Software Review

Journal of Educational and Behavioral Statistics
Vol. XX, No. X, pp. 1–17
DOI: 10.3102/1076998619858624
Article reuse guidelines: sagepub.com/journals-permissions
© 2019 AERA. http://jebs.aera.net

Flexible, Free Software for Multilevel Multiple Imputation: A Review of `Blimp` and `jomo`

Timothy Hayes
Florida International University

Multiple imputation is a popular method for addressing data that are presumed to be missing at random. To obtain accurate results, one's imputation model must be congenial to (appropriate for) one's intended analysis model. This article reviews and demonstrates two recent software packages, `Blimp` *and* `jomo`*, to multiply impute data in a manner congenial with three prototypical multilevel modeling analyses: (1) a random intercept model, (2) a random slope model, and (3) a cross-level interaction model. Following these analysis examples, I review and discuss both software packages.*

Keywords: *missing data; multiple imputation; multilevel modeling; statistical software*

Multilevel modeling has become a popular, widely used statistical framework for analyzing clustered data. Whether one's analysis questions involve repeated measurements nested within individuals or students nested within classrooms, special care must be taken to properly treat the dependencies inherent in multilevel data. When some participants fail to attend a longitudinal measurement session or when students from certain classrooms are not present for the administration of a key assessment, missing data threaten to bias analysis results and inflate standard errors at multiple levels of measurement unless proactive measures are taken to remediate these effects. To aid researchers facing such scenarios, this article reviews and demonstrates two free software packages for conducting multiple imputation (Little & Rubin, 2002) of multilevel data.

A key issue in multiple imputation is *congeniality* of the imputation and analysis models (Meng, 1994). Simply stated, unless one's imputation model preserves all of the relevant features of one's analysis model, the predicted values (imputations) generated will ultimately inject bias into the imputed data rather than alleviating it. Failing to take the multilevel structure of a target analysis into account in one's imputation model is tantamount to making predictions based on a fixed effect (single level) regression, assuming zero variation at higher levels of nesting (i.e., zero intraclass correlation; Enders, Mistler, & Keller, 2016). For this reason, specialized imputation routines are needed which are congenial to analyses involving multilevel, nested data structures.

In this article, I review two free software packages for multilevel multiple imputation, providing code to demonstrate how each package generates imputations congenial to three prototypical multilevel analyses. `Blimp` (Enders, Keller, & Levy, 2018) performs multilevel multiple imputation using a fully conditional specification (FCS) imputation approach that fills in missing values on a variable-by-variable basis by employing a series of univariate regressions (Raghunathan, Lepkowski, Van Hoewyk, & Solenberger, 2001; van Buuren, Brand, Groothuis-Oudshoorn, & Rubin, 2006). In contrast, the `jomo` package (Quartagno & Carpenter, 2017) in `R` (R Core Team, 2013) performs multilevel multiple imputation using a joint model approach (Carpenter & Kenward, 2013; Schafer, 1997) that fills in missing values on several variables simultaneously using a multivariate regression model.

Overview and Comparison of `Blimp` and `jomo`

Both `Blimp` and `jomo` have many positive features to recommend them. Both programs provide state-of-the-art functionality for generating imputations congenial to a wide range of common multilevel analyses, and the developers of both programs have made them freely available for Mac, PC, and Linux. Indeed, for these reasons, I routinely recommend both `Blimp` and `jomo` to researchers interested in addressing multilevel missing data. Nonetheless, each option may appeal to different subgroups of researchers, depending on their software preferences, background experience, and analytic needs. In this section, I provide an overview of both software packages, comparing and contrasting key features and suggesting which types of users may benefit from each.

Perhaps the biggest difference between the two packages is that `Blimp` is a stand-alone multiple imputation program that may be freely downloaded at http://www.appliedmissingdata.com/multilevel-imputation.html, whereas `jomo` is a package designed to be installed and run in `R` (R Core Team, 2013). As such, `jomo` may have a somewhat narrower appeal among researchers whose preference is to perform both substantive analyses and missing data analyses in `R`, whereas `Blimp` may garner broader appeal based on its ability to generate multiply imputed data sets ready for analysis and pooling in a variety of several software packages including, but not limited to, `R` (e.g., outputting imputed data sets in either a stacked file format, ready for analysis in SPSS [IBM Corp, 2017] or as a set of separate, individual data files and a list of their file paths ready for analysis in Mplus [Muthén & Muthén, 2017]).

The divide in user-friendliness is only widened by the difference in software documentation currently offered by the two imputation programs. The `jomo` package incorporates a set of standard help files and function documentation that should be familiar to `R` users (https://cran.r-project.org/web/packages/jomo/jomo.pdf) but currently lacks a more organized manual or tutorial (though a *Journal of Statistical Software* submission is currently being written, according to a personal communication from developer Matteo Quartagno). Given some of the specific nuances and idiosyncrasies in `jomo`'s code,[1] the lack of a thorough user's guide or similar reference may prove an obstacle for inexperienced `R` users. By contrast, the `Blimp` User's Guide (available for download at http://www.appliedmissingdata.com/blimpuserguide-5.pdf) is extremely readable and thorough, and the program's commands have a simple, straightforward, intuitive style quite similar in feel to that of Mplus (Muthén & Muthén, 2017).

