*). The dependent variable y–graduation with Abitur yes or no–in a corresponding regression model is binary. A multilevel logistic regression analysis is well suited to describe such a relationship

[1] A common misconception is that multiple imputation is restricted to data sets where values are missing completely at random. However, this is not the case (van Buuren & Groothuis-Oudshoorn, 2011): An overview of techniques that facilitate tackling missing patterns that are not completely at random is presented in Little (2008) and Albert & Follmann (2008).

properly. However, studying the variable y shows that around five percent of the values of y is missing. It seems implausible to assume that the process creating the missing values in y is completely random. Thus, simply ignoring missing values in the analysis might likely lead to invalid inference. Fortunately, the NEPS data is very rich and, therefore, it is reasonable to state that the probability of an observation being missing can be explained by other study variables. Counting on this circumstance, missing values of y can be imputed using **mice**. For this purpose the function introduced in this paper can be applied: In combination with **mice** it allows us to impute two-level binary data. The remainder of this paper is structured as follows: In Section 2 we describe the newly developed imputation procedure and detail the corresponding R code. We have run Monte Carlo simulations to test the feasibility of the developed routine. The corresponding setting and the respective outcomes are given in Section 3. The paper concludes with a short discussion of the results, a list of extension points, and some ideas for further research.

2 Imputation Method for Multilevel Binary Data

The general framework that **mice** uses to create multiple imputations is a fully conditional specification approach named multiple imputation by chained equations. This kind of approach requires one imputation model be specified for each incompletely observed variable and thus also the corresponding conditional density. On the basis of the conditional densities, imputations can then be drawn. The R package **mice** already offers a comprehensive collection of imputation models, however, none for multilevel binary data. Subsequently, we describe in detail the general imputation processing conducted within **mice**. Then, we introduce our newly designed add-on function that will allow us to impute multilevel binary data.

2.1 Multiple imputation by chained equations

Let the p variables $Y_1, \ldots, Y_p$ forming the data set Y be incompletely observed. We assume that Y follows a p-variate distribution $P(Y \mid \theta)$ where θ describes a vector of unknown parameters $\theta_1, \ldots, \theta_p$. Hence, $P(Y \mid \theta)$ is completely determined by the parameter vector θ, and replacements for missing values can be achieved by simply drawing samples therefrom. The components of θ are defined such that they are specific to the conditional marginal density $P(Y_j \mid Y_{-j}, \theta_j)$, $j = 1, \ldots, p$. Here Y_{-j} denotes the subset of Y excluding Y_j. This kind of specification facilitates constituting $P(Y \mid \theta)$ in a pretty straightforward way: First, based on the observed data Y^{obs} the posterior distribution $P(\theta \mid Y^{obs})$ of θ is computed. Then, from $P(\theta \mid Y^{obs})$ new parameter values θ^* are sampled. Finally, relying on $P(Y \mid Y^{obs}, \theta^*)$ for missing values replacements Y^* are predicted (by simulation). Both distributions to draw from are approximated by Gibbs sampling. Starting from an initial imputation step made up by a sample from observed marginal distributions, the tth iteration step of the accordant sampler involves successively drawing from

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

where $Y_j^{(t)} = (Y_j^{obs}, Y_j^{*(t)})$ is the imputed variant of Y_j at iteration t. To account for the uncertainty of the prediction, the whole procedure is repeated m times to obtain m imputations.

Commonly, between three and ten imputation are deemed to produce sufficiently accurate results (Schafer, 1999).

2.2 Univariate Imputation Model

Multiple imputation by chained equations demands that a univariate imputation model be specified for each incompletely observed variable Y_j. The essence of the imputation model is to take from Y_{-j} a set of predictors assessed to be relevant for explaining Y_j and to compute a corresponding regression model. The kind of model be chosen depends–among other things–on the analysis model as well as on the scale and the type of the variable Y_j. A multilevel logistic regression approach allows us to model binary data when observation units are nested within higher level categories. Alternatively, this kind of model can be described as a mixed-effects logistic regression model. Details about this model type are given in Hedeker (2003) and Pinheiro & Bates (2000). A typical example of such a setting is a two-level model where students are nested within schools. In formula, a corresponding random intercept and slope model can be written as

$$\log\big(\frac{p_{j,ik}}{1-p_{j,ik}}\big) = (\beta_0 + u_{0k}) + (\beta_1 + u_{1k})X_{1,ik} + \beta_2 X_{2,ik} + \ldots + \beta_q X_{q,ik},$$

where

$p_{j,ik} = P(Y_{j,ik} = 1)$ and $Y_{j,ik}$ is the outcome of the binary variable Y_j measured for person i in cluster k,

