jomo: A Flexible Package for Two-level Joint Modelling Multiple Imputation

by Matteo Quartagno, Simon Grund and James Carpenter

Abstract Multiple imputation is a tool for parameter estimation and inference with partially observed data, which is used increasingly widely in medical and social research. When the data to be imputed are correlated or have a multilevel structure — repeated observations on patients, school children nested in classes within schools within educational districts — the imputation model needs to include this structure. Here we introduce our **jo**int **mo**delling package for multiple imputation of multilevel data, **jomo**, which uses a multivariate normal model fitted by Markov Chain Monte Carlo (MCMC). Compared to previous packages for multilevel imputation, e.g. **pan**, **jomo** adds the facility to (i) handle and impute categorical variables using a latent normal structure, (ii) impute level-2 variables, and (iii) allow for cluster-specific covariance matrices, including the option to give them an inverse-Wishart distribution at level 2. The package uses C routines to speed up the computations and has been extensively validated in simulation studies both by ourselves and others.

Introduction

Missing data are ubiquitous in clinical and social research. The most straightforward way to deal with missing data is to exclude all observations with any item missing from the analysis, i.e. a *complete records analysis*. However this strategy is at best inefficient; further unless — given covariates — the probability of a complete record does not depend on the outcome (dependent) variable, it will lead to biased results.

Rubin (1976) described different mechanisms causing missing data: Missing Completely At Random (probability of missingness unrelated to observed and unobserved values, MCAR), Missing At Random (given observed data, occurrence of missing values is independent of the actual values, MAR) and Missing Not At Random (given observed data, occurrence of missing values still depends on the actual values, MNAR).

Multiple Imputation (MI) is a very flexible, practical, tool to deal with missing data. It consists of imputing missing data several times, creating multiple imputed data sets. Then, the substantive model is directly fitted to each of the imputed data sets; the results are then combined for inference using Rubin's rules (Rubin, 1987). An appropriately specified multiple imputation model gives valid, efficient inference if data are MAR (Carpenter and Kenward, 2013, Ch. 2; **?**). Key attractions of MI are that it separates the imputation of missing data from the analysis, thus allowing (a) use of the substantive analysis model that we intended to use with fully observed data and (b) inclusion of auxiliary variables in the imputation model — which provide information about the missing values — without having to include them in the substantive analysis model.

There are several models and associated algorithms which can be used to impute missing data; Schafer (1997) proposed a joint multivariate normal model, fitted by MCMC. The main assumption of this method is that the partially observed data follow a joint multivariate normal distribution; given this, a Gibbs sampler uses the proper conditional distributions to update the parameters of the model and impute the missing data.

One of the advantages of the joint modelling approach is that it extends naturally to multilevel/hierarchical data structures. Such structures arise commonly, for example, when we have repeated observations (level 1) on individuals (level 2), or students (level 1) nested in schools (level 2).

A number of joint modelling multiple imputation packages have been written: **norm** (Novo and Schafer, 2013; Schafer and Olsen, 2000) assumes a multivariate normal model for imputation of single-level normal data, **cat** (Harding et al., 2012) a log-linear model to impute categorical data, and **mix** (Schafer., 2010) uses the general location model (Olkin and Tate, 1961) to impute a mix of continuous and categorical data (Schafer, 1997, Ch. 9). **pan** (Zhao and Schafer, 2013; Schafer and Yucel, 2002) uses a multilevel multivariate normal model for imputation of multilevel normal data.

As far as we are aware, **jomo** is the first R package to extend this to allow for a mix of multilevel (clustered) continuous and categorical data. It is derived from, but also extends the functionality of REALCOM (Carpenter et al., 2011), a standalone program written in Matlab. In REALCOM, binary and categorical variables are handled through a latent normal variables approach presented in Carpenter and Kenward (2013, Ch. 5). The aim of the **jomo** package is to provide an efficient implementation of the REALCOM imputation model in R, while both (i) speeding up the processes through the use of C sub-routines and (ii) adding the flexibility to specify level-2 specific covariance matrices and level-2 random covariance matrices, the latter following the proposal of Yucel (2011).

This paper is organized as follows: after briefly introducing joint modelling multiple imputation and the principal functions within **jomo**, we present a series of short tutorials, in which we explain how to impute (i) single-level continuous and categorical data; (ii) clustered homoscedastic data, (iii) clustered heteroscedastic data and (iv) level-2 variables. We then present functions to help check the convergence of the underlying MCMC algorithms and outline the suggested workflow for using the package. The next section introduces another R package, **mitml**, which provides both a different interface to **jomo** and a number of useful tools to manage and investigate the properties of imputed data sets. The penultimate section provides an overview of both the simulation studies and the applications where the package was used. We conclude with a short discussion.

Joint Modelling Multiple Imputation

To introduce the general ideas of joint modelling multiple imputation, we consider a data set made up of N observations on three continuous variables; we further assume that one of these variables, X, is fully observed, while the other two, Y_1 and Y_2, have missing values. The main idea behind joint modelling imputation is to define a joint multivariate model for all the variables in the data set, to be used for imputation. The simplest joint model is the multivariate normal model:

$$\begin{aligned} Y_{1,i} &= \beta_{0,1} + \beta_{1,1} X_i + \epsilon_{1,i} \\ Y_{2,i} &= \beta_{0,2} + \beta_{1,2} X_i + \epsilon_{2,i} \\ \begin{pmatrix} \epsilon_{1,i} \\ \epsilon_{2,i} \end{pmatrix} &\sim \mathcal{N}(\mathbf{0}, \mathbf{\Omega}) \end{aligned} \tag{1}$$

This model uses X, which is completely observed, as predictor while Y_1 and Y_2 are included as outcomes, being partially observed. Another possibility is a tri-variate normal model with all variables, including X, treated as outcomes. Choosing between these two models has been the matter of debate in the literature, and is not the focus of this paper. We simply note here that the imputation model chosen needs to be (at least approximately) congenial with the analysis model of substantial interest, in order for multiple imputation to lead to correct inference (Meng, 1994).

After having chosen the imputation model, Bayesian methods are used to fit it and impute the missing data. In particular, Gibbs sampling is used, dealing with missing data via a data augmentation algorithm (Tanner and Wong, 1987). This consists of repeatedly drawing new values for all the parameters in the model, one at a time, from the relevant conditional distribution. The parameters of model (1) are the fixed effects $\boldsymbol{\beta}$ and the covariance matrix $\mathbf{\Omega}$, and the missing data.

The sampler has to be run until it reaches the stationary distribution. Then, the current draw of the missing values is combined with the observed data to make the first imputed data set. Further imputed data sets are obtained by running the sampler for a sufficient number of supplementary iterations to guarantee stochastic independence between consecutive imputations.

Before running the Gibbs sampler, it is necessary to choose starting values for the parameters; the more plausible these starting values are, the faster the sampler converges. Furthermore, being a Bayesian method, prior distributions are used. By default, **jomo** uses flat priors for all the parameters in the model, except for the covariance matrices, which are given inverse-Wishart priors with the minimum possible number of degrees of freedom in order to give the greatest possible weight to the observed data.

Extending the same methodology to more complicated situations is conceptually relatively straightforward. For example, binary and categorical variables can be included in the imputation model by means of latent normal variables. Under this model, if Y_1 is binary, a latent continuous variable Y_1^* is included in the model, such that $Y_{1,i}^* > 0$ for individuals i for whom $Y_{1,i} = 0$ and $Y_{1,i}^* \leq 0$ for individuals for whom $Y_{1,i} = 1$ (Goldstein et al., 2009). Similarly, a categorical variable with K categories can be represented by $K - 1$ latent normal variables denoting the differences between categories. In contrast to models without categorical data, this strategy requires constraints on the variance-covariance matrix to guarantee identifiability of the model (for a wider discussion of model identifiability see Carpenter and Kenward (2013, Chap.5)). To sample from this constrained covariance matrix, Metropolis Hastings (MH) steps are introduced to augment the standard Gibbs Sampler.

If observations in the data set are nested in J clusters, model (1), can be readily expanded to include

		Type of variables		
Model type	Missing data	Continuous	Binary/categorical	Mixed
Single-level	Level 1	`jomo1con`	`jomo1cat`	`jomo1mix`
Two-level	Level 1	`jomo1rancon`	`jomo1rancat`	`jomo1ranmix`
	Both levels	`jomo2com`	`jomo2com`	`jomo2com`
Two-level, heteroscedastic	Level 1	`jomo1ranconhr`	`jomo1rancathr`	`jomo1ranmixhr`
	Both levels	`jomo2hr`	`jomo2hr`	`jomo2hr`

Table 1: Summary of subfunctions used by the main "umbrella function" `jomo`, given the type of imputation model, the level at which missing data occur, and the type of the variables with missing data.

random intercepts as follows:

$$\begin{aligned} Y_{1,i,j} &= \beta_{0,1} + u_{1,j} + \beta_{1,1} X_{i,j} + \epsilon_{1,i,j} \\ Y_{2,i,j} &= \beta_{0,2} + u_{2,j} + \beta_{1,2} X_{i,j} + \epsilon_{2,i,j} \\ \begin{pmatrix} \epsilon_{1,i,j} \\ \epsilon_{2,i,j} \end{pmatrix} &\sim \mathcal{N}(\mathbf{0}, \mathbf{\Omega}) \\ \begin{pmatrix} u_{1,j} \\ u_{2,j} \end{pmatrix} &\sim \mathcal{N}(\mathbf{0}, \mathbf{\Omega}_u) \end{aligned} \tag{2}$$

More general random effects structures can be modelled as well, as we will show in a later section.

It is similarly possible to consider a joint imputation model for variables at level 2, e.g. patient-level variables in longitudinal studies (Carpenter and Kenward, 2013, p. 212), or allowing for level-1 heteroscedasticity, which can be particularly appealing in individual patient data (IPD) meta-analyses among other applications.