The remainder of this article is organized as follows. First, I introduce the simulated data used in the subsequent software demonstrations. Next, I demonstrate how to multiply impute data for random intercept, random slope, and cross-level interaction models using `Blimp` before demonstrating these same analyses using `jomo`. Following these demonstrations, I briefly show how to analyze and pool multiply imputed data sets from both packages in `R`. Finally, I conclude with a general discussion comparing the features and strengths of both packages and providing recommendations for which users' needs might best be served by each program.

Simulated Data Used in Examples

The following equations depict three prototypical multilevel analyses, using combined model notation from Raudenbush and Bryk (2002):

$$y_{ij} = \gamma_{00} + \gamma_{10}x_{ij} + \gamma_{01}w_j + u_{0j} + \epsilon_{ij}, \tag{1}$$

$$y_{ij} = \gamma_{00} + \gamma_{10}x_{ij} + \gamma_{01}w_j + u_{0j} + u_{1j}x_{ij} + \epsilon_{ij}, \tag{2}$$

$$y_{ij} = \gamma_{00} + \gamma_{10}x_{ij} + \gamma_{01}w_j + \gamma_{11}x_{ij}w_j + u_{0j} + u_{1j}x_{ij} + \epsilon_{ij}, \tag{3}$$

where x_{ij} is a Level 1 predictor for person i in cluster j, w_i is a cluster-level (Level 2) predictor, the γ coefficients indicate fixed effect regression weights, the us indicate Level 2 residuals, and the ϵs indicate person-specific Level 1 residuals. Equation 1 depicts a random intercept model in which each cluster-specific intercept (β_{0j} in Raudenbush and Bryk's Level 1 model notation) consists of a fixed effect (average) intercept, γ_{00}, plus a Level 2 residual, u_{0j}. Equation 2 depicts a model with both a random intercept and random slope. In this model, the cluster-specific slope of x (β_{1j}, in Level 1 model notation) is decomposed into an average fixed effect slope,γ_{10}, and a cluster-specific Level 2 residual, u_{1j}. Finally, Equation 3 depicts a model with both a random slope and a cross-level interaction.

To demonstrate `Blimp` and `jomo` in the context of each prototypical analysis model, I simulated three data sets containing $C = 30$ clusters with five observations per cluster, yielding a total $N = 5 \times 30 = 150$ cases (drawn from population models corresponding to the three analysis models of interest). Table 1 describes the variables in each simulated data set. In brief, each data set contained two Level 1 predictors, (x_1 and x_2), two level-2 predictors (w_1 and w_2), two

TABLE 1.
Description of Simulated Variables

Variables	Role in Analysis
clus	Cluster identifier
x_1	Level 1 substantive model predictor
x_2	Additional Level 1 predictor
w_1	Level 2 substantive model predictor
w_2	Additional Level 2 predictor
y	Outcome variable in each analysis
a_1	Auxiliary variable used to inject missing data on x_1
a_2	Auxiliary variable used to inject missing data on w_1
x_{m_1}	x_1 with missing data
w_{m_1}	w_1 with missing data
y_m	y with missing data

Note. Variables x_1, w_1, and y, as well as corresponding missing data variables x_{m_1}, $w_{m}1$, and y_m were featured in each substantive analysis models of interest, whereas variables x_2, w_2, a_1, and a_2 were used as auxiliary variables in the imputation models.

TABLE 2.
Percent Missing Data on Each Variable in the Simulated Data Examples

	Simulated Data Set, by Analysis Model		
Variables	Random Intercept (%)	Random Slope (%)	Cross-Level Interaction (%)
x_{m_1}	15	16	31
w_{m_1}	17	17	13
$x_{m_1} \times w_{m_1}$	—	—	42
y_m	13	17	—

Note. Percentages vary on the same variable because each analysis example utilized a different simulated data set.

auxiliary variables (cf. Collins, Schafer, & Kam, 2001) used to inject missing data (a_1: an auxiliary variable used to inject missing data on the Level 1 model variables, and a_2: an auxiliary variable used to inject missing data on Level 2 model variables), and a cluster-level identifier variable (*clus*). Additionally, to facilitate comparison, each data set contained both complete data and missing data versions of key model variables. Table 2 shows the percentage of missing data on each key model variable in each simulated example data set. All three simulated data sets, along with the corresponding `Blimp` and `R` scripts, can be found in the online supplemental materials accompanying this article.

In the following two sections, I first demonstrate how to impute each analysis model using `Blimp`. I then demonstrate how to impute each model using `jomo`.

Readers familiar with missing data analysis will recall that multiple imputation proceeds in three phases: (1) an *imputation phase*, in which the imputation model is specified and multiple copies of the data set are filled in by predicted values from the model; (2) an *analysis phase*, in which the target statistical analysis of interest is performed on all imputed data sets; and (3) a *pooling phase*, in which parameter estimates and standard errors from the analyses are aggregated across the imputed data sets using specialized formulas (Little & Rubin, 2002). Because both `Blimp` and `jomo` are programs for generating multiply imputed data sets to be analyzed and pooled using other software, the initial demonstrations of each package focus exclusively on correctly specifying and executing the imputation model in order to generate imputations congenial to the multilevel models of Equations 1 through 3. Although I assume readers have some familiarity with analyzing and pooling multiply imputed data using their software package of choice, I later provide a brief demonstration of how to analyze and pool the multiply imputed multilevel data sets using the `mitml` package in `R` (Grund, Robitzsch, & Luedtke, 2017).