β_0 is the overall intercept of the model,

$\beta_1, \ldots, \beta_q$ refers to the fixed effects of the model,

u_{0k} is the random intercept of the model,

u_{1k} is the random slope of the model, and

$X_{1,ik}, \ldots, X_{q,ik}$ are the values of the predictor variables $X_1, \ldots, X_q$ measured for person i in cluster k, $\{X_1, \ldots, X_q\} \subseteq Y_{-j}$.

This kind of specification can easily be extended to also account for interaction effects. The random effects are assumed to follow a normal distribution with mean zero. In accordance therewith, the quantities that have to be estimated are the intercept β_0, the fixed effects $\beta_1, \ldots, \beta_q$, as well as the variances $\tau_0 = var(u_{0k})$ amd $\tau_1 = var(u_{1k})$ of the random effects. That is, in sum $q + 3$ parameters have to be estimated. The actual estimation procedure is conducted on the basis of observed values of Y_j and $X_1, \ldots, X_q$. New values for all missing values of Y_j are predicted relying on the estimated model.

2.3 Imputation Function

In **mice** the actual imputation procedure is carried out by the function *mice*. This function demands an input argument *method* determining the univariate imputation models to be used. Passing the string “norm” will instruct that the built-in function for Bayesian linear regression be used. Let the data set Y be composed by Y_1, Y_2, Y_3, and Y_1 be the only variable containing missing values. Then, the command to conduct Bayesian linear regression to impute values for Y_1 is

```
    mice(Y, method=c("norm","",""))
```

Here, the empty quotation marks indicate that the variables Y_2 and Y_3 need not to be imputed. The set of variables used to predict missing information is determined by the *predictorMatrix* argument of the *mice* function. To this argument a square matrix of the dimension equal to the number of variables in the data set has to be assigned. Each row corresponds to one univariate imputation model and each column to one variable in the data set. The entries of the matrix indicate whether the corresponding variable should be used as predictor in the corresponding imputation model. A 0 indicates that the variable should be neglected, whereas 1 states that the variable should be treated as a fixed–but not as a random–effect. A 2 indicates a cluster variable, that is it characterizes a higher level, and -2 specifies a random effect (which additionally determines a fixed effect). Assume that the variable Y_1 should be imputed using the built-in linear multilevel function of **mice** (denoted by "2L.norm"), while the variables Y_2 and Y_3 should serve as predictors. Let Y_3 constitute a cluster variable and Y_2 give a corresponding random effect. Then, the *predictorMatrix* argument is given by

```
pred <- matrix(c(0,2,-2,
                 0,0,0,
                 0,0,0), ncol=3)
```

The corresponding call of *mice* is

```
mice(Y, predictorMatrix=pred, method=c("2L.norm","",""))
```

The **mice** package has been designed in such a way that users can easily add their own functions for univariate imputation methods. New imputation functions have to be named by *mice.impute.name*, where *name* identifies the univariate imputation model. The newly developed function to impute multilevel binary data is labeled *mice.impute.2l.binom*. It requires **mice** version 2.18 (or newer) and the R package **lme4** (Bates et al., 2012). In its current version, the function fits a two-level logistic regression model with multivariate normal random effects using a restricted maximum likelihood estimation approach. That is, it allows only one class variable.

Figure 1 shows the source code of the function. Its input arguments are *y*, *ry*, *x*, and *type*. These are the standard arguments passed by the *mice* function. The *y* argument contains the incompletely observed variable to impute. It has length n. The argument *ry* comprises an vector that indicates whether a value of *y* is observed or not:

$$ry = \begin{cases} \text{TRUE} & \text{, if } y \text{ is observed,} \\ \text{FALSE} & \text{, otherwise.} \end{cases}$$