Package structure

The **jomo** package can be used to impute missing data in continuous and categorical variables in single-level and multilevel data. The main interface to the **jomo** package is the `jomo` function, which automatically selects the correct imputation method for the data, depending on (a) the model specification (e.g., single-level vs. multilevel) and (b) the variables in the data set (e.g., continuous variables of type `numeric` vs. categorical variables as `factor` vs. a mixture of them). In addition, the sub-functions called by `jomo` can be called individually; a list of all sub-functions is given below and in Table 1.

1. `jomo1con`, `jomo1cat` and `jomo1mix`: these impute single-level continuous, categorical and mixed data sets. `jomo1con` is very similar to the `imp.norm` function of the **norm** package. However, `jomo1cat` and `jomo1mix` used the latent normal variables approach described in Carpenter and Kenward (2013, Ch. 5) to impute categorical variables and a mix of continuous and categorical variables, respectively.
2. `jomo1rancon`, `jomo1rancat` and `jomo1ranmix`: these impute clustered continuous, categorical, and mixed data, respectively. `jomo1rancon` is very similar to **pan**, whereas `jomo1rancat` and `jomo1ranmix` use the latent normal model for the categorical variables. All these functions have a fixed, common covariance matrix across all the clusters (level-2 units) in the imputation model;
3. `jomo1ranconhr`, `jomo1rancathr` and `jomo1ranmixhr`: these functions extend the above to allow for either cluster (level-2 unit) specific covariance matrices, or random covariance matrices, where the covariance matrices follow an inverse-Wishart distribution across level-2 units, as described by Yucel (2011) and Quartagno and Carpenter (2016).
4. `jomo2com` and `jomo2hr`: these functions impute missing values in level-2 variables, and can be used in the same manner as those in groups (2) and (3) above.

We next illustrate the use of `jomo` in each of the above situations. Throughout, we assume that the data are MAR.

jomo: tutorial with single-level data

We begin with single-level data sets. We first assume all variables with missing data are continuous, that the substantive model is a linear regression of one variable on some or all the others, and that each variable is approximately normally distributed conditional on the others (so that a joint multivariate normal model is an appropriate choice for imputation).

A key attraction of multiple imputation is that any variables in the data set that are not in the substantive model can still be included in the imputation model. If such *auxiliary variables* are good predictors of missing values, they will help recover missing information. If they are also predictors of the data being missing, they may correct bias. Recall that (simply speaking) the MAR assumption states that, given the observed data, the probability that a value is missing is conditionally independent of the actual value. Therefore, inclusion of judiciously chosen auxiliary variables (even ones that are not themselves complete) can improve the plausibility of the MAR assumption.

In joint modelling imputation (Carpenter and Kenward, 2013, Ch. 3), partially observed variables are dependent variables. However, as hinted above, with fully observed variables we can choose to either condition on them as predictors or include them in the (multivariate) response. The software is equally comfortable with both options, and it makes little difference in practice for single-level data. However, the choice has a bigger impact for clustered data (Quartagno and Carpenter, 2016; Grund et al., 2016b), as we will see in the multilevel imputation section.

Once we have decided on the variables to include in the imputation model, and whether to condition on any fully observed variables as covariates, imputation using the `jomo` function is straightforward.

For illustration, we use an educational data set of students' test scores (`JSPmiss`), which is a subset of the Junior School Project (Mortimore et al., 1988). The fully observed data set is freely available with the MLwiN software (Rasbash et al. (2017); or the related R package **R2MLwiN** Zhang et al. (2016)) and a partially observed version is included with this package (data are MAR). This data set has eight variables: a `school` and an individual identifier, `sex`, fluency in English language (3 categories, `fluent`), a test score at Year 1 (`ravens`) and at Year 3 (`english`) and a binary behavioral score at Year 3 (`behaviour`). Additionally a constant is provided to help the user, as we will see later in this tutorial.

First, we summarize the data:

```
library(jomo)
```

```
> summary(JSPmiss)
     school          id            sex          fluent        ravens         english
 48     : 76   280    :   1   Min.   :0.0000   0   : 32   Min.   : 4.00   Min.   : 0.00
 42     : 52   281    :   1   1st Qu.:0.0000   1   : 29   1st Qu.:22.00   1st Qu.:24.00
 31     : 44   282    :   1   Median :1.0000   2   :823   Median :26.00   Median :40.00
 8      : 43   283    :   1   Mean   :0.5103   NA's:235   Mean   :25.35   Mean   :41.36
 33     : 43   284    :   1   3rd Qu.:1.0000              3rd Qu.:30.00   3rd Qu.:56.00
 5      : 39   285    :   1   Max.   :1.0000              Max.   :36.00   Max.   :98.00
 (Other):822   (Other):1113                               NA's   :246     NA's   :236
       behaviour        cons
 lowerquarter:248   Min.   :1
 upper       :871   1st Qu.:1
                    Median :1
                    Mean   :1
                    3rd Qu.:1
                    Max.   :1
```

This shows the data set has $(248 + 871) = 1119$ observations, of which 236 have missing data on the outcome `english`, 235 have missing data on `fluent` and so on and so forth. For the purpose of this example, we ignore clustering by school and take as the substantive analysis model the following linear regression:

$$\begin{aligned} Y_{english,i} =& \alpha_0 + \alpha_1 \text{ravens}_i + \alpha_2 \text{sex}_i + \epsilon_i \\ & \epsilon_i \sim N(0, \sigma_e^2). \end{aligned} \tag{3}$$

To illustrate the use of `jomo` with continuous variables, let $i = 1, \ldots, 1119$ index observations and use

the following joint imputation model:

$$\begin{aligned} Y_{english,i} &= \beta_{0,e} + \beta_{1,e} X_{sex,i} + \epsilon_{e,i} \\ Y_{ravens,i} &= \beta_{0,r} + \beta_{1,r} X_{sex,i} + \epsilon_{r,i} \\ \begin{pmatrix} \epsilon_{e,i} \\ \epsilon_{r,i} \end{pmatrix} &\sim \mathcal{N}(\mathbf{0}, \mathbf{\Omega}) \end{aligned} \tag{4}$$

Both responses in this bivariate normal model, `english` and `ravens`, are partially observed continuous variables. The binary covariate, `sex`, is fully observed. This is identical to model (1) with X being `sex`.

To impute the missing values we proceed as follows. First, in order to guarantee that the correct function is being used, we need to make sure that continuous variables are stored as numeric vectors in the data frame and binary/categorical variables as `factors`. We can easily test this as follows:

```
# Check that ravens is numeric:
class(JSPmiss$ravens)
# [1] "numeric"

# Were ravens not numeric, we convert it as follows:
JSPmiss <- within(JSPmiss, ravens <- as.numeric(ravens))

# Define the data.frame with the outcomes
Y <- JSPmiss[, c("english", "ravens")]

# The imputation model requires an intercept variable (simply a column of 1's)
# to include in the covariate matrix X. The JSPmiss data set already contains
# such a variable (cons). Were it not present, we could define it as follows:
JSPmiss$cons <- 1

# Define the data.frame with the covariates:
X <- JSPmiss[, c("cons", "sex")]

# Set the seed (so we can replicate the results exactly if desired)
set.seed(1569)

# Run jomo and store the imputed data sets in a new data frame, imp
imp <- jomo(Y = Y, X = X, nburn = 1000, nbetween = 1000, nimp = 5)
```

Running this code, with the above seed, the following output is shown on screen:

```
# No clustering, using functions for single-level imputation.
# Found  2 continuous outcomes and no categorical. Using function jomo1con.
# ..........................................................................
# ..........................First imputation registered.
# ..........................................................................
# ..........................Imputation number  2 registered
# ..........................................................................
# ..........................Imputation number  3 registered
# ..........................................................................
# ..........................Imputation number  4 registered
# ..........................................................................
# ..........................Imputation number  5 registered
# The posterior mean of the fixed effects estimates is:
#             cons       sex
# english 38.15030  6.408079
# ravens  25.32372 -0.556898
#
# The posterior covariance matrix is:
#           english   ravens
# english 458.39885 64.92863
# ravens   64.92863 36.41841
```

The first sentence informs us that, as we did not pass any clustering indicator to the function, `jomo` is

using single-level imputation. The second sentence is telling us which of the sub-functions was used. As we have two numeric dependent variables, `jomo1con` has been chosen.

Then, the software prints a '.' for each 10 burn-in updates of the MCMC sampler, followed by a notification that it has created the first imputed data set. It then prints a '.' for each 10 further updates of the MCMC sampler, before imputations 2, 3, 4, and 5. The default values for the burn-in and between-imputation updates are both 1000, resulting in 100 dots printed in each case.

Finally, the software prints the estimated posterior mean of the regression coefficients for `english` ($\beta_{0,e}, \beta_{1,e}$), `ravens` ($\beta_{0,r}, \beta_{1,r}$), and the elements of the covariance matrix:

$$\begin{pmatrix} \epsilon_{e,i} \\ \epsilon_{r,i} \end{pmatrix} \sim \mathcal{N}\left[\mathbf{0}, \mathbf{\Omega} = \begin{pmatrix} 458.40 & 64.93 \\ 64.93 & 36.42 \end{pmatrix}\right].$$

The same results can be obtained by running `jomo1con` in place of `jomo` with exactly the same arguments. In general, to exactly replicate the results, you will have to (i) have the data sorted in the same order and (ii) use the same seed. Sometimes we will wish to suppress the output generated by `jomo`; the option `output = 0` controls this.

