NEPS
National Educational Panel Study

NEPS Working Papers

Sabine Zinn

An Imputation Model For Multilevel Binary Data

ERRATUM

Erratum to NEPS Working Paper No. 31

Bamberg, August 2014, April 2015, &
January 2016

SPONSORED BY THE
Federal Ministry
of Education
and Research

Background:

This erratum supplements the Working Paper No. 31 "An Imputation Model For Multilevel Binary Data", published in November 2013, by three aspects.

First, in Section 2.3 on page 7 we write that "In mice.impute.2l.binom, the functionality to predict probabilities for observations in new classes is not yet implemented". However, we do not state any reason for this default. We now overcome this negligence: The reason is that it is difficult to come up with a valid posterior predictive distribution for the random effect of a new class. It is rather straightforward to derive average random effects for new classes (compare, e.g., Gelman & Hill (2007, p. 274)), but it is not clear how to specify accordant variance-covariance matrices. Therefore, at the moment, if one or more levels of the class variable comprise only missing values, the function aborts with a corresponding error message.

Second, in Section 3.1 and Section 3.2 we present two simulation studies to validate the imputation routine developed. Simulation 1 deals with a random intercept model and Simulation 2 runs a random intercept random slope model. Overall, we find that our imputation function gives feasible results. Nevertheless, the coverage rate for the intercept β_0 appears to be pretty small. To check whether this is the result of the particular setting or whether this is caused by the imputation routine we conducted further simulation studies. We increased the number of replications from $n = 200$ to $n = 500$. Furthermore, we tested different parameter sets. All in all, we conclude that the small coverage rate which we found in the initial example results from the (relatively) small number of replications. To underline this statement, we report the results of two further simulation studies (with $n = 500$ replications). The Tables 1 and 2 show the results of the conducted simulation studies. In conclusion, we find reasonable coverage rates for both fixed effects.

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

	β_0	β_1	σ
Q	1.000	0.750	0.300
$\hat{Q}$	1.057	0.757	0.295
$SD_{\hat{Q}}$	0.096	0.099	0.150
B	−0.057	−0.007	0.005
CR	0.954	0.952	–

Table 2: *Monte Carlo Statistics of Simulation Setting 2 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
$\hat{Q}$	1.033	0.730	0.238	0.190	0.038
$SD_{\hat{Q}}$	0.096	0.103	0.152	0.118	0.590
B	−0.033	−0.030	0.062	0.010	−0.038
CR	0.960	0.960	–	–	–