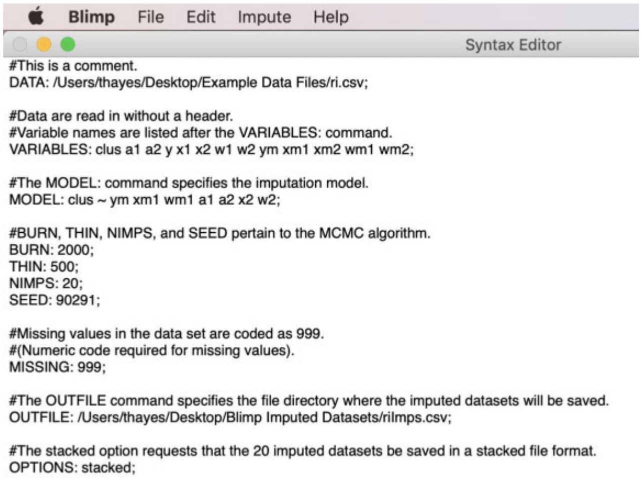
Conducting Multiple Imputation Using `Blimp`

To multiply impute a data set using `Blimp`, one must first write a `Blimp` script, save it with a .imp file extension, and then run the script in order to generate the imputations. Although `Blimp` may conveniently be used from the terminal command line, for the majority users these steps can most easily be accomplished via `Blimp`'s graphical user interface (GUI). Upon opening `Blimp`, a blank syntax editor window automatically appears. To introduce the `Blimp` interface and its syntax conventions, Figure 1 displays a screenshot of a `Blimp` syntax file, with syntax corresponding to the random intercept model described in the previous section. Like `R`, comments in `Blimp` are preceded by the number (pound) sign, #. In the example syntax in Figure 1, comments throughout the script explain the main syntax conventions. For a comprehensive discussion of `Blimp`'s syntax, consult the User's Guide. In the discussion that follows, I focus primarily on the specification of the imputation model using the `MODEL:` command, drawing only limited attention to other commands.

The `MODEL:` command specifies the imputation model. For multilevel imputation, the model syntax takes the form

```
cluster identifier(s) ~ imputation model variables;
```

For single-level imputation, one would simply omit the cluster identifier(s) on the left-hand side of the tilde (~). `Blimp` automatically detects the level at which each variable is measured as well as whether the variable contains missing data. In the present example, `ym`, `xm1`, and `wm1` contain missing data. Variables `x2`, `w2`, `a1`, and `a2` are included as additional predictors in the imputation model.



```
#This is a comment.
DATA: /Users/thayes/Desktop/Example Data Files/ri.csv;

#Data are read in without a header.
#Variable names are listed after the VARIABLES: command.
VARIABLES: clus a1 a2 y x1 x2 w1 w2 ym xm1 xm2 wm1 wm2;

#The MODEL: command specifies the imputation model.
MODEL: clus ~ ym xm1 wm1 a1 a2 x2 w2;

#BURN, THIN, NIMPS, and SEED pertain to the MCMC algorithm.
BURN: 2000;
THIN: 500;
NIMPS: 20;
SEED: 90291;

#Missing values in the data set are coded as 999.
#(Numeric code required for missing values).
MISSING: 999;

#The OUTFILE command specifies the file directory where the imputed datasets will be saved.
OUTFILE: /Users/thayes/Desktop/Blimp Imputed Datasets/riImps.csv;

#The stacked option requests that the 20 imputed datasets be saved in a stacked file format.
OPTIONS: stacked;
```

FIGURE 1. `Blimp` *syntax editor, displayed on Mac, with syntax for random intercept analysis.*

The `NIMPS:` command indicates the number of imputed data sets to save (20, in the present example). The `OPTIONS: stacked` command indicates that the imputations are to be saved in a single file, stacked vertically (each data set saved on top of each other in the CSV spreadsheet). This format can be conveniently read into `R` (or SPSS) and subsequently analyzed and pooled (e.g., using the `mitml` package, as demonstrated in below). Conveniently, the `stacked` key word can be replaced with the `separate` key word, which saves each imputed data set as a separate file, along with a file list indicating the file paths to each imputed data file. This output format is ideal for analyzing and pooling the imputed data sets in Mplus (Muthén & Muthén, 2017), which requires a file list as input, specifying directory paths to separately saved imputed data files. Details of the remaining commands can be found in the `Blimp` User's Guide.

The .imp script above can easily be run from the `Blimp` GUI by clicking Run under the `Impute` drop-down menu. Importantly, after the imputed data sets are successfully saved, `Blimp` prints the following message, indicating the order of the variables in the imputed data file:

```
VARIABLE ORDER IN SAVED DATA:
imp# clus a1 a2 y x1 x2 w1 w2 ym xm1 xm2 wm1 wm2
```

Since `Blimp` outputs files without column names (with no header), this information will be crucial to have on hand when reading the imputed data file into `R` or another statistical package (see online supplemental files for comprehensive syntax).