Running `jomo` creates the object `imp`, which is a data frame:

```
class(imp)
#[1] "data.frame"

# with the following names:
names(imp)
#[1] "english"    "ravens"        "cons"       "sex"        "id"         "Imputation"

# and dimension:
dim(imp)
# [1] 6714    6
```

The `names` include the original variable names, inherited from `JSPmiss`, and a new variable `Imputation`, which indexes the original data (`0`) and the imputed data sets. We have five imputed data sets. Let's look at the top of the original data, and the top of the fifth imputation:

```
# View original (partially observed) data:
head(imp)

#   english ravens cons sex id Imputation
# 1      39     NA    1   1  1          0
# 2      NA     15    1   0  2          0
# 3      65     19    1   1  3          0
# 4      NA     22    1   0  4          0
# 5      30     NA    1   1  5          0
# 6      12     NA    1   0  6          0

# View last imputation (the left most column is the row number):
head(imp[imp$Imputation == 5,])

#      english   ravens cons sex id Imputation
# 5596 39.00000 32.17927    1   1  1          5
# 5597 54.37720 15.00000    1   0  2          5
# 5598 65.00000 19.00000    1   1  3          5
# 5599 46.57598 22.00000    1   0  4          5
# 5600 30.00000 21.30211    1   1  5          5
# 5601 12.00000 24.56964    1   0  6          5
```

Starting values and prior distributions

To impute missing data, `jomo` fits a Bayesian model using MCMC. Therefore, we need to provide starting values and priors for each parameter in the model. The default starting values are a matrix of zeros for the fixed effects parameter $\boldsymbol{\beta}$, and the identity matrix for the covariance matrix. In the majority of situations, changing the starting values will not change the results materially. Nevertheless, good starting values may considerably reduce the number of iterations needed for the algorithm to converge.

In order to represent the maximum uncertainty and to give the greatest weight to the data, `jomo`

assumes flat improper priors for all the parameters in the model, except the covariance matrix. For this, an inverse-Wishart prior is used, with degrees of freedom set to the minimum possible, i.e. the dimension of the covariance matrix; this represents the greatest uncertainty (least information). Changing the scale matrix of the inverse-Wishart prior may have some impact when we have a very small number of observations. In our example, with 1119 observations and only two outcomes, the impact is immaterial.

We now illustrate how to set the starting values for all the parameters, the scale matrix of the inverse-Wishart prior for the covariance matrix as well as the number of burn-in iterations for the MCMC sampler, the number of iterations between imputations, and the number of imputations:

```
# Set starting values for fixed effect parameters beta
beta.start <- matrix(1, 2, 2)

# Set starting value for covariance matrix
l1cov.start <- diag(2, 2)

# Set scale matrix of the inverse-Wishart prior for the covariance matrix:
l1cov.prior <- diag(2, 2);

# Set seed to get results below
set.seed(1569)

# Impute:
# [Note for new R users: the inputs set above have the same names as their
# corresponding options in the function. Hence, when we set <option> = <object
# name>, we have the same string on both sides of '=']

imp2 <- jomo(Y, X = X, beta.start = beta.start, l1cov.start = l1cov.start,
             l1cov.prior = l1cov.prior, nburn = 200,
             nbetween = 200, nimp = 5)
# ....................First imputation registered.
# ....................Imputation number  2 registered
# ....................Imputation number  3 registered
# ....................Imputation number  4 registered
# ....................Imputation number  5 registered
#             cons        sex
# english 38.21636  6.4469156
# ravens  25.33982 -0.5456155
#
# The posterior covariance matrix is:
#           english   ravens
# english 460.37902 65.34086
# ravens   65.34086 36.58763
```

We see no material change from the previous results. In simple problems, the default burn-in of `nburn = 1000` is often enough for the sampler to converge. Further below, we show how to visually check whether the stationary distribution has been reached.

Analysing the imputed data

Recall our substantive linear regression model above. Once we have created our imputed data sets, we follow the usual multiple imputation rules and fit our substantive model to each imputed data set, before summarising the results for final inference using Rubin's rules, which are implemented in **mitools**:

```
library(mitools) # load if not done so above

# Use the object imp which we created with the original run above. First
# convert the data frame of results imp to a list of imputations
imp.list <- imputationList(split(imp, imp$Imputation)[-1])

# Fit model to each of the 5 imputed data sets
fit.imp <- with(data = imp.list, lm(english ~ ravens + sex))
```

```
# Extract coefficients and variances
coefs <- MIextract(fit.imp, fun = coef)
vars <- MIextract(fit.imp, fun = function(x) diag(vcov(x)))

# Pool results with Rubin's rules
results <- MIcombine(coefs, vars)
summary(results)

# Multiple imputation results:
#       MIcombine.default(coefs, vars)
#               results        se     (lower     upper) missInfo
# (Intercept) -6.549183 3.0013631 -12.618685 -0.4796807     35 %
# ravens       1.767782 0.1124461   1.540232  1.9953322     36 %
# sex          7.037727 1.3021658   4.424013  9.6514414     31 %
```

There are multiple alternative implementations of Rubin's rules in R. These include `pool` from **mice**, `runMI` from **semTools**, particularly appealing with Structural Equation Models, `MI.inference` from **BaBooN** (Meinfelder, 2011), and `testEstimates` from **mitml**, which we present more in depth in the penultimate section of this paper. Other packages with their own implementation of Rubin's rules include **Amelia**, **mi**, and **lavaan.survey**.

Categorical variables

Fully observed binary covariates can be included in the X matrix of the imputation model as type `numeric`, exactly as with `sex` in this example. To include fully observed categorical covariates with three or more categories, appropriate dummy variables have to be created. For this purpose, we might use the R package **dummies** (Brown, 2012) or the function `constrasts` in base R.

jomo also readily imputes a mix of binary, categorical and continuous variables. This is done using a latent normal model (see Goldstein et al., 2009; Carpenter and Kenward, 2013, Ch. 4). To illustrate this, we continue to use the data set `JSPmiss` but now also impute the partially observed fluency level (3 categories). The underlying joint imputation model remains a multivariate normal model, but `fluent` is represented by two latent normal variables:

$$\begin{aligned} Y_{english,i} &= \beta_{0,e} + \beta_{1,e} X_{sex,i} + \epsilon_{e,i} \\ Y_{ravens,i} &= \beta_{0,r} + \beta_{1,r} X_{sex,i} + \epsilon_{r,i} \\ Y^*_{flu,1,i} &= \beta_{0,f1} + \beta_{1,f1} X_{sex,i} + \epsilon_{f1,i} \\ Y^*_{flu,2,i} &= \beta_{0,f2} + \beta_{1,f2} X_{sex,i} + \epsilon_{f2,i} \end{aligned} \tag{5}$$

where:

$$\begin{aligned} Pr(Y_{flu,i} = 1) &= Pr\left(Y^*_{flu,1,i} = \max_{j=1,2} Y^*_{flu,j,i} \text{ and } Y^*_{flu,1,i} > 0\right) \\ Pr(Y_{flu,i} = 2) &= Pr\left(Y^*_{flu,2,i} = \max_{j=1,2} Y^*_{flu,j,i} \text{ and } Y^*_{flu,2,i} > 0\right) \\ Pr(Y_{flu,i} = 3) &= Pr\left(Y^*_{flu,j,i} < 0 \text{ for } j = 1,2\right), \end{aligned} \tag{6}$$

$$\begin{pmatrix} \epsilon_{e,i} \\ \epsilon_{r,i} \\ \epsilon_{f1,i} \\ \epsilon_{f2,i} \end{pmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Omega}) \tag{7}$$

As highlighted above, in order for this model to be estimable, we need to constrain the variance-covariance matrix of $(\epsilon_{f1,i}, \epsilon_{f2,i})^T$, (i.e. the bottom right 2×2 submatrix of $\mathbf{\Omega}$) to be

$$\begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}.$$

These constraints are automatically implemented in the software.

The code for imputation is essentially the same as before, but now we need to make sure that `fluent`, being a categorical variable, is included in the dependent variable data frame as a `factor`:

```
# convert "fluent" to factor
```

```
JSPmiss <- within(JSPmiss, fluent <- factor(fluent))

# Define the data frame with the dependent (outcome) variables for the imputation model:
Y <- JSPmiss[, c("english", "ravens", "fluent")]

# Define the data.frame with the (fully observed) covariates of the imputation model
X <- JSPmiss[, c("cons", "sex")]


# fit the model and impute the missing data:
set.seed(1569)
imp <- jomo(Y = Y, X = X)
```

This will take a little longer than the previous examples, and returns the following output:

```
# The posterior mean of the fixed effects estimates is:
#               cons        sex
# english  38.263132  6.3636755
# ravens   25.324376 -0.5548762
# fluent.1 -1.574369 -0.2318260
# fluent.2 -1.836201  0.1829313
#
# The posterior covariance matrix is:
#             english    ravens   fluent.1  fluent.2
# english  458.164036 64.437168 -11.379005 -8.888263
# ravens    64.437168 36.743243  -2.141006 -1.500893
# fluent.1 -11.379005 -2.141006   1.000000  0.500000
# fluent.2  -8.888263 -1.500893   0.500000  1.000000
```

The output illustrates that `jomo` recognized that `fluent` was a factor variable and therefore used the function for the imputation of mixed data types.

The matrix posterior mean of the fixed effect estimates links directly to (5). Specifically, $\hat{\beta}_{0,e} = 38.26, \hat{\beta}_{1,e} = 6.36, \ldots, \hat{\beta}_{0,f2} = -1.84, \hat{\beta}_{1,f2} = 0.18$. Likewise, the posterior means of the covariance matrix terms are labelled, and correspond directly to (5). Specifically,

$$\widehat{\text{Var}} \begin{pmatrix} \epsilon_{e,i} \\ \epsilon_{r,i} \\ \epsilon_{f1,i} \\ \epsilon_{f2,i} \end{pmatrix} = \begin{pmatrix} 458.16 & 64.44 & -11.38 & -8.89 \\ 64.44 & 36.74 & -2.14 & -1.50 \\ -11.38 & -2.14 & 1 & 0.5 \\ -8.89 & -1.50 & 0.5 & 1 \end{pmatrix}.$$

Calling the relevant sub-function (`jomo1mix`) is possible again but a bit more complex, because continuous and categorical outcomes must be passed as separate arguments.

We can specify starting values explicitly if we wish. To specify all the starting values, we need to specify $n-1$ starting β values for each n-category variable and a proper starting value for the covariance matrix. For example, in the present case, `fluent` has 3 categories, which are modelled with 2 latent normal variables. As a result, $\boldsymbol{\beta}$ is a 2×4 matrix of regression coefficients, and the covariance matrix is of size 4×4 (i.e., two predictors with two continuous and two latent dependent variables). So, continuing with `Y` and `X` defined as above:

```
# Starting value for beta
beta.start <- matrix(0, 2, 4) # Specify a 2 by 4 matrix of zeros

# Starting value for covariance matrix; the software disregards impossible values:
l1cov.start <- diag(2, 4)

set.seed(1569)
imp <- jomo(Y = Y, X = X, beta.start = beta.start, l1cov.start = l1cov.start)
```

As our starting values are different from the default, we get slightly different estimates of the posterior means.