The set of complete or completed predictor variables to be used for modeling y is given by the $n \times q$ matrix x. Note that *mice.impute.2l.binom* does not require that extra intercept variables be specified: The model comprises always an intercept for the fixed effects model part and the specification of a class variable automatically entails a random intercept. The *type* argument shows whether predictor variables should be considered as fixed or random effects or whether they should be handled as class variables. It is determined by the *predictorMatrix* argument of the *mice* function. Because *mice.impute.2l.binom* supports only two levels, it will abort with an error message if *type* reports more than one class variable. After having transformed all input arguments appropriately, *mice.impute.2l.binom* fits a two-level logistic regression model using the *lmer* function of the **lme4** package. On the basis of the fitted model, the posterior predictive distribution function $P(Y \mid Y^{obs}, \theta^*)$ of y is determined and new values are drawn for the missing values of y. To update afore the parameter vector θ^*, we employ a two-stage procedure. (A similar processing is also suggested by Gelman & Hill (2007, p. 541).) In a first step, new parameters $\beta^* = (\beta_0^*, \ldots, \beta_q^*)$ are simulated from a $(q+1)$-variate normal distribution with mean $\hat{\beta}$ and a covariance matrix equal to the variance-covariance matrix of $\hat{\beta}$, which is the posterior predictive distribution of β. For this purpose, the product of the transpose of the Cholesky decomposition of the variance-covariance matrix of $\hat{\beta}$ and a $q+1$ dimensional vector of $N(0,1)$ random values is added to $\hat{\beta}$ (Raghunathan et al., 2001). In a second step, in order to draw new random effects from their correct posterior predictive distribution, we compute for each level k of the class variable the variance-covariance matrix of the respective random effects u_{rk}, $r = 0, \ldots, s$. For this purpose, we apply the *ranef* function of the **lme4** package

```
  ranEff <- ranef(fitted.model, condVar=TRUE)
  ranEffCovMat <- attributes(ranEff[[1]])$postVar
```

Here, the input argument `fitted.model` refers to the R object resulting from the *lmer* function. All other input parameters are fixed. On the basis of the variance-covariance matrices of the random effects, for each class level k new random effect values $u_k^* = (u_{0k}^*, \ldots, u_{sk}^*)$ are simulated employing the processing already applied to yield β^*, see also Raghunathan et al. (2001). By means of the newly simulated parameters β^* and u_k^* and the afore-fitted logistic regression model, probabilities are predicted for all incompletely observed values of y. Based on these probabilites, accordant two-level binary data are simulated and passed over to the *mice* function. Hence, each call of *mice.impute.2l.binom* implies passing through one iteration of the Gibbs sampler shown in formula (1). In *mice.impute.2l.binom*, the functionality to predict probabilities for observations in new classes is not yet implemented. That is, if one or more levels of the class variable comprise only missing values, the function aborts with a corresponding error message.

3 Monte Carlo Simulations

To evaluate the newly developed add-on function, we run Monte Carlo simulations based on two types of multilevel models. First, we perform simulations using a random intercept model, then we employ a random intercept and slope model. In both settings, we simulate–on the basis of the model considered–a data set comprising fifty groups, each consisting of one five thousand units. The set of true parameters used for this purpose is denoted by Q. Then, in each simulation step, we sample from this data set one thousand observations. We assume that missing values occur only in the dependent variable y, whereas the predictor variables x are assumed to be fully observed. Missing values are introduced via Bernoulli experiments. The probability p_i indicating whether a value y_i of y is missing is defined as follows:

$$p_i = \text{invlogit}(-1 + x).$$

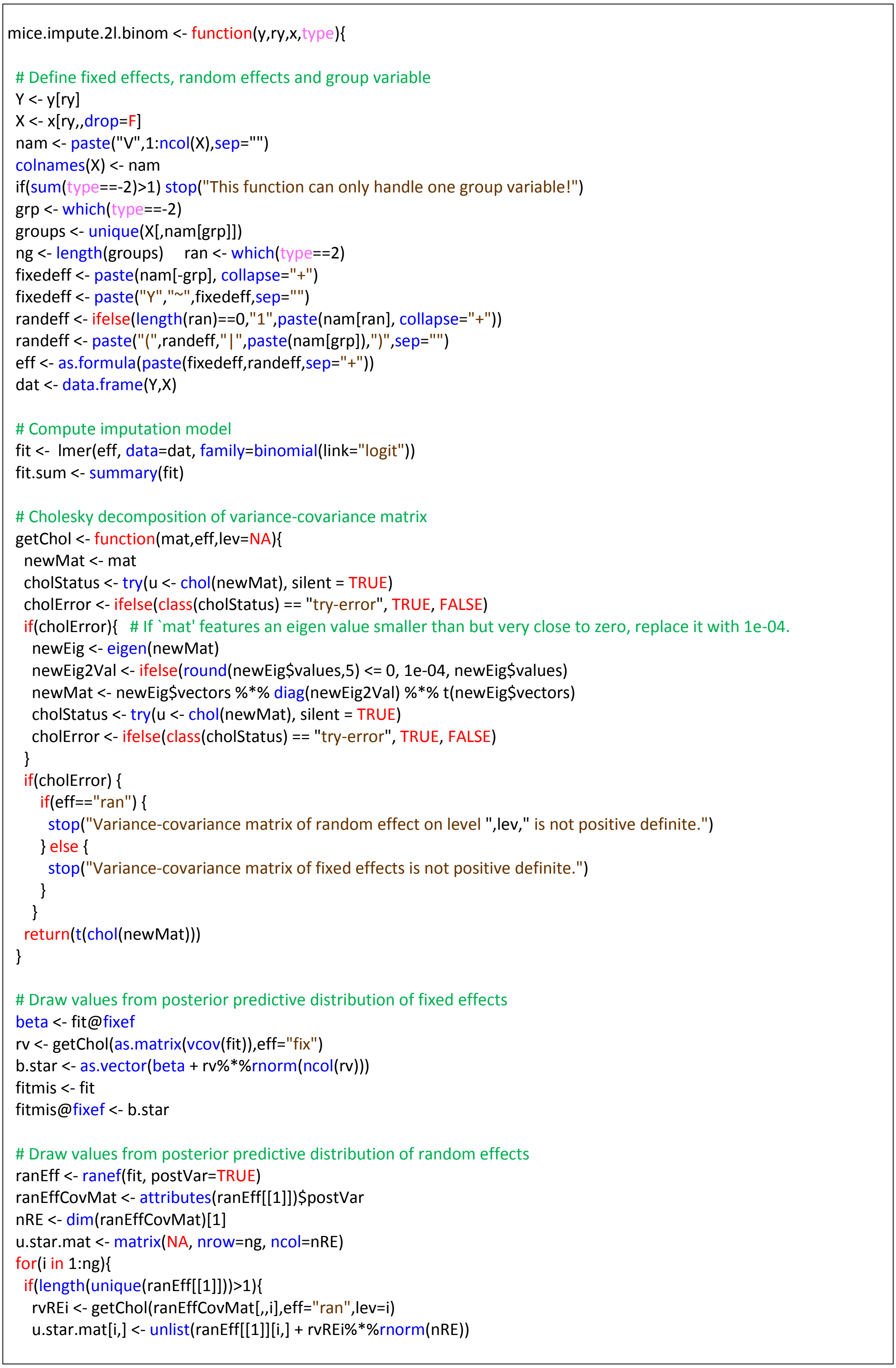
This processing yields an average percentage of 31% of missing values in y. After having constructed data sets this way, the missing values of each simulated data set are imputed five times using the **mice** package in combination with the newly designed add-on function *mice.impute.2l.binom.* Then a logistic regression model allowing for random effects is fitted to each of the imputed data sets, and the results are combined according to Rubin's rules (Rubin, 1987). The whole procedure is repeated 200 times. As Monte Carlo statistics we report the following quantities:

(i) the average combined parameter estimate $\hat{Q}$ across the 200 replications,