For multilevel analyses with random intercepts but no random slopes or cross-level interactions, the syntax displayed in Figure 1 is sufficient. This model syntax invokes an FCS imputation algorithm that fills in the missing value on each variable on the right-hand side of the tilde in the model command one at a time via a series of univariate regressions (Enders et al., 2016, 2018; van Buuren, 2007; van Buuren et al., 2006).

In the absence of random coefficients or cross-level interactions, all variables may be imputed in the same manner regardless of their roles as predictors or outcomes in the eventual substantive analysis of interest. When a target analysis features a random slope or cross-level interaction, however, the imputation procedure must be amended to account for the distributional complexity implied by the model. As an example, take the random slope model from Equation (2). When data are missing on both the outcome, y_m, and the predictor with a random slope, x_{m_1}, standard FCS imputation assuming normally distributed Level 1 and Level 2 residuals will produce biased results (Enders, Du, & Keller, 2017, 2019; Enders, Hayes, & Du, 2019).

Although the computational details are beyond the scope of this software review, the solution to this problem is to switch from standard FCS imputation to an extension called substantive model compatible FCS imputation (SMC-FSC imputation; cf. Bartlett, Seaman, White, & Carpenter, 2015, also termed fully Bayesian imputation by Enders, Du, et al., 2019). The basic conceit of SMC-FCS is to selectively choose imputations that have a high joint likelihood of being produced by both (a) the substantive analysis model predicting the outcome, y_m, and (b) a multilevel regression model in which all other predictor variables (but not the outcome, y_m) are used to impute x_{m_1}. For this reason, whereas traditional FCS imputation treats each variable to be imputed as the outcome in its own (multilevel) regression with no particular need to treat y_m any differently than any other variable to be imputed, the joint likelihood used by SMC-FCS imputation requires the user to distinguish between the outcome variable and the predictors in the eventual substantive analysis model of interest.

The details of the SMC-FCS algorithm may be technical, but implementing it in Blimp is straightforward. This is accomplished by adding two new pieces to the syntax above. First, one must designate which variable serves as the substantive model outcome using the `OUTCOME:` command. Second, one may designate a random slope between a predictor and outcome by inserting a colon (:) between them. For the random slope model from Equation 2, the `MODEL:` and `OUTCOME:` syntax are

```
MODEL: clus ~ ym:xm1 wm1 a1 a2 x2 w2;
OUTCOME: ym;
```

where the OUTCOME: command specifies `ym` as the substantive model outcome, and the notation `ym:xm1` specifies a random slope between y_m and x_{m_1}.

Note that a random slope can be construed as a form of interaction in which the *x*-*y* slope is moderated by cluster. Viewed in this way, it comes as little surprise that other types of interaction models, including the cross-level interaction model of Equation 3, suffer from similar distributional issues. In `Blimp`, the same SMC-FCS methods that make random slope estimation possible can also be used to impute interactions and polynomial terms. Once again, the outcome variable in one's substantive model must be specified using the `OUTCOME:` command. To specify an interaction between two variables, their product may be indicated with an asterisk (*) between them on the `MODEL:` line. For the cross-level interaction of Equation 3, the `MODEL:` and `OUTCOME:` syntax are as follows:

```
MODEL: clus ~ y:xm1 xm1*wm1 a1 a2 x2 w2;
OUTCOME: y;
```

Note that, once again, the outcome variable, `y`, is indicated using the OUTCOME: command. Additionally, the asterisk (*) is used to indicate the cross-level interaction between variables `xm1` (Level 1) and `wm1` (Level 2). This model also specifies a random slope between `y` and `xm1`, `y:xm1`.

Conducting Multiple Imputation Using `jomo`

In the `jomo` package, the `jomo()` and `jomo.smc()` functions are the powerhouses for conducting multiple imputation. The `jomo()` function conducts multilevel, joint model multiple imputation appropriate for random intercept models that do not contain random slopes or cross-level interactions, whereas the `jomo.smc()` function conducts multilevel, joint model multiple imputation appropriate for more complex substantive models including random coefficients and/or cross-level interactions. Although the syntax required differs between the two functions, they share at least two common conventions: (1) In both functions, users must distinguish between Level 1 and Level 2 variables in the multilevel imputation model; and (2) in both functions, users must explicitly indicate when regression intercepts are to be included in the imputation model. In this section, I first demonstrate how to use the `jomo()` function to multiply impute data for a random intercept model and then demonstrate how to use the `jomo.smc()` function to multiply impute data for the random slope and cross-level interaction models.