While the software is designed for unordered categorical data, it can be used for ordinal data too. Our simulation results show that if variables are truly ordinal it gives good results with only a marginal loss in efficiency (Quartagno and Carpenter, 2019).

jomo: tutorial with multilevel data

When we wish to impute missing data with a multilevel substantive model, our imputation model should itself preserve the multilevel structure (Lüdtke et al., 2017; Andridge, 2011); one key benefit of **jomo** is that it allows us to do this.

Three different approaches for multilevel multiple imputation have been implemented in **jomo**, providing a flexible framework for the treatment of missing data in multilevel data sets. They allow:

1. imputation with a common level-1 covariance matrix across level-2 units (the default);
2. imputation with a cluster-specific level-1 covariance matrices, and
3. imputation allowing for the level-1 covariance matrix to be randomly distributed across level-2 units, following the proposal by Yucel (2011) and as developed by Quartagno and Carpenter (2016).

Option (2) requires sufficient data within each level-2 unit to estimate the covariance matrix; option (3) is a practical choice when we suspect there is heterogeneity across level-2 units, but there is insufficient information within each level-2 unit for option (2).

We illustrate the software again with the `JSPmiss` data set, distributed with the package. Let j index school and i students within schools. Our substantive model is:

$$Y_{english,i,j} = \alpha_0 + \alpha_1 \text{ravens}_{i,j} + \alpha_2 \text{sex}_{i,j} + \alpha_3 1[\text{fluent}_{i,j} == 2] + \alpha_4 1[\text{fluent}_{i,j} == 3] + u_j + \epsilon_{i,j} \tag{8}$$

$$\begin{aligned} u_j &\sim N(0, \sigma_u^2) \\ e_{i,j} &\sim N(0, \sigma_e^2). \end{aligned} \tag{9}$$

We now use multilevel imputation for the missing data. This is done by extending the joint imputation model described above to the multilevel setting. As before, the variables `english`, `ravens` and `fluent` are responses, and the fully observed variable `sex` a covariate. The imputation model has random intercepts at level 2, and a common level-1 covariance matrix (approach 1 above).

$$\begin{aligned} Y_{english,i,j} &= \beta_{0,e} + \beta_{1,e} X_{sex,i,j} + u_{e,j} + \epsilon_{e,i,j} \\ Y_{ravens,i,j} &= \beta_{0,r} + \beta_{1,r} X_{sex,i,j} + u_{r,j} + \epsilon_{r,i,j} \\ Y^*_{flu,1,i,j} &= \beta_{0,f1} + \beta_{1,f1} X_{sex,i,j} + u_{f1,j} + \epsilon_{f1,i,j} \\ Y^*_{flu,2,i,j} &= \beta_{0,f2} + \beta_{1,f2} X_{sex,i,j} + u_{f2,j} + \epsilon_{f2,i,j} \end{aligned} \tag{10}$$

where:

$$\begin{aligned} Pr(Y_{flu,i,j} = 1) &= Pr\left(Y^*_{flu,1,i,j} = \max_{k=1,2} Y^*_{flu,k,i,j} \text{ and } Y^*_{flu,1,i,j} > 0\right) \\ Pr(Y_{flu,i,j} = 2) &= Pr\left(Y^*_{flu,2,i,j} = \max_{k=1,2} Y^*_{flu,k,i,j} \text{ and } Y^*_{flu,2,i,j} > 0\right) \\ Pr(Y_{flu,i,j} = 3) &= Pr\left(Y^*_{flu,k,i,j} < 0 \text{ for } k = 1, 2\right) \end{aligned} \tag{11}$$

$$\boldsymbol{\epsilon}_{i,j} = \begin{pmatrix} \epsilon_{e,i,j} \\ \epsilon_{r,i,j} \\ \epsilon_{f1,i,j} \\ \epsilon_{f2,i,j} \end{pmatrix} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}_e) \qquad \mathbf{u}_j = \begin{pmatrix} u_{e,j} \\ u_{r,j} \\ u_{f1,j} \\ u_{f2,j} \end{pmatrix} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}_u) \tag{12}$$

This is the same as model (5), apart from the addition of the covariance matrix $\boldsymbol{\Omega}_u$ for the vector of random intercepts, $\mathbf{u}_j$.

Apart from specifying the level-two identifier, imputing the missing values proceeds the same as before:

```
# Define cluster/group indicator
clus <- JSPmiss$school

# Define the data.frame with outcomes of the imputation model
JSPmiss <- within(JSPmiss, fluent <- factor(fluent))
Y <- JSPmiss[, c("english", "ravens", "fluent")]

# Define the data frame with covariates of the imputation model
```

```
X <- JSPmiss[, c("cons", "sex")]

# Perform multilevel imputation:
set.seed(1569)
imp <- jomo(Y = Y, X = X, clus = clus, nburn = 2000, nbetween = 1000, nimp = 5)

# Clustered data, using functions for two-level imputation.
# Found  2 continuous outcomes and  1 categorical. Using function jomo1ranmix.
# ..................................................................................
# ..................................................................................
# .........................First  imputation registered.
# ..................................................................................
# ............Imputation number  2 registered
# ..................................................................................
# ............Imputation number  3 registered
# ..................................................................................
# ............Imputation number  4 registered
# ..................................................................................
# ............Imputation number  5 registered
# The posterior mean of the fixed effects estimates is:
#               cons        sex
# english  38.145706  5.7460263
# ravens   25.207758 -0.5280992
# fluent.1 -1.889772 -0.1980197
# fluent.2 -2.133932  0.3046025
#
# The posterior mean of the random effects estimates is:
#     english.Z1   ravens.Z1 fluent.1.Z1  fluent.2.Z1
# 1   -9.3394300 -2.39167726  0.30978733  0.359526051
# 2   -0.6734767  0.09111794  0.16371750 -0.140409675
# 3    2.0366252  1.66523294 -0.52408207 -0.433563083
# 4   -0.7524001  0.15352258 -0.14147716  0.016889639
# 5    7.4086502  1.79372314 -0.46164853  0.029898809
# [...]
# 46   7.6419790  1.93470732 -0.43765561  0.099247985
# 47   0.1680530  0.56124947  1.02219296  1.083265124
# 48  -1.9106040 -0.10743067 -0.24463642 -0.101618838
# 49   1.5480441  0.23546197 -0.31258927 -0.072163795
# 50   2.8932869  1.21095460 -0.46288361 -0.351124776
#
# The posterior mean of the level 1 covariance matrices is:
#             english    ravens  fluent.1  fluent.2
# english  394.192120 53.461640 -9.709166 -7.894670
# ravens    53.461640 32.882006 -1.636866 -1.391647
# fluent.1  -9.709166 -1.636866  1.000000  0.500000
# fluent.2  -7.894670 -1.391647  0.500000  1.000000
#
# The posterior mean of the level 2 covariance matrix is:
#             english.Z1  ravens.Z1 fluent.1.Z1 fluent.2.Z1
# english*Z1   67.791552 13.5839586  -4.1394619  -2.2062157
# ravens*Z1    13.583959  4.1667944  -1.0068761  -0.5742672
# fluent.1*Z1  -4.139462 -1.0068761   0.6051437   0.3374155
# fluent.2*Z1  -2.206216 -0.5742672   0.3374155   0.4628897
```

Simply including `clus` was enough to tell `jomo` that we have a multilevel structure and to impute accordingly. The assumed model included a random intercept for each outcome variable; we will see later in this section how to specify the design matrix for the random effects differently. The output follows the same format as that from fitting (5) above, with two additions:

1. we obtain the posterior mean of the $\mathbf{u}_j$, the random intercepts for each of the 5 responses, for

each of the $j = 1, \ldots, 50$ schools. For example, for school 2, the posterior means are

$$\begin{pmatrix} u_{e,2} \\ u_{r,2} \\ u_{f1,2} \\ u_{f2,2} \end{pmatrix} = \begin{pmatrix} -0.67 \\ 0.09 \\ 0.16 \\ -0.14 \end{pmatrix}$$

2. we obtain the estimated level-2 variance covariance matrix of the random intercepts (common across all ten cities):

$$\widehat{\text{Var}} \begin{pmatrix} u_{e,j} \\ u_{r,j} \\ u_{f1,j} \\ u_{f2,j} \end{pmatrix} = \begin{pmatrix} 67.79 & 13.58 & -4.14 & -2.21 \\ 13.58 & 4.17 & -1.01 & -0.57 \\ -4.14 & -1.01 & 0.61 & 0.34 \\ -2.21 & -0.57 & 0.34 & 0.46 \end{pmatrix}.$$

Analysing the imputed data

To fit the substantive multilevel model to each imputed data set we proceed as before:

```
imp.list <- imputationList(split(imp, imp$Imputation)[-1])

# Fit model to each of the 5 imputed data sets
fit.imp <- with(data = imp.list, lmer(english ~ ravens + sex + factor(fluent) + (1|clus)))

# Extract coefficients and variances
coefs <- MIextract(fit.imp, fun = fixef)
vars <- MIextract(fit.imp, fun = function(x) diag(vcov(x)))

# Pool results with Rubin's rules
results <- MIcombine(coefs, vars)
summary(results)

# Multiple imputation results:
#       MIcombine.default(coefs, vars)
#                     results        se     (lower    upper) missInfo
# (Intercept)     -16.133067 3.7371534 -23.521995 -8.744139     18 %
# ravens            1.622946 0.1197557   1.374339  1.871554     48 %
# sex               6.837264 1.1452177   4.573386  9.101142     18 %
# factor(fluent)1   5.000691 4.2413878  -3.420624 13.422006     22 %
# factor(fluent)2  14.345070 3.1391521   8.057088 20.633051     29 %
```

In order to get multiple imputation inference for the random coefficients, we recommend using the **mitml** package, as described in the penultimate section below.

