Blimp Software Manual
(Version Beta 6.8)

Brian T. Keller
bkeller2@g.ucla.edu
Craig K. Enders
cenders@psych.ucla.edu

March 2017

Developed by Craig K. Enders and Brian T. Keller.

Blimp was developed with funding from

Institute of Educational Sciences award R305D150056.

Craig K. Enders, P.I. Email: cenders@psych.ucla.edu.

Programming by Brian T. Keller. Email: bkeller2@ucla.edu

GUI programming by Nitish Mehta and Brian T. Keller

The citation for this documentation is:

Keller, B. T., & Enders, C. K. (2017). Blimp Software Manual (Version Beta 6.8). Los Angeles, CA.

Contents

1 Introduction

Blimp is a software program designed to perform multiple imputation via fully conditional specification (FCS). The software can perform single-level and multilevel imputation (up to three levels) with mixtures of continuous (normal), ordinal, and nominal variables. In the context of multilevel imputation, Blimp can accommodate analysis models with (a) random intercepts and random slopes, (b) different within- and between-cluster covariance structures (e.g., contextual effects analyses, multilevel structural equation models), and (c) heterogeneous within-cluster variance structures (e.g., daily diary data). Technical details of the Blimp FCS algorithm are available in the following manuscript, which is available upon request.

Enders, C. K., Keller, B. T., & Levy, R. (2016). A chained equations imputation approach for multilevel data with categorical and continuous variables. Manuscript submitted for publication.

1.1 DISCLAIMER

This is beta software with no expressed license given. There is no right to distribute the software or documentation. This is for your sole use only. As beta software, this is given as is. Any published work derived from the use of this software must cite the technical work above and this document. The citation for the documentation is:

Keller, B. T., & Enders, C. K. (2017). Blimp Software Manual (Version Beta 6.8).

Los Angeles, CA.

2 Quick Start Guide

This chapter provides a brief tutorial to help users quickly begin using the Blimp command language. Separate guides for the graphical user interface (GUI) are available at `www.appliedmissingdata.com`. All of the information presented in this section is repeated in more detail in the subsequent chapters.

Example 2.1 depicts a prototypical Blimp syntax file for multilevel imputation. Single-level imputation follows a nearly identical structure.

Example 2.1: Quick Start Example

```
# This is a comment

DATA: /Users/name/path/to/file/mydata.dat;

VARIABLES: id av y x1 x2
          x3 w1 w2 w3;

ORDINAL: x2 w2;

NOMINAL: x3 w3;

MODEL: id ~ av y:x1 y:x2 x3 w1 w2 w3;

BURN: 2000;

THIN: 500;

NIMPS: 10;

MISSING: -99;

SEED: 84398;

OUTFILE: /Users/name/Desktop/myimps*.dat;

OPTIONS: hov clmean separate dat;
```

2.1 Explanation of Syntax in Example 2.1

Conventions

Blimp commands can be entered in the syntax editor of the GUI, or in a plain text file (raw ASCII format) with a '.imp' extension. The Blimps syntax uses the following conventions: (a) the program is not case sensitive, (b) command names (e.g., `DATA`, `VARIABLES`, etc.) are followed by a colon (`:`), (c) each command is terminated with a semicolon (`;`), (d) commands can span multiple lines (e.g., the `VARIABLES` command in Example 2.1), and (e) the number sign (`#`) is used to create a comment line that Blimp ignores.

Reading Data

The `DATA`, `VARIABLES`, `ORDINAL`, `NOMINAL`, and `MISSING` commands specify features of the raw data file. The `DATA` command specifies the full file path to the input data set. When the file path includes spaces, the file path should not be enclosed quotations (the file path in Example 2.1 is consistent with macOS). The input data set must be saved as a CSV (comma separated values) format or a whitespace delimited file (including tab delimited files). The `VARIABLES` command specifies the variables in the input file. The variable names must be alphanumeric with no periods or other special punctuation marks. All missing values must be assigned the same numeric code, the value of which is specified via the `MISSING` command.

Unless otherwise specified, Blimp assumes a normal distribution for all variables. The `ORDINAL` and `NOMINAL` commands are used to specify the metrics of categorical variables. For computational reasons, we recommend specifying binary variables on the ORDINAL line, although the underlying statistical model invoked by the `NOMINAL` command is equivalent. Nominal variables must be represented as a single variable with numeric codes. Complete nominal variables are automatically recoded into a set of dummy codes during imputation. Example 2.1 depicts an imputation model with two ordinal variables (`x2` and `w2`) and two nominal variables (`x3` and `w3`).