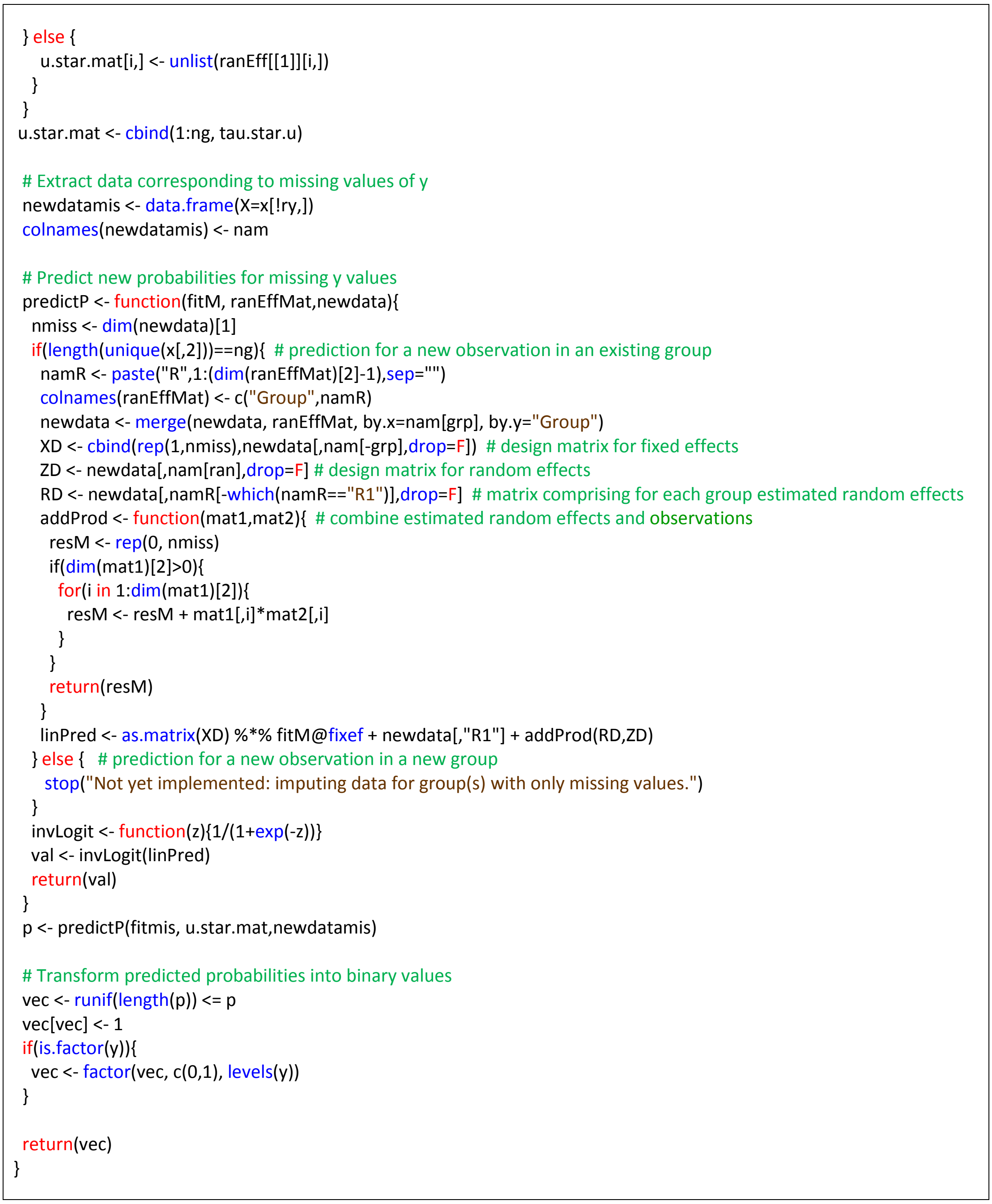
(ii) the standard deviation $SD_{\hat{Q}}$ of the combined parameter estimates across the 200 replications,

(iii) the bias B of the parameter estimation, which is quantified as $B = Q - \hat{Q}$, and