Design matrix for random effects

We now show how to specify additional random effects in the imputation model, apart from the random intercept that is included by default. This is done by specifying the design matrix of the random effects, Z. When this is not specified, it defaults to a random intercept. When it is specified by the user, the random intercept has to be included (if desired).

```
Z <- JSPmiss[, c("cons", "sex")] # intercept and sex have random effects

imp <- jomo(Y = Y, X = X, Z = Z, clus = clus)
# Output omitted
```

Starting values and prior distributions

The starting values of the sampling algorithm can again be overridden by the user, thus potentially leading to better sampling behaviour and faster convergence. In comparison with the single-level case, we now have two additional sets of parameters: (i) the matrix of random effects, whose rows contain the random effects vectors for all level-2 units (`u.start`), and (ii) the level-2 covariance matrix

(`l2cov.start`). Note that with small numbers of level-2 units, the impact of the scale matrix of the prior for the level-2 covariance matrix can be substantial. We proceed as follows:

```
beta.start <- matrix(1, 2, 4) # initialise fixed effects to zero
u.start <- matrix(0.5, nlevels(JSPmiss$school), 4) # initialise all random effects to 0.5
l1cov.start <- diag(2, 4)     # initialise diagonal covariance matrix of 2's for level 1
l2cov.start <- diag(2, 4)     # initialise diagonal covariance matrix of 2's for level 1
l2cov.prior <- diag(2, 4);    # set scale matrix of inverse-Wishart prior for level-2
                              # covariance matrix

set.seed(1569)
imp <- jomo(Y = Y, X = X, clus = clus, beta.start = beta.start, u.start = u.start,
            l1cov.start = l1cov.start, l2cov.start = l2cov.start,
            l2cov.prior = l2cov.prior, nburn = 2000, nbetween = 1000, nimp = 5)

# Output omitted; as these are not all the default values, the posterior means differ
# from the previous results by Monte Carlo error.
```

Cluster-specific covariance matrices

In some cases, it is implausible that all of the clusters share the same level-1 covariance matrix. For example, when aggregating individual patient data from different studies to perform a meta-analysis, it is often reasonable to assume that covariance matrices are different across studies.

Continuing to use (10) as an example, the only difference from before is that now the level-1 covariance matrix is not modelled as constant but as different across level-2 units. Thus instead of $\mathbf{\Omega}_e$, we have $\mathbf{\Omega}_{e,j}$, $j = 1, \ldots, 50$. We fit this model by specifying the additional argument `meth = "fixed"`:

```
# Define the data.frame with outcomes of the imputation model
Y <- JSPmiss[, c("english", "ravens", "fluent")]

# Define the data.frame with covariates of the imputation model
X <- JSPmiss[, c("cons", "sex")]

# Define cluster/group indicator
clus <- JSPmiss$school

# Fixed cluster-specific covariance matrices
imp2 <- jomo(Y = Y, X = X, clus = clus, meth = "fixed")
# Output omitted
```

Note that the output is now considerably longer, as the posterior mean for each of the level 1 covariance matrices is reported. Compared to the previous models, this model is more complex and requires estimation of a larger number of parameters.

Random cluster-specific covariance matrices

There are several reasons why we may wish to go beyond the setting above and allow the covariance matrices to be random across level-2 units. For example, we may have reason to believe that different level-2 units have different covariance matrices but that the number of observations on some of these level-2 units is insufficient to estimate level-2 specific covariance matrices reliably. In this case, sharing information across level-2 units is desirable. Another situation is when some variable is fully missing from some clusters, and therefore it is necessary to share information with clusters where it was observed through the specification of a hierarchical distribution for the covariance matrices.

Continuing to use (10) as an example, the cluster-specific covariance matrices are now assumed to follow a specific distribution. Thus instead of $\mathbf{\Omega}_{e,j}$, we have $\mathbf{\Omega}_{e,j} \sim IW(a, S)$ $j = 1, \ldots, 10$. Here, a and S are the degrees of freedom and scale matrix of the inverse-Wishart distribution, respectively.

We can simply fit this model by specifying the option `meth = "random"`:

```
imp3 <- jomo(Y = Y, X = X, clus = clus, meth = "random")
```

and then analyse the imputed data sets in the usual way. For full details on the algorithm used to fit the random covariance matrices algorithm initially proposed by Yucel (2011), see the appendix of Quartagno and Carpenter (2016).

The sub-function called by `jomo` for this type of data is `jomo1ranmixhr`, which can be called directly but with a slightly more complex syntax.

With random covariance matrices, we have one further parameter, `a`, denoting the degrees of freedom of the inverse-Wishart distribution for the cluster-specific covariance matrices. The default starting value for this parameter, `a.start`, is the minimum possible, i.e., the dimension of the level-1 covariance matrix. This is also the default for the hyperparameter `a.prior` of the chi-square prior distribution for a.

With random level-1 covariance matrices we can also specify starting values for the n_{clus} covariance matrices. Below, we show how to do this by (i) first creating the matrix for the first level 2 unit, a 4-by-4 diagonal matrix with all entries '2', using `diag(2,4)` then (ii) stacking 50 copies of this:

```
# Starting values for the 5 by 5 level-1 covariance matrix for the first level-2 unit
l1cov.start.1 <- diag(2, 4)
# Stack 10 copies of this matrix (one for each of the level-2 units)
l1cov.start <- matrix(l1cov.start.1, nrow = 4 * nlevels(JSPmiss$school), ncol = 4,
                      byrow = TRUE)

# Choose a starting value for the degrees of freedom, a (automatically >= 5)
a.start <- 7

# Run jomo
imp <- jomo(Y = Y, X = X, clus = clus, l1cov.start = l1cov.start, a = a.start,
            meth = "random")
```

Imputing level-2 variables

In many applications, we have variables describing aspects of the level-2 units, and these may also have missing values. For example, in longitudinal studies, time-independent variables related to individuals (level-2 units), such as sex or the baseline variable `ravens`, may be affected by missing data. We can impute any missing level-2 values naturally with **jomo** (Carpenter and Kenward, 2013, Ch.9). As described above, we can use either a single common or multiple cluster-specific level-1 covariance matrices.

To illustrate this, we use a new data set, `ExamScores`. The fully observed version of this data set is again available with MLwiN and **R2MLwiN**, and it represents a subset of a larger data set of examination results from six inner London Education Authorities. The partially observed version that we use here is available with **jomo**. As in the previous example, this data set contains data from pupils (level 1) clustered in schools (level 2). Some of the variables are related to students at level 1 (`normexam`, a normalised version of exam score at age 16 and `standlrt`, London Reading Test (LRT) score at age 11). The other variables describe features of the schools at level 2, for example `avslrt` (continuous), representing the average LRT score for pupils in a particular school.

The two-level multivariate normal joint model for `normexam` (n), `standlrt` (s) and `avslrt` (a) is:

$$\begin{aligned} Y_{normexam,i,j} &= \beta_{0,n} + u^{(1)}_{n,j} + \epsilon_{n,i,j} \\ Y_{standlrt,i,j} &= \beta_{0,s} + u^{(1)}_{s,j} + \epsilon_{s,i,j} \\ Y_{avslrt,j} &= \beta 0, a + u^{(2)}_{a,j} \end{aligned} \tag{13}$$

$$\boldsymbol{\epsilon}_{i,j} = \begin{pmatrix} \epsilon_{n,i,j} \\ \epsilon_{s,i,j} \end{pmatrix} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}_e) \qquad \mathbf{u}_j = \begin{pmatrix} u^{(1)}_{n,j} \\ u^{(1)}_{s,j} \\ u^{(2)}_{a,j} \end{pmatrix} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}_u), \tag{14}$$

where the superscripts (1) and (2) indicate the level-2 random effect for a level-1 and level-2 covariate, respectively (level-2 covariates have no level-1 residuals).

Fitting this model is very similar to fitting previous models; we simply have to define the level-2 variables.

```
# Define data.frame with level-1 outcomes of imputation model
Y <- ExamScores[, c("normexam", "standlrt")]

# Define data.frame with level-2 outcomes of imputation model
```

```
Y2 <- ExamScores[, "avslrt", drop = FALSE]

# Define clustering indicator
clus <- ExamScores$school

# Run jomo
set.seed(1569)
imp <- jomo(Y = Y, Y2 = Y2, clus = clus)

# 2-level data, using functions for two-level imputation.
# Found  2 level 1 continuous and  0 level 1 categorical outcomes, 1  level 2 continuous
# and  0  level 2 categorical outcomes. Using function jomo2com, assuming common
# covariance matrix across clusters

# Output partially omitted [...]

# The posterior mean of the fixed effects estimates is:
#                    X1
# normexam -0.009685026
# standlrt -0.002263212

# The posterior mean of the level 2 fixed effects estimates is:
#               X2.1
# avslrt -0.03268919

# The posterior mean of the random effects estimates is:
#     normexam.Z1  standlrt.Z1       avslrt
# 1   0.485259093  0.146952219  0.198863738
# 2   0.896523031  0.378616363  0.429914831
# 3   0.871480465  0.464155974  0.546844580
# [...]
# 63  0.586499213  0.161779179  0.188900255
# 64  0.310554391  0.391077349  0.466833301
# 65 -0.320429589 -0.086303587 -0.202661000

# The posterior mean of the level 1 covariance matrix is:
#           normexam  standlrt
# normexam 0.8536697 0.5045721
# standlrt 0.5045721 0.8876558

# The posterior mean of the level 2 covariance matrix is:
#             normexam.Z1 standlrt.Z1     avslrt
# normexam*Z1   0.2023980  0.10005877 0.10599527
# standlrt*Z1   0.1000588  0.12408893 0.09844939
# avslrt        0.1059953  0.09844939 0.13067251
```

As above, we can specify the starting values for all the parameters in the model, and in particular the parameter of the level-2 variable β_2 with the input `l2.beta.start`. As we noted above, with small cluster sizes, the scale matrix for the prior of the level 2 covariance matrix, `l2cov.prior`, may have a non-negligible impact on the results.

The sub-functions for imputation of level 2 variables are `jomo2com` and `jomo2hr`. Cluster-specific covariance matrices can be specified as before by setting `meth = "common"` or `meth = "random"`.