`MODEL` Command

The `MODEL` command specifies the variables that are included in the imputation regression models. At a minimum, this list must include all variables and effects of interest in the analysis model(s), but the list may also include additional auxiliary variables. Note that the variable list should include both complete and incomplete variables, and the user need not specify which variables have missing values. Additionally, Blimp will automatically determine the level at which a variable is measured in a multilevel data set.

For single-level imputation, the imputation model variables are listed to the right of the tilde symbol. Multilevel imputation requires a cluster-level identifier variables to the left of the tilde symbol. Two-level imputation requires a single identifier for the level-2 sampling unit (cluster), and three-level imputation requires level-2 and level-3 identifier variables (the order of the identifier variables does not matter). Example 2.1 depicts a two-level imputation problem, where `id` is the level-2 identifier variable.

The FCS algorithm can accommodate multilevel analyses with random slopes. To specify a random slope between two variables, simply place a colon (`:`) between two variables instead of a space (see Section 3.3). In Example 2.1, the imputation model includes random slopes for `y` and `x1` and `y` and `x2`.

Algorithmic Options

The `BURN`, `THIN`, `SEED`, `NIMPS`, and `OPTIONS` commands specify features of the MCMC algorithm. In Example 2.1 the `BURN` command is set to 2000, meaning that 2000 iterations are performed prior to saving the first imputed data set. The `THIN` command specifies the so-called between-imputation interval. In Example 2.1, imputed data sets are saved after every 500^{th} iteration. The `NIMPS` command specifies the number of imputations desired. The `SEED` command specifies the seed value for Blimp's pseudorandom number generator. The `OPTIONS` command specifies various miscellaneous options, such as the format of the imputed data sets (e.g., imputations stacked in a single file or output to separate files), the prior distributions used for variance components, etc. Section 3.5 presents more information about the specific options and their defaults.

Finally, the `OUTFILE` command specifies the complete file path for the resulting imputed data set(s). Example 2.1 saves each imputed data set to a separate file. This specification requires an asterisk in the file name, and this asterisk is subsequently replaced by a numeric value (e.g., myimps1.dat, myimps2.dat, etc.).

2.2 Running Blimp

Blimp is executed via the Run pull-down option from the GUI or via the command-line. A terminal emulator (e.g., Terminal.app in macOS, cmd.exe in Windows) is used to run Blimp from the command-line. To run Blimp via a command-line, type blimp followed by the full file path to the syntax file. For macOS, the command-line tools must be installed first. See chapterch:run for more information. The command-line specification for macOS is as follows:

```
$ blimp ∼/desktop/InputExample.imp
```

where 'InputExample.imp' is a Blimp syntax script saved in the user's Desktop directory. Most terminal applications automatically generate the appropriate file paths if you drag the application and syntax files into the terminal window.

2.3 Explanation of Output from Example 2.1

After executing Blimp (GUI or shell), a header is printed to the output window. This header provides the version number along with contact information. An example header is shown below.

```
                                  Blimp
                                Beta v6.8

                Developed by Craig K. Enders and Brian T. Keller.
                      Blimp was developed with funding from
               Institute of Educational Sciences award R305D150056.
               Craig K. Enders, P.I. Email: cenders@psych.ucla.edu.
             Programming by Brian T. Keller. Email: bkeller2@ucla.edu
              This is Beta software with no expressed license given.


                   DO NOT DISTRIBUTE WITHOUT WRITTEN CONSENT
```

Below the header, Blimp lists the algorithmic options specified in the syntax file. In addition to specified options, the list includes default settings that may not have been explicitly specified in the syntax. For example, in Example 2.1 the keywords `hov` and `clmean` were specified in the `OPTIONS` command, so these options are listed in the output. The syntax did not specify prior distributions for variance components, thus triggering appropriate default settings.

```
Algorithmic Options Specified:
        hov, clmean, Raneff Prior 1, Resvar Prior 1.
```