Figure 1: Add-on function for the **R** package *mice* to impute two-level binary data.



```
mice.impute.2l.binom <- function(y,ry,x,type){

 # Define fixed effects, random effects and group variable
 Y <- y[ry]
 X <- x[ry,,drop=F]
 nam <- paste("V",1:ncol(X),sep="")
 colnames(X) <- nam
 if(sum(type==-2)>1) stop("This function can only handle one group variable!")
 grp <- which(type==-2)
 groups <- unique(X[,nam[grp]])
 ng <- length(groups)     ran <- which(type==2)
 fixedeff <- paste(nam[-grp], collapse="+")
 fixedeff <- paste("Y","~",fixedeff,sep="")
 randeff <- ifelse(length(ran)==0,"1",paste(nam[ran], collapse="+"))
 randeff <- paste("(",randeff,"|",paste(nam[grp]),")",sep="")
 eff <- as.formula(paste(fixedeff,randeff,sep="+"))
 dat <- data.frame(Y,X)

 # Compute imputation model
 fit <-  lmer(eff, data=dat, family=binomial(link="logit"))
 fit.sum <- summary(fit)

 # Cholesky decomposition of variance-covariance matrix
 getChol <- function(mat,eff,lev=NA){
  newMat <- mat
  cholStatus <- try(u <- chol(newMat), silent = TRUE)
  cholError <- ifelse(class(cholStatus) == "try-error", TRUE, FALSE)
  if(cholError){   # If `mat' features an eigen value smaller than but very close to zero, replace it with 1e-04.
   newEig <- eigen(newMat)
   newEig2Val <- ifelse(round(newEig$values,5) <= 0, 1e-04, newEig$values)
   newMat <- newEig$vectors %*% diag(newEig2Val) %*% t(newEig$vectors)
   cholStatus <- try(u <- chol(newMat), silent = TRUE)
   cholError <- ifelse(class(cholStatus) == "try-error", TRUE, FALSE)
  }
  if(cholError) {
    if(eff=="ran") {
     stop("Variance-covariance matrix of random effect on level ",lev," is not positive definite.")
    } else {
     stop("Variance-covariance matrix of fixed effects is not positive definite.")
    }
   }
  return(t(chol(newMat)))
 }

 # Draw values from posterior predictive distribution of fixed effects
 beta <- fit@fixef
 rv <- getChol(as.matrix(vcov(fit)),eff="fix")
 b.star <- as.vector(beta + rv%*%rnorm(ncol(rv)))
 fitmis <- fit
 fitmis@fixef <- b.star

 # Draw values from posterior predictive distribution of random effects
 ranEff <- ranef(fit, postVar=TRUE)
 ranEffCovMat <- attributes(ranEff[[1]])$postVar
 nRE <- dim(ranEffCovMat)[1]
 u.star.mat <- matrix(NA, nrow=ng, ncol=nRE)
 for(i in 1:ng){
  if(length(unique(ranEff[[1]]))>1){
   rvREi <- getChol(ranEffCovMat[,,i],eff="ran",lev=i)
   u.star.mat[i,] <- unlist(ranEff[[1]][i,] + rvREi%*%rnorm(nRE))
```



```
  } else {
    u.star.mat[i,] <- unlist(ranEff[[1]][i,])
   }
  }
 u.star.mat <- cbind(1:ng, tau.star.u)

  # Extract data corresponding to missing values of y
  newdatamis <- data.frame(X=x[!ry,])
  colnames(newdatamis) <- nam

  # Predict new probabilities for missing y values
  predictP <- function(fitM, ranEffMat,newdata){
   nmiss <- dim(newdata)[1]
   if(length(unique(x[,2]))==ng){  # prediction for a new observation in an existing group
    namR <- paste("R",1:(dim(ranEffMat)[2]-1),sep="")
    colnames(ranEffMat) <- c("Group",namR)
    newdata <- merge(newdata, ranEffMat, by.x=nam[grp], by.y="Group")
    XD <- cbind(rep(1,nmiss),newdata[,nam[-grp],drop=F])  # design matrix for fixed effects
    ZD <- newdata[,nam[ran],drop=F] # design matrix for random effects
    RD <- newdata[,namR[-which(namR=="R1")],drop=F]  # matrix comprising for each group estimated random effects
    addProd <- function(mat1,mat2){  # combine estimated random effects and observations
     resM <- rep(0, nmiss)
     if(dim(mat1)[2]>0){
      for(i in 1:dim(mat1)[2]){
       resM <- resM + mat1[,i]*mat2[,i]
      }
     }
     return(resM)
    }
    linPred <- as.matrix(XD) %*% fitM@fixef + newdata[,"R1"] + addProd(RD,ZD)
   } else {   # prediction for a new observation in a new group
     stop("Not yet implemented: imputing data for group(s) with only missing values.")
   }
   invLogit <- function(z){1/(1+exp(-z))}
   val <- invLogit(linPred)
   return(val)
  }
  p <- predictP(fitmis, u.star.mat,newdatamis)

  # Transform predicted probabilities into binary values
  vec <- runif(length(p)) <= p
  vec[vec] <- 1
  if(is.factor(y)){
   vec <- factor(vec, c(0,1), levels(y))
  }

  return(vec)
 }
```