Recall the random intercept model from Equation 1. To multiply impute data for a random intercept model using `jomo`, one must first load the `jomo` package (e.g., by running `library(jomo)`) and then proceed in two steps: (1) first add a vector of 1s to the data frame to be imputed if one wishes to include intercept parameters in the imputation model and then, finally, (2) conduct the imputation using the `jomo()` function:

```
#Create a vector of 1s to model the intercept.
```

```
#riData is the name of the data frame containing the data for the random intercept model.

riData$icept <- 1 #this adds a column of 1s named icept

#multiply impute the data and save the list of imputed data sets as an object named jomoImps
jomoImps <- jomo(Y = riData[,c("xm1","ym")], Y2 = riData[,"wm1", drop = F], X = riData[,c("icept", "x2", "a1")], X2 = riData[,c("icept","a2", "w2")], clus = riData$clus, nimp = 20)
```

Readers considering using `jomo` are advised to take special note that the first line of uncommented code creates a variable named `icept` set equal to a vector of 1s. Although easy to miss, if one examines the `jomo()` function help file (e.g., by typing `?jomo` into the `R` console) the documentation clearly describes that a vector of 1s is necessary if one wishes to include intercepts in the imputation model. Since one nearly always *does* want to include, rather than omit, intercept parameters, this is crucial code to be included in any `jomo` analysis.

After adding the `icept` variable to the `riData` data frame, the next step is to multiply impute the data. The `jomo()` function is a wrapper that encompasses the functionality of `jomo`'s main suite of imputation functions. The first two arguments, `Y` and `Y2`, take as input data frames containing the Level 1 and the Level 2 variables to be imputed, respectively. The third and fourth arguments, `X` and `X2`, take as input data frames containing Level 1 predictor variables with complete data and Level 2 predictor variables with complete data, respectively, to be included as predictors in the imputation model. Note that the `icept` variable is specified as a predictor at both Level 1 (in the `X` argument) and Level 2 (in the `X2` argument) in the `jomo()` function, indicating that intercepts should be modeled at both levels. The argument `clus` specifies the variable(s) that indicate clusters in the multilevel data frame. Finally, the argument `nimp` specifies the number of imputations to be saved (here, `nimp = 20` specifies 20 imputed data sets).

The `jomo()` function outputs a data frame object containing the imputed data sets. The data are saved in stacked format, with the `$Imputation` variable indicating which observations in the data frame correspond to which imputed data set. For example, if one has imputed 20 data sets, the unique values of the `$Imputation` variable will range from 0 to 20, with 0 indicating rows corresponding to the original data set with missing values, 1 indicating rows corresponding to the first imputed data set, and so on.

The code above is sufficient for imputing random intercept models, but what about random slopes and cross-level interactions? Recall that joint model imputation methods fill in missing observations on multiple variables simultaneously using a multivariate regression approach. A joint imputation model can be uncongenial to an analysis model if the assumed multivariate distribution (e.g., multivariate normal) does not match the true joint distribution of variables implied by the analysis as is the case when there are random coefficients or cross-level interactions.

To address this issue, `jomo`, like `Blimp`, has recently added functions allowing users to conduct SMC Joint Model (SMC-JM) imputation (cf. Goldstein, Carpenter, & Browne, 2014). Like the SMC-FCS method available in `Blimp`, SMC-JM involves implementing an additional Markov-Chain Monte Carlo (MCMC) step and choosing imputations with high joint likelihoods of being compatible with both the substantive model for the outcome, *y* (including the random slope term), and a model involving the predictors. A primary difference is that the `jomo` version imputes the predictors simultaneously using joint model imputation methods.

To use the `jomo.smc()` function, one must provide five primary pieces of information: (1) a formula for the substantive regression of interest, explicitly including the intercept if desired; (2) a data frame including both (a) the variables indicated in the model formula and (b) any auxiliary variables to be included in the imputation model; (3) a vector of 1s and 2s, designating which columns of the data frame are Level 1 and Level 2 variables, respectively; (4) the number of imputed data sets to be saved; and (5) the name of the function to be used to analyze the substantive model (e.g., "`lm`," "`lmer`," "`glm`," "`glmer`").

The code for SMC-JM imputation of the random slope model in jomo appears below:

```
#Substantive Model Compatible (SMC) Method
jomoImpsSMC <- jomo.smc(ym ~ xm1 + wm1 + (1 + xm1|clus),
data = rsData[,c("clus","a1", "a2", "x2", "w2", "xm1",
"wm1","ym")], level = c(2, 1, 2, 1, 2, 1, 2, 1), nimp = 20,
model = "lmer")
```

Here, the model being used is the `lmer()` function from the `lme4` package for linear mixed-effects modeling in `R` (Bates, Mächler, Bolker, & Walker, 2014). The first part of formula for the substantive model indicates that the outcome, `y`, is to be regressed on variables `xm1` and `wm1`. The final term, `(1 + xm1|clus)`, depicts the random effects. The `1` explicitly indicates that the random intercept is to be modeled. The term `xm1|clus` indicates that `xm1` will have a random slope that varies by cluster. The data argument provides the data frame object that includes all variables to be included in the imputation model (model variables and additional auxiliary variables). As is often the case, not all variables in the data set are to be included in the imputation model, so the code subsets the `rsData` data frame (short for "random slope data set") to include only the variables of interest. The `level` argument specifies which columns of this subsetted data frame are Level 1 versus Level 2 variables. Finally, the `nimp` argument specifies that 20 imputed data sets are to be saved.