Following the algorithmic options, Blimp displays the progress of the Gibbs sampler algorithm. Example 2.1 requests 2000 burn-in iterations, and the program issues a message when it has completed 25%, 50%, 75%, and 100% of these initial iterations. Following the burn-in phase, Blimp prints a time and date stamp as it saves each imputed data set.

```
Starting Burn-in on Wed Nov  2 12:06:24 2016
    Burn-in iteration 500 complete on Wed Nov  2 12:06:24 2016
    Burn-in iteration 1000 complete on Wed Nov  2 12:06:24 2016
    Burn-in iteration 1500 complete on Wed Nov  2 12:06:24 2016
    Burn-in iteration 2000 complete on Wed Nov  2 12:06:24 2016
Burn-in complete on Wed Nov  2 12:06:24 2016
    Imputation Saved 1 on Wed Nov  2 12:06:24 2016
    Imputation Saved 2 on Wed Nov  2 12:06:24 2016
    Imputation Saved 3 on Wed Nov  2 12:06:24 2016
    Imputation Saved 4 on Wed Nov  2 12:06:25 2016
    Imputation Saved 5 on Wed Nov  2 12:06:25 2016
    Imputation Saved 6 on Wed Nov  2 12:06:25 2016
    Imputation Saved 7 on Wed Nov  2 12:06:25 2016
    Imputation Saved 8 on Wed Nov  2 12:06:25 2016
    Imputation Saved 9 on Wed Nov  2 12:06:25 2016
    Imputation Saved 10 on Wed Nov  2 12:06:25 2016
```

Finally, after all imputations are saved, Blimp prints the order of the variabls in the inputed data set(s). All variables listed on the `VARIABLES` command are saved to the file(s), regardless of whether the variables are used in imputation.

```
Variable Order: id av y x1 x2 x3 w1 w2 w3
```

Variables are saved in the same order as they appear on the `VARIABLES` command, with one important exception. When imputations are stacked in a single file (the stacked keyword of the `OPTION` command), the first variable in the file is an identifier variable that indexes the imputed data sets.

3 Syntax File

Blimp commands can be entered in the syntax editor of the GUI, or in a plain text file (raw ASCII format) with a '.imp' extension. The main commands used in a standard Blimp input and are listed in Sections 3.2, 3.3, and 3.4.

3.1 General Conventions

The Blimp syntax uses the following conventions: (a) the program is not case sensitive, (b) command names (e.g., `DATA`, `VARIABLES`, etc.) are followed by a colon (`:`), (c) each command is terminated with a semicolon (`;`), (d) commands can span multiple lines (e.g., the `VARIABLES` command in Example 2.1), and (e) the number sign (`#`) is used to create a comment line that Blimp ignores.

3.2 Reading Data

`DATA` The `DATA` command specifies the full file path to the input data set. When the file path includes spaces, the path should not be enclosed quotations. Currently Blimp accepts a comma separated value (CSV) file or a whitespace delimited file (including tab delimited files). Data must be provided in plain text with no header. It is important to note that Blimp currently accepts only numerical values in the data sets.

`VARIABLES` The `VARIABLES` command specifies the variables in the input file. The variable names must be alphanumeric with no periods or other special punctuation marks. The variable list may include variables that are not used in the imputation regression models. All variables listed on the `VARIABLES` line will appear in the imputed data sets returned by Blimp.

`ORDINAL` Unless otherwise specified, Blimp assumes a normal distribution for all variables. The `ORDINAL` command is used to specify incomplete ordinal variables. For computational efficiency, we recommend listing binary variables on the `ORDINAL` line, but these variables could also be treated as nominal.

`NOMINAL` Unless otherwise specified, Blimp assumes a normal distribution for all variables. The `NOMINAL` command is used to specify nominal variables. Nominal variables must be represented as a single variable with numeric codes. Complete nominal variables are automatically recoded into a set of dummy codes during imputation.

`MISSING` The `MISSING` command is used to specify the numeric code that represents a missing value in the input data set. All missing values must be coded with a single numeric value (e.g., -9999).

3.3 Model Command

`MODEL` The `MODEL` command specifies the variables that are included in the imputation regression models. At a minimum, this list must include all variables and effects of interest in the analysis model(s), but the list may also include additional auxiliary variables. Note that the variable list should include both complete and incomplete variables, and the user need not specify which variables have missing values. Additionally, Blimp will automatically determine the level at which a variable is measured in a multilevel data set.

3.3.1 Specifying the Model Statement

The model statement for single-level imputation consists of a tilde symbol followed by a variable list, as follows.

$$\sim \text{ Variables included in imputation}$$

The model statement for multilevel imputation contains two parts: an identifier part and a model part. A model statement takes the following form

$$\text{Identifier variables} \sim \text{Variables included in imputation}$$

Identifier Two-level imputation requires a level-2 identifier variable before the tilde (e.g., in a data set with students nested in schools, the school identifier code), and three-level imputation requires level-2 and level-3 identifier variables. The order of the identifier variables does not matter.