(iv) the coverage rate CR which denotes the percentage of the 95% confidence intervals that cover the true parameters.[2]

This simulation frame is borrowed, to a large extent, from Kleinke & Reinecke (2013). All experiments have been run on a desktop workstation equipped with Intel(R) Core(TM) i7, CPU

[2] As the sampling distribution of variance estimates of random effects is in general strongly asymmetric, standard errors are usually a poor characterization of the uncertainties in variance estimates (Bates, 2013, Chapter 1.5). In line therewith, the multilevel function *lmer* of the R package **lme4**, which we use to fit our models, does not report any standard error estimates or confidence intervals for random effects. Hence, coverage rates are only given for fixed effects. Inference on estimated random effects can nevertheless be carried out by using, for example, parametric bootstrapping, MCMC methods, or profile likelihood (Bolker, 2013). However, this is beyond the scopes of this work.

2.80GHz, 8GB RAM, under Windows 7, using a 64bit system.

3.1 Simulation 1: Random Intercept Model

In the first simulation setting we generate data sets using a pretty simple model: We determine the binary variable y to depend on one single individual level predictor x only. Furthermore we assume that the model intercept can be decomposed into the grand mean β_0, which is the same for all individuals, and a group-specific component u_0, which varies between groups. The corresponding data-generating process can be denoted as follows:

$$\begin{aligned} P(y = 1 \mid x) &= \text{invlogit}(\beta_0 + u_0 + \beta_1 x) \\ \beta_0 &= 1 \\ \beta_1 &= 0.75 \\ \sigma &= 0.3 \\ u_0 &\sim N(0, \sigma) \\ x &\sim N(0, 1) \end{aligned}$$

where $\text{invlogit}(z) = 1/(1 + exp(-z))$, and $N(0, \sigma)$ describes the cumulative distribution function of the normal distribution with mean zero and standard deviation σ. Table 1 shows the results of the conducted simulations. We find that the estimated parameters are very close to the true ones, with acceptable standard errors. Furthermore, we find marginal bias for all parameter estimates and reasonable coverage rates. In sum, this allows us to conclude that for the model specification considered the newly designed imputation function works well. The mean execution time to generate one set of multiple imputed data sets was with 8.40 seconds considerably short.

Table 1: *Monte Carlo Statistics of the First Simulation Setting Relying on a Random Intercept Model.*

	β_0	β_1	σ
Q	1.000	0.750	0.300
$\hat{Q}$	1.050	0.751	0.304
$SD_{\hat{Q}}$	0.097	0.095	0.150
B	−0.053	−0.002	−0.003
CR	0.945	0.950	–

3.2 Simulation 2: Random Slope Model

In a further Monte Carlo study, we specify a model that includes–besides a random intercept–also a random slope. We determine that the binary variable y depends on one single individual level predictor x only. The corresponding data-generating process can be denoted as follows:

$$\begin{aligned} P(y = 1 \mid x) &= \text{invlogit}(\beta_0 + u_0 + (\beta_1 + u_1) x) \\ \beta_0 &= 1 \\ \beta_1 &= 0.75 \\ \sigma_0 &= 0.3 \\ \sigma_1 &= 0.2 \\ u_0 &\sim N(0, \sigma_0) \\ u_1 &\sim N(0, \sigma_1) \\ x &= N(0, 1). \end{aligned}$$

Table 2 shows the results of the conducted simulations. In summary, we find that the estimated parameters do not remarkably differ from the true model parameters. The standard errors of the parameter estimates for β_1, β_2, σ_0, and σ_1 are reasonable. Only the standard error of the correlation ρ between the two random effects u_0 and u_1 seems to be out of range. However, this result should not really surprise us given that in every run only a small sample is drawn from the whole population. The coverage rates indicate that the imputation function produces feasible results. In average one imputation round took 24.82 seconds–which we assess as being fairly fast. Overall, we can conclude that also for this model specification our imputation function delivers feasible results.