Checking convergence of MCMC

When using MCMC for model fitting and imputation, it is crucial to be confident of having reached the stationary distribution of the sampler before starting to register imputations. We do this by monitoring the parameter chains generated by the MCMC algorithm. To facilitate this, we introduced a `.MCMCchain` version of each function in the package, which allows convergence assessment without imputation. We illustrate this with a simple example:

```
# Define data.frames with outcomes and covariates of imputation model
```

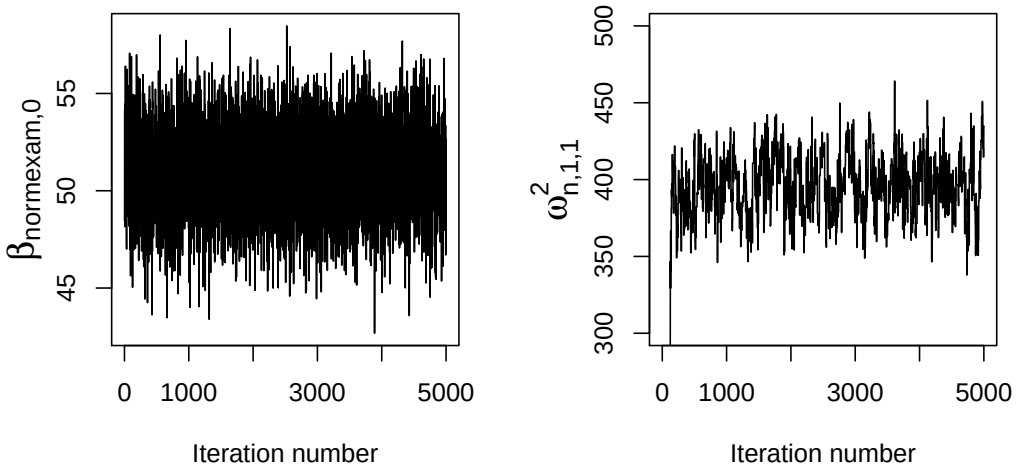

Figure 1: MCMC chain for $\beta_{e,0}$ (left panel) and $\omega^2_{e,1,1}$ (right panel).

```
Y <- JSPmiss[, c("english", "ravens")]
X <- JSPmiss[, c("cons", "sex")]

# Run jomo.MCMCchain
imp <- jomo.MCMCchain(Y = Y, X = X, nburn = 5000)
```

This updates the sampler `nburn` times, but does not create any imputed data sets. Instead, the output of this function is a list containing three elements:

- `finimp`: the final state of the data set, which would be the first imputation if we ran the `jomo` function with `nburn` burn-in iterations;
- `collectbeta`: a three-dimensional array containing the fixed effect parameter draws at each of the `nburn` iterations;
- `collectomega`: a three-dimensional array containing the level-1 covariance matrix draws at each of the `nburn` iterations;

When running the corresponding `.MCMCchain` functions for multilevel imputation we will also have:

- `collectu`: a three-dimensional array containing the random effects draws at each of the `nburn` iterations;
- `collectcovu`: a three-dimensional array containing the level-2 covariance matrix draws at each of the `nburn` iterations;

We can then check the convergence of the sampler by looking at the trace plot for each parameter value. For example, in Figure 1 (left panel), we can see the plot for $\beta_{e,0}$, which we obtain by running:

```
plot(imp$collectbeta[1, 1, 1:5000], type = "l", ylab = expression(beta["e,0"]),
     xlab = "Iteration number" )
```

In this case, we can see that a burn in of 100–500 is reasonable; the sampler clearly converges very quickly.

Plots for elements of the covariance matrix updated though Metropolis-Hastings steps may look different, because these chains have higher auto-correlation (as they are not guaranteed to be updated at each iteration). The right panel of Figure 1 gives an example; this was obtained by running the following commands:

```
JSPmiss <- within(JSPmiss, fluent <- factor(fluent))
Y <- JSPmiss[, c("english", "ravens", "fluent")]
X <- JSPmiss[, c("cons", "sex")]
```

```
imp2 <- jomo.MCMCchain(Y = Y, X = X, nburn = 5000)

plot(imp2$collectomega[1, 1, 1:5000], type = "l", ylab = expression(omega[e,1,1]^2),
     xlab = "Iteration number", ylim=c(300,500) )
```

Note there is little point in plotting the constrained elements of the covariance matrix — these will always give a straight line!

Using jomo in practice

The `.MCMCchain` functions only register a single imputation, but the state of the sampler at this point is captured. This provides a mechanism for combining multiple runs of `.MCMCchain` and/or `jomo` in a flexible manner, for example, to obtain the full set of posterior draws for the model parameters with multiple runs of `.MCMCchain` or to generate mildly informative prior distributions to be used with jomo. Specifically, at the end of the `.MCMCchain` run, the following objects capture the state of the MCMC sampler:

- `start.imp` for the level-1 variables with missing values;
- (where present) `l2.start.imp` for level-2 variables with missing values, and
- `finimp.latnorm`: the final state of the imputed data set using latent normals in place of categorical variables.

In practice, we typically need to use `finimp.latnorm`, together with the last value of the fixed parameters and the covariance matrix at level 1 (and at level 2 if present). The following code illustrates the approach:

```
# Define data frames for outcomes and covariates of imputation model and
# convert "fluent" to factor
JSPmiss <- within(JSPmiss, fluent <- factor(fluent))
Y <- JSPmiss[, c("english", "ravens", "fluent")]
X <- JSPmiss[, c("cons", "sex")]

# Run jomo to register 2 imputations
set.seed(1569)
imp <- jomo(Y = Y, X = X, nimp = 2)

# OR, run jomo.MCMCchain to register first imputation
set.seed(1569)
imp1 <- jomo.MCMCchain(Y = Y, X = X)

# Capture the state of the sampler as starting values for the second set of iterations:
beta.start <- imp1$collectbeta[,,1000]   # capture the fixed parameter values
l1cov.start <- imp1$collectomega[,,1000] # capture the level-1 covariance matrix values
start.imp <- imp1$finimp.latnorm         # capture the final imputed data set (with
                                         # latent normals for categorical variables)

# Run jomo.MCMCchain to register second imputation
imp2 <- jomo.MCMCchain(Y = Y, X = X, beta.start = beta.start, l1cov.start = l1cov.start,
                       start.imp = start.imp, nburn = 1000)
```

In practice, it often works well to use this function to find plausible initial values for the scale matrices of the level-1 and level-2 covariance matrix priors. This allows us to provide 'weakly informative' priors consistent with the data, and avoids imputations being unnecessarily variable.

To do this, we run `jomo.MCMCchain` first, using the default prior. We retain the last draw (or the posterior mean of the latter part of the chain) as the covariance matrix prior. We use these to assign values to `l1cov.prior` or `l2cov.prior` and then we apply `jomo` as usual. Specifically:

```
# Define data frame as usual
JSPmiss <- within(JSPmiss, fluent <- factor(fluent))
Y <- JSPmiss[, c("english", "ravens", "fluent")]
X <- JSPmiss[, c("cons", "sex")]

# Run jomo.MCMCchain with default prior
```

```
imp1 <- jomo.MCMCchain(Y = Y, X = X)

# Collect posterior mean of covariance matrix
l1cov.guess <- apply(imp1$collectomega, c(1, 2), mean)
# Multiply by degrees of freedom, i.e. dimension of the matrix (4), to get scale matrix
l1cov.prior <- l1cov.guess*4

# Run jomo
imp <- jomo(Y = Y, X = X, l1cov.prior = l1cov.prior)
```

However, if a proper prior guess for the value of all the parameters was available, this would be preferable, as it would avoid using the same data twice, to fit the model and to estimate the hyperparameters of the priors.

When using **jomo**, we recommended the following workflow:

1. Before running the imputation model (which may take some time), perform a "dry run", to check the software is fitting the model we intended. We can do this using the `.MCMCchain` function with `nburn = 2` and checking the output.
2. Re-run the same function for a larger number of iterations (e.g. 5000) and analyse the resulting trace and autocorrelation plots to choose a sensible number of burn-in and between-imputation iterations for the final imputation process.
3. Run the **jomo** function for the chosen number of iterations.
4. Fit the substantive model on the imputed data sets and apply Rubin's rules.

mitml: an alternative interface to jomo

The **mitml** package (Grund et al., 2016c) provides an alternative interface to joint modeling multiple imputation with **jomo**. Originally created as an interface to the **pan** package, **mitml** also provides access to most of the features implemented in **jomo** and includes a number of additional tools for managing, visualising, and analysing multiply imputed data sets.

Specification of the imputation model

The main interface to **jomo** is provided by the function `jomoImpute`, which offers two convenient ways of specifying the imputation model. The first option uses a `formula`-based syntax similar to packages for multilevel modelling such as **lme4**, **nlme** (Pinheiro et al., 2017), and others. The following operators can be used to define such a `formula`:

- `~` : separates the dependent variables (left-hand side) and predictor variables (right-hand side) of the imputation model;
- `+` : adds dependent and predictor variables to the model;
- `*` : adds interactions of two or more predictors to the model;
- `|` : specifies the cluster indicator and adds cluster-specific random effects to the model (e.g., `1|school`), and
- `I()` : defines additional transformation of predictor variables to be included in the model.

For example, to fit the imputation model in (10) for the substantive model in (8) with `JSPmiss`, the model `formula` can be specified as:

```
fml <- english + ravens + fluent ~ sex + (1|school)
```

The imputation is then run with `jomoImpute` by specifying the incomplete data, the imputation model, the number of imputations (`m`), and the number of iterations for burn-in (`n.burn`) and between imputations (`n.iter`). Like **jomo**, `jomoImpute` requires that categorical variables are formatted as `factors`.

```
# Convert "fluent" to factor
JSPmiss <- within(JSPmiss, fluent <- factor(fluent))

# Run imputation
imp <- jomoImpute(data = JSPmiss, formula = fml, n.burn = 1000, n.iter = 1000, m = 5,
                  seed = 1569)
```

In addition, imputation models can be run independently in subsets of the data. For this purpose, `jomoImpute` includes an optional `group` argument, denoting the name of a variable in the data set. If specified, the imputation model is run separately for each level of `group`.