Imputation model

At a minimum, the imputation model list should include all variables and effects of interest in the analysis model(s), but the list may also include additional auxiliary variables. Note that the variable list should include both complete and incomplete variables, and the user need not specify which variables have missing values. Additionally, Blimp will automatically determine the level at which a variable is measured in a multilevel data set.

The following example illustrates the `MODEL` statement for a two-level imputation problem. The single identifier variable `idvar` to the left of the tilde indicates a two-level data structure, and the variables included in imputation are `X`, `Y`, and `Z`.

```
MODEL: idvar ~ X Y Z;
```

3.3.2 Specifying Random Slopes

Blimp automatically includes random intercepts for all incomplete variables in a multilevel data set. The FCS algorithm can also accommodate multilevel analyses with random slopes. To specify a random slope between two variables, simply place a colon (`:`) between two variables instead of a space. Returning to the previous example, suppose that `X` and `Y` had a random association in the analysis model. The imputation model for this problem would specify a random slope as follows.

```
MODEL: idvar ~ X:Y Z;
```

Note that the `MODEL` statement makes no reference to a variable's role in the analysis model. In the example above, `X` serves as a random predictor in the `Y` imputation model, and `Y` serves as a random predictor in the `X` imputation model. If `Y` is to have a random slope with another variable, `Z`, then `Y` is repeated twice in the model statement, as follows:

```
MODEL: idvar ~ X:Y Y:Z;
```

If a random slope is specified between `Y` and `X`, `Y` and `Z`, and `X` and `Z`, then one can chain the statements in a compact form, as follows:

```
MODEL: idvar ~ X:Y:Z;
```

3.4 Algorithmic Options

`BURN` The `BURN` command is used to specify the number of burn-in iterations that are performed prior to saving the first imputed data set. The number of iterations should be determined by the convergence diagnostics addressed in Chapter 6.

`THIN` The `THIN` command is used to specify the thinning (between-imputation) interval, which is the number of iterations separating each data set. For example, a value of 1000 would generate an imputed data set after every 1000th computational cycle. Again, the convergence diagnostics discussed in Chapter 6 can be useful for specifying this value.

`NIMPS` The `NIMPS` command is used to specify the number of imputed data sets.

`SEED` The `SEED` command is used to specify the pseudorandom number generator seeding value. The input value to the command is required to be a positive integer with nine or fewer digits. Note that Blimp requires a seed value to run.

`CHAINS` The `CHAINS` command is used to specify the number of Gibbs sampler chains. When multiple chains are specified, Blimp takes the number of imputed data sets requested and distributes them across the chains. For example, if four imputations are requested from two chains, each chain will generate two imputed data sets. By default Blimp will run chains simultaneously up to the number of physical cores present in the computer at a time. This can drastically speed up the imputation process by distributing the imputed data sets requested across multiple processors. To override this default by specifying less processors one can modify the number of processors that will be used with the `processors` keyword followed by the number desired:

```
CHAINS: 10 processors 1;
```

The above line will specify ten chains using only one processor at a time.

The default value for number of chains is one, unless the `psr` keyword is specified on the `OPTIONS` command, in which case two chains are used. It is important to note that each chain will have a different seeding value along with different randomly perturbed start values for the parameters in the imputation model.

`OUTFILE` The `OUTFILE` command is used to specify the location and name of the imputed data set(s). The `OUTFILE` command requires the full file path to be specified. As explained in the `OPTIONS` section below, Blimp can stack the imputed data sets in a single file, or it can save each imputed data set to a separate file. When saving imputations to separate files, an asterisk (`*`) is required in the file path (e.g., see Example 2.1 in the quick start section of the document). This asterisk will be replaced with an integer in the file name (e.g., a filename of myimps*.dat would produce imputed data sets named myimps1.dat, myimps2.dat, etc.). The asterisk convention is also required when employing Blimp's internal simulation features, which are described in Section 7.1. No asterisk is needed when writing data sets to a single stacked file.

`OPTIONS` The `OPTIONS` command is used to specify algorithmic and other miscellaneous options (e.g., the format of the output data sets). If this command is excluded, Blimp will rely on default settings. Keywords (described below) are separated by a space (e.g., see Example 2.1 in the quick start section).