Table 2: *Monte Carlo Statistics of the Second Simulation Setting Relying on a Random Intercepts and Slopes model.*

	β_0	β_1	σ_0	σ_1	ρ
Q	1.000	0.700	0.300	0.200	0.000
$\hat{Q}$	0.965	0.717	0.270	0.208	-0.016
$SD_{\hat{Q}}$	0.096	0.100	0.025	0.148	0.615
B	0.035	-0.017	0.030	-0.008	0.016
CR	0.895	0.950	–	–	–

Note. ρ denotes the correlation between the two random effects considered.

4 Conclusion

This paper introduces a univariate imputation function for binary multilevel data for the R package **mice**. To yield reasonable imputations, the function fits a mixed-effects logistic regression model. For this purpose, it employs the *lmer* function of the R package **lme4**. By carrying out two different Monte Carlo simulation studies, we have shown that the newly designed function works fine within a random intercepts and slopes model framework.

To prove the usefulness of the introduced imputation model, some further issues have to be addressed. First, the general question whether it is worth to include multilevel components into the imputation procedure must be tackled. Comparing models with and without random effects might shed light on this question. A one-level imputation method that has proven to work well with hierarchical data is the method of predictive mean matching. The method relies on a Bayesian normal model and replaces missing values from the distribution of actually observed values (Heitjan & Little, 1991). Several studies have shown that the method preserves observed hierarchical data structures quite well (Yu et al., 2007). Thus, to further evaluate the necessity of an imputation function for binary multilevel data, both imputation methods should be compared.

To demonstrate its value, in a second step, the newly designed imputation function should be applied to real data. Studying school and student characteristics is ideally suited for such a project. For example, a promising research question in this context is whether or not in Germany the aspiration of students concerning their educational attainment is affected by their migration background. The NEPS provides–among many other things–panel data on students in Grade 9 that will help find answers to this question. However, this remains a task for future work.

References

Albert, P., & Follmann, D. (2008). Shared-parameter models. In G. Fitzmaurice, M. Davidian, G. Verbeke, & G. Molenberghs (Eds.), *Longitudinal data analysis* (p. 433-452). Boca Raton: Chapman & Hall\CRC.

Bates, D. (2013). *lme4: Mixed-effects modeling with R.* Retrieved from `http://lme4.r-forge.r-project.org/book/` (Unpublished)

Bates, D., Maechler, M., & Bolker, B. (2012). lme4: Linear mixed-effects models using s4 classes [Computer software manual]. Retrieved from `http://CRAN.R-project.org/package=lme4` (R package version 0.999999-0)

Bolker, B. (2013, October). *DRAFT r-sig-mixed-models FAQ.* Retrieved from `http://glmm.wikidot.com/faq`

Gelman, A., & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models.* Cambridge: University Press.

Hedeker, D. (2003). A mixed-effects multinomial logistic regression model. *Statistics in Medicine*, *22*(9), 1433-1446.

Heitjan, D. F., & Little, R. J. (1991). Multiple imputation for the fatal accident reporting system. *Applied Statistics*, *40*(1), 13-29.

Kleinke, K., & Reinecke, J. (2013). Multiple imputation of incomplete zero-inflated count data. *Statistica Neerlandica*, *67*(3), 311-336.

Little, R. (2008). Selection and pattern-mixture models. In G. Fitzmaurice, M. Davidian, G. Verbeke, & G. Molenberghs (Eds.), *Longitudinal data analysis* (p. 409-432). Boca Raton: Chapman & Hall\CRC.

Pinheiro, J. C., & Bates, D. M. (2000). *Linear mixed-effects models: basic concepts and examples.* New York: Springer.

Raghunathan, T. E., Lepkowski, J. M., van Hoewyk, J., & Solenberger, P. (2001). A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey Methodology*, *27*(1), 85-96.

Rubin, D. (1987). *Multiple imputation for nonresponse in surveys.* New York: Wiley.

Schafer, J. (1999). Multiple imputation: A primer. *Statistical Methods in Medical Research*, *8*, 3-15. Retrieved from `http://sites.stat.psu.edu/~jls/mifaq.html`

van Buuren, S., Brand, J. P. L., Groothuis-Oudshoorn, C. G. M., & Rubin, D. B. (2006). 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 in R. *Journal of Statistical Software*, *45*(3), 1-67.

Yu, L., Burton, A., & Rivero-Arias, O. (2007). Evaluation of software for multiple imputation of semi-continuous data. *Statistical Methods in Medical Research*, *16*, 243-258.