As an alternative to specifying a `formula`, the imputation model can be specified with the `type` argument of `jomoImpute`. The `type` argument is an integer vector denoting the role of each variable in the imputation model. The following values are supported:

- `1` : dependent variables (fully or partially observed);
- `2` : predictor variables (fully observed) with fixed effect;
- `3` : predictor variables (fully observed) with fixed and random effect;
- `-1` : (optional) grouping variable; if specified, imputations are run separately within each group;
- `-2` : cluster indicator, and
- `0` : variables excluded from the imputation model.

In applications with missing data at both level 1 and 2, `formula` and `type` are specified as a list of two formulas or type vectors, denoting the imputation model for variables at level 1 and 2, respectively.

Analysis of the imputed data sets

The **mitml** package can also be used to manage, visualise, and analyse the imputed data. For example, the `summary` and `plot` methods display information about the imputed data object and the convergence of the MCMC algorithm.

```
# Summarize model and display convergence statistics
summary(imp)

# Call:

# jomoImpute(data = JSPmiss, formula = fml, n.burn = 1000, n.iter = 1000,
#     m = 5, seed = 1569)
#
# Cluster variable:         school
# Target variables:         english ravens fluent
# Fixed effect predictors:  (Intercept) sex
# Random effect predictors: (Intercept)
#
# Performed 1000 burn-in iterations, and generated 5 imputed data sets,
# each 1000 iterations apart.
#
# Potential scale reduction (Rhat, imputation phase):
#
#          Min   25%  Mean Median   75%   Max
# Beta:  1.000 1.002 1.009  1.011 1.013 1.021
# Psi:   1.001 1.002 1.006  1.004 1.006 1.019
# Sigma: 1.000 1.002 1.054  1.014 1.058 1.258
#
# Largest potential scale reduction:
# Beta: [1,3], Psi: [4,2], Sigma: [3,1]
#
# Missing data per variable:
#     school english ravens fluent id sex behaviour cons
# MD% 0      21.1    22.0   21.0   0  0   0         0
#
# Display convergence plots (not shown here)
plot(imp, trace = "all")
```

The function `mitmlComplete` can be used to extract a list of imputed data sets. Each data set can be transformed and analysed with the functions `with` and `within` similar to base R. For example, the following code extracts the imputed data and fits the model in (8) to each of the data sets:

```
# Extract list of completed data sets
imp.list <- mitmlComplete(imp, print = "all")
```

```
# Fit the substantive model to each of the imputed data sets
fit.imp <- with(imp.list, lmer(english ~ ravens + sex + fluent +  (1|school)))
```

Finally, **mitml** allows pooling the results obtained from the imputed data sets. For example, `testEstimates` can be used to pool the estimates of individual parameters such as fixed effects and variance components (Rubin's rules).

```
testEstimates(fit.imp, var.comp = TRUE)

# Call:
#
# testEstimates(model = fit.imp, var.comp = TRUE)
# Final parameter estimates and inferences obtained from 5 imputed data sets.
#              Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
# (Intercept)   -16.133     3.737    -4.317   139.189     0.000     0.204     0.181
# ravens          1.623     0.120    13.552    21.626     0.000     0.755     0.476
# sex             6.837     1.145     5.970   142.006     0.000     0.202     0.179
# fluent1         5.001     4.241     1.179    94.051     0.241     0.260     0.223
# fluent2        14.345     3.139     4.570    56.201     0.000     0.364     0.292
#
#                             Estimate
# Intercept~~Intercept|school   32.231
# Residual~~Residual           291.538
# ICC|school                     0.099
# Unadjusted hypothesis test as appropriate in larger samples.
```

Many different pooling methods are supported by **mitml**, including Rubin's rules with and without correction for smaller samples (`testEstimates`), pooled Wald and likelihood-ratio tests (LRTs) for multiple parameters and model comparisons (`testModels`, `anova`), and tests of constraints on the model parameters via the "delta method" (`testConstraints`, see Casella and Berger, 2002).

Note that, in order to use **mitml** directly with **jomo**, the imputed data must be converted to the `mitml.list` format. This conversion can be achieved with the function `jomo2mitml.list`.

Simulations and applications

The **jomo** package has been extensively evaluated in simulation studies and has been used in various applications. In this section, we provide a brief overview of these studies. For continuous data, Quartagno and Carpenter (2016); Audigier et al. (2018); Grund et al. (2018c,b) showed that jomo provides accurate parameter estimates and inferences. Quartagno and Carpenter (2019); Audigier et al. (2018); Grund et al. (2018c,b) provided similar results for binary categorical data, and Quartagno and Carpenter (2019) showed that the same procedures can be used for categorical and ordinal data. Similar findings were reported by Grund et al. (2018a,b,c) for missing data at level 2 and by Quartagno and Carpenter (2016) for applications with group-specific fixed or random covariance matrices at level 1. Further, **jomo** has been used to study the performance of MI for handling missing data in clustered randomised trials (Hossain et al., 2017a,b) and matched case-control studies (Seaman and Keogh, 2015). Finally, it has been used in applications, particularly, but not exclusively, for imputation of missing data in individual patient data meta-analyses (e.g., (Bloos et al., 2017)).

Conclusions and further developments

In this article, we have introduced a flexible new package for performing joint modelling multiple imputation for multilevel data. This package provides three important contributions: (i) it handles mixed data types, including continuous, categorical and binary data in a flexible way, (ii) it allows for either a common level-1 covariance matrix across level-2 units , cluster-specific level-1 covariance matrices, or random level-1 covariance matrices, and (iii) it gives valid imputation of missing values on level-2 variables. This makes **jomo** an effective choice for treating missing data in many applications, including single-level and multilevel data, cross-sectional and longitudinal data, and meta-analyses with individual participant data.

As with all statistical techniques, multiple imputation has to be used carefully. In particular:

- We should check that the stationary distribution has been reached, before acting on our results. As described above, the package provides tools to facilitate this.

- With many (level-1) variables and relatively few observations, a careful choice of the prior for the level-1 covariance matrix is important. We recommend a weakly informative prior and, in particular, following the strategy described above, running the `.MCMCchain` functions to find a sensible choice for the scale matrices for the inverse-Wishart priors.
- Like most imputation software, ours assumes that data are MAR. If data are MNAR, the results of analyses under MAR may be biased.

In this paper we present functions for multilevel imputation, which make better and more efficient use of all the available data compared to ad-hoc strategies, like imputing including cluster as a fixed effect or imputing separately by cluster. The first approach has been discussed in various publications (Lüdtke et al., 2017; Audigier et al., 2018; Drechsler, 2015) which broadly concluded that the approach is unsatisfactory for small clusters and low intra-cluster correlation. Additionally, under this approach dealing with random slopes is problematic and it is not possible to impute systematically missing variables. Similar considerations are likely to hold for the second strategy consisting in imputing separately by cluster, with the additional complication that level 2 variables cannot be imputed with this method.

If we are imputing a variable which has a random slope in the substantive model (Grund et al., 2016a), then (i) as usual, this variable will be a response in the imputation model and (ii) we should also allow its association with the outcome to be cluster-specific in the imputation model by allowing the level-1 covariance matrix to be random across level-2 units. However, although this approach performs better than a simpler one using a common covariance matrix, it is not a perfectly compatible approach (Quartagno and Carpenter, 2018; Enders et al., 2018), and functions for substantive model compatible imputation (Goldstein et al., 2014) should be preferred to impute missing data in those settings, when possible. These have been recently added to **jomo** and they will be presented in a second paper soon. When interactions or non-linear terms are present in the model of interest, ignoring them in the imputation model may lead to bias; instead, they should be included as covariates (Carpenter and Kenward, 2013, p. 130). When these terms involve partially observed variables, the solution consists again in using substantive model compatible functions.

When using functions for random cluster-specific covariance matrices, users should note that this specifies an inverse-Wishart distribution matrix for the level-1 covariance matrices across the level-2 units. Our simulations (Quartagno and Carpenter, 2016) suggest when this assumption is not appropriate there will be some (usually immaterial) loss of efficiency. In principle, **jomo** could be extended to incorporate other distributions.

All the illustrated functions make use of either Gibbs or Metropolis-Hastings sampling; however, other sampling algorithms such as Hamiltonian Monte-Carlo may provide interesting alternatives in the future.

Future updates and additions to the package will be advertised on `www.missingdata.org.uk`, together with an up-to-date list of publications related to the package. We hope the package is useful to readers and welcome their feedback.

Sources of funding

Matteo Quartagno was supported by funding from the European Community's Seventh Framework Programme FP7/2011: Marie Curie Initial Training Network MEDIASRES ("Novel Statistical Methodology for Diagnostic/Prognostic and Therapeutic Studies and Systematic Reviews"; www.mediasres-itn.eu) with the Grant Agreement Number 290025.
James Carpenter is supported by the MRC grant MC_UU_12023/21

Aknowledgements

The authors of the package would like to thank Christopher Charlton and Professor Harvey Goldstein from Bristol University for their help in creating the package. We would like to thank Alexander Robitzsch, Vincent Audigier, Anower Hossain, Manuel Gomes, Nicole Erler and all the other people that found bugs and imperfections in the code.