3.5 `OPTIONS` Keywords

This section describes the keywords available with the `OPTIONS` command. Note that **`BOLDED UPPERCASE`** keywords are default settings that will be used unless otherwise specified.

`CSV` / `dat` Blimp can save imputed data sets in comma separated values (CSV) format or in space-delimitted format. The `csv` and `dat` keywords specify these options, respectively, with `csv` as the default.

`STACKED` / `separate` The `stacked` keyword (the default) produces a single file with imputed data sets stacked one on top of another. In this case, the first variable in the output data set is an identifier variable that indexes the data sets. The `separate` keyword writes each imputed data set to a separate text file. This specification requires an asterisk in the file name, and this asterisk is subsequently replaced by a numeric value (e.g., a filename of myimps*.dat would produce imputed data sets named myimps1.dat, myimps2.dat, etc.). The choice of stacked or separate files depends on the analysis software being used. Among others, SAS and SPSS require stacked input files, whereas Mplus and HLM require imputations in separate files. To facilitate the use of Blimp data with Mplus, the `separate` keyword also creates a list file that contains the file paths of the imputed data sets. This list file is used as the input data for an Mplus analysis.

HEV / `hov` In a two-level data set, the `hev` (`heterogeneous`) keyword specifies an imputation model with heterogeneous within-cluster residual variances (see Kasim & Raudenbush, 1998; Van Buuren et al., 2011). Such a specification might be warranted with daily diary data, for example. The `hov` (`homogenous`) keyword specifies a common residual variance for all clusters, which is in line with the standard representation of the multilevel model.

CLMEAN / `noclmean` With multilevel data, the `clmean` keyword (the default) introduces the cluster means as additional predictors in the imputation model. For example, suppose that `X` and `Y` are level-1 variables in a two-level imputation scheme. The `clmean` keyword would introduce `X` and its cluster means as predictors in the `Y` imputation model, and vise versa. The `clmean` keyword is particularly important for analyses that posit unique within-cluster and between-cluster covariance structures. Such is the case with contextual effects regression models and many multilevel structural equation models (e.g., a multilevel factor model that posits different factor structures or factor loadings at level-1 and level-2). The `noclmean` keyword specifies a parsimonious imputation model that does not partition associations into their within- and between-cluster components. The `noclmean` option is appropriate for analysis models where the level-1 and level-2 regressions are assumed to be identical (i.e., a model with no contextual effects).

PRIOR1 / `prior2` For all regression coefficients in the imputation model, Blimp implements standard non-informative prior distributions, and these priors cannot be modified. The program offers two choices of prior distribution for variance parameters. In a multilevel model, priors can be specified for level-2 (and level-3) covariance matrices and the within-cluster residual variance. In a single-level model, the residual variance is the only parameter to which the priors apply. The `prior1` keyword implements an inverse Wishart, $W^{-1}(\mathbf{I}, p+1)$, where $\mathbf{I}$ is a $p \times p$ identity matrix and p is the number of variance estimates in the covariance matrix. The `prior2` keyword specifies an inverse Wishart prior, $W^{-1}(0, -p-1)$, where p is the number of variance estimates in the covariance matrix. In a multilevel application, the `prior1` keyword is the default, and `prior2` is the default for single-level models. In

Monte Carlo simulations, we have determined that these default priors tend to provide the best parameter recovery in a broad range of scenarios, so there is often no reason to alter the default settings.

`NOPSR` / `psr` The `psr` keyword prints a table of potential scale reduction (PSR) factors (Gelman & Rubin, 1992). The PSR values can be used to diagnose the convergence of the iterative Gibbs sampler. Imputation for each incomplete variable is based on a regression model, the parameters of which define a distribution of plausible imputations. In a multilevel model, the parameters consist of regression coefficients, covariance matrices (level-2 or level-3), a within-cluster residual variance, and threshold parameters (ordinal variables with more than two categories). In a single-level model, the imputation model parameters are regression coefficients, a residual variance, and possibly threshold parameters. PSR values are computed for every parameter in the imputation models, and the `psr` keyword prints a brief summary table that reflects the worst (highest) PSR value for the regression coefficients, variance/covariance estimates, and threshold parameters (if applicable). The `nopsr` keyword is the default, meaning that no diagnostic information is printed. See Chapter 6 for more information.