This same function can be used to impute substantive models that include cross-level interactions. The code for using SMC-JM to multiply impute a data set in a manner congenial to a substantive model featuring a random slope and cross-level interaction is:

```
jomoImpsSMC <- jomo.smc(y ~ xm1 + wm1 + xm1*wm1 + (1 +
xm1|clus), data = intData[,c("clus","a1", "a2", "x2",
"w2", "xm1", "wm1","y")], level = c(2, 1, 2, 1, 2, 1, 2, 1),
nimp = 20, model = "lmer")
```

The primary difference between this code and the previous example is the inclusion of the term `xm1*wm1` in the model formula, indicating the presence of a cross-level interaction between Level 1 variable `xm1` and Level 2 variable `wm1`.

Analyzing and Pooling Imputed Data Sets Using `mitml`

The `mitml` package (*m*ultiple *i*mputation *t*ools for *m*ulti*l*evel data; Grund et al., 2017) in `R` provides a suite of functions for analyzing and pooling the results of multilevel, multiply imputed data sets, making it an extremely useful adjunct to both `jomo` and `Blimp`. In both cases, the structure of the `mitml` procedure follows the same basic steps. First, the analyst uses the `split()` function to split the imputed, stacked data frame into a list object containing one imputed data set per list element. Next, the resulting list is converted to an `mitml` list (a list object with an `mitml.list` class designation that can be detected and uniquely acted upon by other functions) using the `as.mitml.list()` function. The example code below accomplishes these steps simultaneously, demonstrated on the `Blimp` imputations (syntax for `jomo` proceeds identically, as documented in the online supplemental `R` scripts):

```
#Convert to mitml list.
blimplist <- as.mitml.list(split(blimpImps, blimpImps$
Imputation))

#Analyze imputed datasets using lmer() function
blimpmodel <- with(blimplist, lmer(ym ~ xm1 + wm1 +
(1|clus)))

#Pool and display results.
blimpFit <-testEstimates(blimpmodel, var.comp = T)
blimpFit #display results.
```

The next step involves using the `with()` function in conjunction with each imputation list to perform the analysis of interest (here a random intercept analysis using `lmer` in the `lme4` package) on each imputed data set (i.e., on the each data frame saved in each element of the mitml list). Finally, the estimates and

TABLE 3.
*Coefficients and Standard Errors (*SE*) by Analysis and Missing Data Handling Method*

	Complete		Listwise		`jomo`		`Blimp`	
Parameter	Estimate	*SE*	Estimate	*SE*	Estimate	*SE*	Estimate	*SE*
Random intercept model								
Intercept	5.12	.14	5.10	.15	5.22	.15	5.24	.16
x_1 slope	0.22	.08	0.21	.11	0.30	.10	0.32	.11
w_1 slope	0.38	.13	0.40	.14	0.31	.15	0.31	.15
Intercept variance	0.41	—	0.32	—	0.41	—	0.43	—
Residual variance	0.89	—	0.91	—	0.98	—	0.97	—
Random slope model								
Intercept	5.02	.11	5.08	.16	4.98	.14	5.01	.14
x_1 slope	0.28	.09	0.33	.11	0.29	.10	0.31	.10
w_1 slope	0.06	.08	0.09	.11	0.04	.11	0.02	.12
Intercept variance	0.19	—	0.30	—	0.35	—	0.29	—
Slope variance	0.08	—	0.08	—	0.10	—	0.07	—
Covariance	0.12	—	0.16	—	0.13	—	0.12	—
Residual variance	0.91	—	0.91	—	0.81	—	0.94	—
Cross-level interaction model								
Intercept	5.10	.14	5.16	.17	5.05	.14	5.06	.14
x_1 slope	0.45	.12	0.59	.16	0.59	.12	0.56	.14
w_1 slope	−0.30	.15	−0.51	.19	−0.38	.16	−0.38	.17
$x_1 \times w_1$ slope	0.31	.11	0.24	.18	0.32	.13	0.33	.14
Intercept variance	0.23	—	0.22	—	0.23	—	0.25	—
Slope variance	0.17	—	0.29	—	0.20	—	0.25	—
Covariance	0.07	—	0.13	—	0.11	—	0.10	—
Residual variance	1.18	—	1.07	—	1.06	—	1.02	—

Note. For all multiple imputation analyses, the analysis and pooling phases were conducted using the `mitml` package in conjunction with `lme4`.

standard errors from each of the 20 imputed data sets are pooled using the `tes-tEstimates()` function.

Table 3 presents coefficients and standard errors from `jomo` and `Blimp` for all three analysis models, computed using this exact procedure in `mitml`, contrasted with analyses of each complete data set (with no missing data), and an analysis using listwise deletion. Of particular note is the superior performance of `jomo` and `Blimp` over listwise deletion in recovering the random intercept variance in the first analysis, the random slope variance in the second analysis, and the cross-level interaction term in the final analysis.

General Discussion, Review, and Comparison of the Two Software Packages

The utility of a multilevel multiple imputation software package rests on its ability to generate imputations that are congenial to the features of one's substantive analysis of interest. Both `Blimp` and `jomo` perform well at imputing data for use in all three prototypical analysis models considered here. Additionally, both `Blimp` and `jomo` provide facilities for imputing Level 2 predictor variables (w_{m_1} in all example analyses) as well as categorical variables (though not explicitly demonstrated here).