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

Third, the source code of the newly developed imputation routine *mice.impute.2l.binom* has been adapted slightly and a typing error in line 80 has been removed. Subsequently, the reworked source code is given.

```
1  mice.impute.2l.binom <- function(y, ry, x, type){
2
3    # Define fixed effects, random effects and group variable
4    Y <- y[ry]
5    X <- x[ry, , drop = F]
6    nam <- paste("V", 1 : ncol(X), sep = "")
7    colnames(X) <- nam
8    if(sum(type == -2) > 1)
9        stop("This function can only handle one group variable!")
10   grp <- which(type == -2)
11   groups <- unique(X[, nam[grp]])
12   ng <- length(groups)
13   ran <- which(type == 2)
14   fixedeff <- paste(nam[- grp], collapse = "+")
15   fixedeff <- paste("Y", "~", fixedeff, sep = "")
16   randeff <- ifelse(length(ran) == 0, "1", paste(nam[ran], collapse = "+"))
17   randeff <- paste("(", randeff, "|", paste(nam[grp]), ")", sep = "")
18   eff <- as.formula(paste(fixedeff, randeff, sep = "+"))
19   dat <- data.frame(Y, X)
20   noInf <- names(which(t(table(dat$Y, X[, grp])) == 0, arr.ind = T)[, 1])
21   if(length(noInf) > 0){
22     warning("There are groups without variation.")
23   }
24
25   # Compute imputation model
26   fit <- suppressWarnings(glmer(eff, data = dat, family = binomial(link = "logit")))
27     # If there are groups without variation (i.e., all Y are either zero or one),
28     # glmer gives a warning. Gelman & Hill (2007, p.276) argue that even in a such
29     # case for the affected groups  partial information is available allowing for
30     # estimating regression parameters.
31   fit.sum <- summary(fit)
32
33   # Function for cholesky decomposition of variance-covariance matrix
34   getChol <- function(mat, eff, lev = NA){
35     newMat <- mat
36     cholStatus <- try(u <- chol(newMat), silent = TRUE)
37     cholError <- ifelse(class(cholStatus) == "try-error", TRUE, FALSE)
38     if(cholError){
39       newEig <- eigen(newMat)
40       # If 'mat' features an eigen value smaller than but very close to zero,
41       # replace it with 1e-04.
42       newEig2Val <- ifelse(round(newEig$values,5) <= 0, 1e-04, newEig$values)
43       newMat <- newEig$vectors %*% diag(newEig2Val) %*% t(newEig$vectors)
44       cholStatus <- try(u <- chol(newMat), silent = TRUE)
45       cholError <- ifelse(class(cholStatus) == "try-error", TRUE, FALSE)
46     }
47     if(cholError) {
48       if(eff == "ran") {
49          stop("Variance-covariance matrix of random effect on level ", lev,
50           " is not positive definite.")
51       } else {
52          stop("Variance-covariance matrix of fixed effects is not positive
```

```
53            definite.")
54        }
55      }
56      return(t(chol(newMat)))
57    }
58
59    # Draw values from posterior predictive distribution of fixed effects
60    beta <- fit@beta
61    rv <- getChol(as.matrix(vcov(fit)), eff = "fix")
62    b.star <- as.vector(beta + rv%*%rnorm(ncol(rv)))
63    fitmis <- fit
64    fitmis@beta <- b.star
65
66    # Draw values from posterior predictive distribution of random effects
67    ranEff <- ranef(fit, condVar = TRUE)
68    ranEffCovMat <- attributes(ranEff[[1]])$postVar
69    nRE <- dim(ranEffCovMat)[1]
70    ngrV <- dim(ranEffCovMat)[3]
71    u.star.mat <- matrix(NA, nrow = ngrV, ncol = nRE)
72    for(i in 1 : ngrV){
73      if(length(unique(ranEff[[1]])) > 1){
74          rvREi <- getChol(ranEffCovMat[, , i], eff = "ran", lev = i)
75          u.star.mat[i, ] <- unlist(ranEff[[1]][i, ] + rvREi %*% rnorm(nRE))
76        } else {
77          u.star.mat[i, ] <- unlist(ranEff[[1]][i, ])
78        }
79      }
80    u.star.mat <- cbind(1 : ngrV, u.star.mat)
81
82    # Data corresponding to missing values of y
83    newdatamis <- data.frame(X = x[!ry, ])
84    colnames(newdatamis) <- nam
85
86    # Predict new values for missing values of y
87    predictP <- function(fitM, ranEffMat, newdata){
88      nmiss <- dim(newdata)[1]
89      # prediction for a new observation in an existing group
90      if(length(unique(x[, 2])) == ng){
91        namR <- paste("R",1:(dim(ranEffMat)[2]-1),sep="")
92        colnames(ranEffMat) <- c("Group", namR)
93        newdata <- merge(newdata, ranEffMat, by.x = nam[grp], by.y = "Group")
94        # design matrix for fixed effects
94        XD <- cbind(rep(1, nmiss),newdata[, nam[-grp], drop = FALSE])
95        # design matrix for random effects
96        ZD <- newdata[, nam[ran], drop = FALSE]
97        # matrix comprising for each group estimated random effects
98        RD <- newdata[, namR[- which(namR == "R1")], drop = FALSE]
99        # combine estimated random effects and observations
100       addProd <- function(mat1, mat2){
101         resM <- rep(0, nmiss)
102         if(dim(mat1)[2] > 0){
103           for(i in 1 : dim(mat1)[2]){
```

```
104             resM <- resM + mat1[, i] * mat2[, i]
105           }
106         }
107         return(resM)
108       }
109       linPred <- as.matrix(XD) %*% fitM@beta + newdata[, "R1"] + addProd(RD, ZD)
110     } else {
111       # prediction for a new observation in a new group
112       stop("Not yet implemented: imputing data for group(s) with only
113         missing values.")
114     }
115      invLogit <- function(z){1 / (1 + exp(- z))}
116      val <- invLogit(linPred)
117      return(val)
118   }
119   p <- predictP(fitmis, u.star.mat, newdatamis)
120   vec <- runif(length(p)) <= p
121   vec[vec] <- 1
122   if(is.factor(y)){
123     vec <- factor(vec, c(0,1), levels(y))
124   }
125   return(vec)
126 }
```

Reference

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