Bibliography

P. Mortimore, P. Sammons, L. Stoll, D. Lewis, and R. Ecob. *School Matters.* Wells: Open Books., 1988. [p4]

R. R. Andridge. Quantifying the Impact of Fixed Effects Modeling of Clusters in Multiple Imputation for Cluster Randomized Trials. *Biom J*, 53(1):57–74, 2011. [p10]

V. Audigier, I. R. White, S. Jolani, T. P. A. Debray, M. Quartagno, J. Carpenter, S. van Buuren, and M. Resche-Rigon. Multiple imputation for multilevel data with continuous and binary variables. *Statist. Sci.*, 33(2):160–183, 2018. URL https://doi.org/10.1214/18-sts646. [p20, 21]

F. Bloos, H. Ruddel, D. Thomas-Ruddel, D. Schwarzkopf, C. Pausch, S. Harbarth, T. Schreiber, M. Grundling, J. Marshall, P. Simon, M. M. Levy, M. Weiss, A. Weyland, H. Gerlach, T. Schurholz, C. Engel, C. Matthaus-Kramer, C. Scheer, F. Bach, R. Riessen, B. Poidinger, K. Dey, N. Weiler, A. Meier-Hellmann, H. H. Haberle, G. Wobker, U. X. Kaisers, and K. Reinhart. Effect of a Multifaceted Educational Intervention for Anti-Infectious Measures on Sepsis Mortality: a Cluster Randomized Trial. *Intensive Care Med*, 43(11):1602–1612, 2017. [p20]

C. Brown. *Dummies: Create Dummy/Indicator Variables Flexibly and Efficiently*, 2012. URL http://CRAN.R-project.org/package=dummies. R package version 1.5.6. [p8]

J. R. Carpenter and M. G. Kenward. *Multiple Imputation and Its Application*. John Wiley & Sons, 2013. ISBN: 978-0-470-74052-1. [p1, 2, 3, 4, 8, 14, 21]

J. R. Carpenter, H. Goldstein, and M. G. Kenward. Realcom-impute software for multilevel multiple imputation with mixed response types. *Journal of Statistical Software.*, 45(5):1–14, 2011. [p1]

G. Casella and R. L. Berger. *Statistical Inference*. Duxbury Press, 2nd edition, 2002. [p20]

J. Drechsler. Multiple imputation of multilevel missing data—rigor versus simplicity. *Journal of Educational and Behavioral Statistics*, 40(1):69–95, 2015. URL https://doi.org/10.3102/1076998614563393. [p21]

C. K. Enders, T. Hayes, and H. Du. A comparison of multilevel imputation schemes for random coefficient models: Fully conditional specification and joint model imputation with random covariance matrices. *Multivariate Behavioral Research*, 53(5):695–713, 2018. URL https://doi.org/10.1080/00273171.2018.1477040. PMID: 30693802. [p21]

H. Goldstein, J. R. Carpenter, M. G. Kenward, and K. A. Levin. Multilevel models with multivariate mixed response types. *Statistical Modelling.*, 9(3):173–197, 2009. DOI: 10.1177/1471082X0800900301. [p2, 8]

H. Goldstein, J. R. Carpenter, and W. J. Browne. Fitting multilevel multivariate models with missing data in responses and covariates that may include interactions and non-linear terms. *Journal of the Royal Statistical Society A*, 177(2):553–564, 2014. DOI: 10.1111/rssa.12022. [p21]

S. Grund, O. Lüdtke, and A. Robitzsch. Multiple Imputation of Missing Covariate Values in Multilevel Models with Random Slopes: a Cautionary Note. *Behav Res Methods*, 48(2):640–649, 2016a. [p21]

S. Grund, O. Lüdtke, and A. Robitzsch. Multiple imputation of multilevel missing data: An introduction to the r package pan. *SAGE Open*, 6(4):2158244016668220, 2016b. URL https://doi.org/10.1177/2158244016668220. [p4]

S. Grund, A. Robitzsch, and O. Lüdtke. *Mitml: Tools for Multiple Imputation in Multilevel Modeling*, 2016c. URL https://CRAN.R-project.org/package=mitml. R package version 0.3-0. [p18]

S. Grund, O. Lüdtke, and A. Robitzsch. Multiple imputation of missing data at level 2: A comparison of fully conditional and joint modeling in multilevel designs. *Journal of Educational and Behavioral Statistics*, page 1076998617738087, 2018a. URL https://doi.org/10.3102/1076998617738087. [p20]

S. Grund, O. Lüdtke, and A. Robitzsch. Multiple imputation of missing data for multilevel models: Simulations and recommendations. *Organizational Research Methods*, 21(1):111–149, 2018b. URL https://doi.org/10.1177/1094428117703686. [p20]

S. Grund, O. Lüdtke, and A. Robitzsch. Missing data in multilevel research. In S. E. Humphrey and J. M. LeBreton, editors, *Handbook for Multilevel Theory, Measurement, and Analysis*. American Psychological Association, Washington, DC, 2018c. ISBN 978-1-4338-3001-3. [p20]

T. Harding, F. Tusell, and J. L. Schafer. *Cat: Analysis of Categorical-Variable Datasets with Missing Values*, 2012. URL http://CRAN.R-project.org/package=cat. R package version 0.0-6.5. [p1]

A. Hossain, K. Diaz-Ordaz, and J. W. Bartlett. Missing continuous outcomes under covariate dependent missingness in cluster randomised trials. *Statistical Methods in Medical Research*, 26(3):1543–1562, 2017a. URL https://doi.org/10.1177/0962280216648357. PMID: 27177885. [p20]

A. Hossain, K. DiazOrdaz, and J. W. Bartlett. Missing binary outcomes under covariate-dependent missingness in cluster randomised trials. *Statistics in Medicine*, 36(19):3092–3109, 2017b. URL https://doi.org/10.1002/sim.7334. [p20]

O. Lüdtke, A. Robitzsch, and S. Grund. Multiple Imputation of Missing Data in Multilevel Designs: A Comparison of Different Strategies. *Psychol Methods*, 22(1):141–165, 2017. [p10, 21]

F. Meinfelder. *BaBooN: Bayesian Bootstrap Predictive Mean Matching - Multiple and Single Imputation for Discrete Data*, 2011. URL http://CRAN.R-project.org/package=BaBooN. R package version 0.1-6. [p8]

X.-L. Meng. Multiple-imputation inferences with uncongenial sources of input. *Statistical Science*, 9(4): 538–558, 1994. URL http://dx.doi.org/10.1214/ss/1177010269. [p2]

A. A. Novo and J. L. Schafer. *Norm: Analysis of Multivariate Normal Datasets with Missing Values*, 2013. URL http://CRAN.R-project.org/package=norm. R package version 1.0-9.5. [p1]

I. Olkin and R. F. Tate. Multivariate correlation models with mixed discrete and continuous variables. *The Annals of Mathematical Statistics*, 32(2):448–465, 1961. URL http://dx.doi.org/10.1214/aoms/1177705052. [p1]

J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and R Core Team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2017. URL https://CRAN.R-project.org/package=nlme. R package version 3.1-131. [p18]

M. Quartagno and J. R. Carpenter. Multiple imputation for ipd meta-analysis: Allowing for heterogeneity and studies with missing covariates. *Statistics in Medicine*, 35(17):2938–2954, 2016. ISSN 1097-0258. URL https://doi.org/10.1002/sim.6837. sim.6837. [p3, 4, 10, 13, 20, 21]

M. Quartagno and J. R. Carpenter. Multilevel multiple imputation in presence of interactions, nonlinearities and random slopes. In *Studies in Theoretical and Applied Statistics - SIS2018 - 49th Meeting of the Italian Statistical Society, Palermo 20-22 June 2018*. Springer-Verlag, 2018. [p21]

M. Quartagno and J. R. Carpenter. Multiple imputation for discrete data: An evaluation of the joint latent normal model. *Biometrical Journal*, In press, 2019. [p9, 20]

J. Rasbash, F. Steele, W. J. Browne, and H. Goldstein. A user's guide to mlwin, v3.00. *Centre for Multilevel Modelling, University of Bristol.*, 2017. [p4]

D. B. Rubin. Inference and missing data. *Biometrika.*, 63(3):581–592, 1976. DOI:10.1093/biomet/63.3.581. [p1]

D. B. Rubin. *Multiple Imputation for Nonresponse in Surveys.* John Wiley & Sons, New York, 1987. [p1]

J. L. Schafer. *Analysis of Incomplete Multivariate Data.* Chapman & Hall., 1997. ISBN 0-412-04061-1. [p1]

J. L. Schafer. *Mix: Estimation/Multiple Imputation for Mixed Categorical and Continuous Data*, 2010. URL http://CRAN.R-project.org/package=mix. R package version 1.0-8. [p1]

J. L. Schafer and M. K. Olsen. Multiple imputation for multivariate missing-data problems: A data analyst's perspective. *Multivariate Behavioral Research*, 33, 2000. [p1]

J. L. Schafer and R. M. Yucel. Computational strategies for multivariate linear mixed-effects models with missing values. *Journal of Computational and Graphical Statistics*, 11(2):437–457, 2002. URL https://doi.org/10.1198/106186002760180608. [p1]

S. R. Seaman and R. H. Keogh. Handling missing data in matched case-control studies using multiple imputation. *Biometrics*, 71(4):1150–1159, 2015. ISSN 1541-0420. URL https://doi.org/10.1111/biom.12358. [p20]

M. A. Tanner and W. H. Wong. The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82(398):528–540, 1987. ISSN 01621459. URL http://www.jstor.org/stable/2289457. [p2]

R. M. Yucel. Random-covariances and mixed-effects models for imputing multivariate multilevel continuous data. *Statistical Modelling.*, 11(4):351–370, 2011. DOI: 10.1177/1471082X100110040. [p1, 3, 10, 13]

Z. Zhang, R. Parker, C. Charlton, G. Leckie, and W. Browne. R2mlwin: A package to run mlwin from within r. *Journal of Statistical Software, Articles*, 72(10):1–43, 2016. ISSN 1548-7660. URL https://doi.org/10.18637/jss.v072.i10. [p4]

J. H. Zhao and J. L. Schafer. *Pan: Multiple Imputation for Multivariate Panel or Clustered Data*. R Foundation for Statistical Computing, 2013. R Package, version 0.9. [p1]

Matteo Quartagno
MRC Clinical Trials Unit at UCL
90 High Holborn, London WC1V 6LC
United Kingdom
m.quartagno@ucl.ac.uk

Simon Grund
IPN - Leibniz Institute for Science and Mathematics Education at Kiel University
Olshausenstraße 62, D-24118 Kiel
Germany
grund@ipn.uni-kiel.de

James Carpenter
London School of Hygiene and Tropical Medicine
Keppel Street, Bloomsbury, London WC1E 7HT
United Kingdom
james.carpenter@lshtm.ac.uk