It is worth noting that although each analysis example in this article uses a single, potentially idiosyncratic simulated data set, the pattern of results demonstrated here generally mirror those of recent simulation studies comparing these two imputation approaches using these exact software packages (Enders, Du, et al., 2019; Enders, Hayes, et al., 2019). Because SMC-JM methods have only recently been added in `jomo`, direct comparisons of these approaches to Blimp's SMC-FCS algorithm via simulation have not yet been conducted. The simulated examples presented in this article suggest that substantive model compatible imputation is broadly helpful in accurately recovering model parameter estimates, regardless of whether the SMC imputation is conducted via the FCS or joint model framework. Conscientious users are advised to keep up-to-date on the simulation work in this area, however, as future studies may clarify the conditions under which each of these SMC imputation frameworks might be preferable.

Although `Blimp` and `jomo` incorporate many of the same features, there are some key differences in implementation worth mentioning. First, given the inherent complexity of multilevel data structures, it is crucial to take proactive steps to inspect the MCMC algorithm underlying an imputation model for proper convergence. In `Blimp`, users may assess convergence by requesting the potential scale reduction factor (Gelman et al., 2014; Gelman & Rubin, 1992) on an initial run of a given imputation model, examining the number of iterations required for the statistic to drop below a cutoff value of 1.05. Alternatively, in `jomo`, users may assess convergence by creating and visually inspecting trace plots (line graphs) of the parameter estimates at each iteration of the MCMC chain (Schafer & Olsen, 1998), examining how many iterations are required to obtain adequate, representative coverage of the distribution of the parameter. Here again, the commands required by `jomo` to extract the requisite parameter estimates and construct the trace plots should prove quite easy for researchers who are both knowledgeable of multiple imputation methodology and experienced in using `R` but may prove more difficult for researchers with less experience in either area.

Second, although both `Blimp` and `jomo` are capable of imputing categorical data, they differ in the procedures used for binary and ordinal outcomes. Whereas `Blimp` uses a threshold-based latent probit approach, `jomo` imputes all categorical outcomes using a more richly parameterized nominal probit model (cf. Asparouhov & Muthén, 2010a, 2010b; Carpenter & Kenward, 2013; Enders, Hayes, et al., 2019; Wu, Jia, & Enders, 2015). Although either of these approaches should produce accurate estimates, it is possible that the more complex nominal imputation model may produce more identification and convergence problems than the more parsimonious ordinal imputation approach. Finally, it is worth mentioning that `Blimp` features a variety of other useful options including built-in facilities for conducting Monte Carlo simulation studies and an option for generating group-specific imputations to address missing data in multiple group Structural Equation Models.

Despite these differences, however, there are important areas in which the two packages are comparable. Both `Blimp` and `jomo` are capable of accurately imputing multilevel models featuring random intercepts, random slopes, and cross-level interactions. Both packages are kept up-to-date, with new features regularly added reflecting the latest advances in multiple imputation methodology. Additionally, the developers of both packages (Craig Enders for `Blimp` and Matteo Quartagno for `jomo`) have been consistently gracious and responsive to e-mail inquiries requesting clarification of theoretical imputation details, algorithm implementation, and general technical support during the writing of this review, inspiring confidence that users of either program will be able to obtain answers to technical questions that may arise.

As the foregoing discussion implies, the FCS and Joint Model imputation methods utilized by `Blimp` and `jomo`, respectively, represent the two major approaches to multiple imputation advocated in the literature. Currently, these approaches are generally treated by users as interchangeable, but it is possible that future research will shed light on the circumstances in which each method should be preferred over the other. For this reason, it is useful for researchers to have both options in their arsenal for addressing missing data.

At the present time, `Blimp` should appeal to researchers with a preference for FCS imputation and those who wish to analyze and pool their imputations in any of several software packages including `R`, SPSS, and Mplus. By contrast, `jomo` should appeal to researchers with a preference for joint model imputation and those who prefer to work exclusively in `R`, without the need to export data to a separate stand-alone package and then reimport the imputed data sets (although it should be noted that this procedure is extremely straightforward in `Blimp`).

To summarize, both `Blimp` and `jomo` provide state-of-the-art facilities for generating imputations that are germane to any of the three prototypical analysis models described above (random intercept, random slope, and cross-level interaction). I highly recommend both of these programs to researchers hoping to address missing data in the context of their own multilevel analyses. It is my hope that the discussion and examples provided in this review will help point substantive researchers toward the multiple imputation program most compatible with their software preferences and analytic needs.

Acknowledgments

I sincerely thank Craig Enders and Matteo Quartagno for their responsiveness over e-mail and for their general enthusiasm in answering detailed questions about `Blimp` and `jomo`, respectively.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iD

Timothy Hayes https://orcid.org/0000-0001-7530-0241

Note

1. For example, the requirement of explicitly adding a vector of 1s to predictor matrices or data frames in order to model, rather than omit, intercept parameters, as later demonstrated.

References

Asparouhov, T., & Muthén, B. (2010a). *Bayesian analysis using Mplus*. Retrieved from http://www.statmodel.com/download/Bayes3.pdf

Asparouhov, T., & Muthén, B. (2010b). *Multiple imputation with Mplus*. Retrieved from https://www.statmodel.com/download/Imputations7.pdf

Bartlett, J. W., Seaman, S. R., White, I. R., & Carpenter, J. R. (2015). Multiple imputation of covariates by substantive-model compatible fully conditional specification. *Statistical Methods in Medical Research*, *24*, 462–487. doi:10.1177/0962280214521348

Bates, D., Mächler, M., Bolker, B., & Walker, S. (2014). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, *67*, 1–48. doi:10.18637/jss.v067.i01

Carpenter, J. R., & Kenward, M. G. (2013). *Multiple imputation and its application*. West Sussex, England: Wiley. doi:10.1002/9781119942283

Collins, L. M., Schafer, J. L., & Kam, C. M. (2001). A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychological Methods*, *6*, 330–351. doi:10.1037/1082-989X.6.4.330

Enders, C. K., Du, H., & Keller, B. T. (2017). *A fully Bayesian imputation procedure for random coefficient models (and other pesky product terms)*. Paper presented at the Society for Multivariate Experimental Psychology, Minneapolis, MN.

Enders, C. K., Du, H., & Keller, B. T. (2019). A model-based imputation procedure for multilevel regression models with random coefficients, interaction effects, and nonlinear terms. *Manuscript under Revision*.

Enders, C. K., Hayes, T., & Du, H. (2019). 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*, 695–713. doi:10.1080/00273171.2018.1477040

Enders, C. K., Keller, B. T., & Levy, R. (2018). A fully conditional specification approach to multilevel imputation of categorical and continuous variables. *Psychological Methods*, *23*, 298–317. doi:10.1037/met0000148

Enders, C. K., Mistler, S. A., & Keller, B. T. (2016). Multilevel multiple imputation: A review and evaluation of joint modeling and chained equations imputation. *Psychological Methods*, *21*, 222–240. doi:10.1037/met0000063

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2014). *Bayesian data analysis* (3rd ed.). Boca Raton, FL: CRC Press.

Gelman, A., & Rubin, D. B. (1992). *Inference from iterative simulation using multiple sequences. Source: Statistical science* (*Vol. 7*). Retrieved from https://doi.org/https://www.jstor.org/stable/2246093

Goldstein, H., Carpenter, J. R., & Browne, W. J. (2014). 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. Series A.*, *177*, 553–564. doi:10.1111/rssa.12022

Grund, S., Robitzsch, A., & Luedtke, O. (2017). *mitml: Tools for multiple imputation in multilevel modeling*. Retrieved from https://cran.r-project.org/package=mitml

IBM Corp. (2017). *IBM SPSS Statistics for Macintosh* (Version 25.0). Armonk, NY: Author.

Little, R. J. A., & Rubin, D. B. (2002). *Statistical analysis with missing data*. Hoboken, NJ: Wiley. doi:10.1002/9781119013563

Meng, X. (1994). Multiple-imputation inferences with uncongenial sources of input. *Statistical Science*, *9*, 538–558.

Muthén, L. K., & Muthén, B. (2017). *Mplus user's guide* (8th ed.). Los Angeles, CA: Author.

Quartagno, M., & Carpenter, J. (2017). *jomo: A package for multilevel joint modelling multiple imputation*. Retrieved from https://cran.r-project.org/package=jomo

R Core Team. (2013). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from http://r-project.org/

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*, 85–95.

Raudenbush, S. W., & Bryk, A. S. (2002). *Hierarchical linear models* (2nd ed.). Thousand Oaks, CA: Sage.

Schafer, J. L. (1997). *Analysis of incomplete multivariate data*. New York, NY: Chapman & Hall.

Schafer, J. L., & Olsen, M. K. (1998). Multiple imputation for multivariate missing-data problems: A data analyst's perspective. *Multivariate Behavioral Research*, *33*, 545–571. doi:10.1207/s15327906mbr3304_5

van Buuren, S. (2007). Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical Methods in Medical Research*, *16*, 219–242. doi:10.1177/0962280206074463 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*, 1049–1064. doi:10.1080/10629360600810434

Wu, W., Jia, F., & Enders, C. (2015). A comparison of imputation strategies for ordinal missing data on likert scale variables. *Multivariate Behavioral Research*, *50*, 484–503. doi:10.1080/00273171.2015.1022644

Author

TIMOTHY HAYES is an assistant professor of quantitative psychology at Florida International University, 11200 SW 8th Street, DM Room 256, Miami, FL 33199, USA; email: thayes@fiu.edu. His research interests broadly concern missing data, structural equation models, multilevel models, and the intersection of the three. One line of research focuses on evaluating and proposing novel methods for addressing missing data in single-level and multilevel contexts. Another more recent line of work focuses on extending factor score regression approaches to connected measurement models and using factor score residuals to diagnose correlated uniquenesses.

Manuscript received February 8, 2018
Revision received March 14, 2019
Accepted May 29, 2019