4 Running Blimp

Blimp is executed via the Run pull-down option from the GUI or via the command-line. Double-clicking a Blimp syntax file (a text file ending in the .imp extension) will automatically open the GUI. The syntax file is then submitted for execution by selecting the Run option from the Impute pull-down menu.

Alternatively, Blimp can be executed via the command-line. A terminal emulator program (e.g., Terminal.app in macOS, cmd.exe in Windows) is used to run Blimp from the command-line. The command-line arguments for Blimp provide flexibility for advanced applications, primarily Monte Carlo computer simulations. This application is covered in Chapter 7, and this chapter will focus on running Blimp with a single data set.

The Windows version of Blimp automatically allows users to execute files via the command-line. In Mac OS, the executable file is packaged in the application bundle and is not immediately accessible to users. To enable this functionality from the GUI, select the command-line Tool option from the main Blimp pull-down menu. A pop-up dialog box will ask whether the command-line tool should be installed. After enabling this functionality, Blimp can be executed from a shell program such as the Terminal.app application.

To run Blimp from a command-line, type blimp followed by the full file path to the syntax file. For macOS, the command-line specification for macOS is as follows:

```
$ blimp ~/desktop/InputExample.imp
```

where 'InputExample.imp' is a Blimp syntax script saved in the user's Desktop directory. Most terminal applications (Mac or Windows) automatically generate the appropriate file paths when the application and syntax files are dragged and dropped into the terminal window.

5 Blimp Ouput

After executing Blimp (GUI or shell), a header is printed to the output window. This header provides the version number along with contact information. An example header is shown below.

```
                              Blimp
                            Beta v6.8

              Developed by Craig K. Enders and Brian T. Keller.
                   Blimp was developed with funding from
             Institute of Educational Sciences award R305D150056.
             Craig K. Enders, P.I. Email: cenders@psych.ucla.edu.
           Programming by Brian T. Keller. Email: bkeller2@ucla.edu
            This is Beta software with no expressed license given.


                 DO NOT DISTRIBUTE WITHOUT WRITTEN CONSENT
```

Below the header, Blimp lists the algorithmic options specified in the syntax file. In addition to specified options, the list includes default settings that may not have been explicitly specified in the syntax. An example summary of the algorithmic options is shown below.

```
Algorithmic Options Specified:
        hov, clmean, Raneff Prior 1, Resvar Prior 1.
```

Following the algorithmic options, Blimp displays the progress of the Gibbs sampler algorithm. For the initial period of burn-in iterations the program issues a message when it has completed 25%, 50%, 75%, and 100%. Following the burn-in phase, Blimp prints a time and date stamp as it saves each imputed data set.

```
Starting Burn-in on Wed Nov  2 12:06:24 2016
    Burn-in iteration 500 complete on Wed Nov  2 12:06:24 2016
    Burn-in iteration 1000 complete on Wed Nov  2 12:06:24 2016
    Burn-in iteration 1500 complete on Wed Nov  2 12:06:24 2016
    Burn-in iteration 2000 complete on Wed Nov  2 12:06:24 2016
Burn-in complete on Wed Nov  2 12:06:24 2016
    Imputation Saved 1 on Wed Nov  2 12:06:24 2016
    Imputation Saved 2 on Wed Nov  2 12:06:24 2016
    Imputation Saved 3 on Wed Nov  2 12:06:24 2016
    Imputation Saved 4 on Wed Nov  2 12:06:25 2016
    Imputation Saved 5 on Wed Nov  2 12:06:25 2016
    Imputation Saved 6 on Wed Nov  2 12:06:25 2016
    Imputation Saved 7 on Wed Nov  2 12:06:25 2016
    Imputation Saved 8 on Wed Nov  2 12:06:25 2016
    Imputation Saved 9 on Wed Nov  2 12:06:25 2016
    Imputation Saved 10 on Wed Nov  2 12:06:25 2016
```

After all imputations are saved, Blimp prints the order of the variables in the inputed data set(s). All variables listed on the `VARIABLES` command are saved to the file(s), regardless of whether the variables are used in imputation. An example variable list is shown below.

```
Variable Order: id av y x1 x2 x3 w1 w2 w3
```

Variables are saved in the same order as they appear on the `VARIABLES` command, except when imputations are stacked in a single file (the `stacked` keyword of the `OPTION` command). In this case, the first variable in the file is an identifier variable that indexes the imputed data sets.

Finally, the Blimp output will print the PSR summary tables if the `psr` keyword is specified on the `OPTIONS` command. These tables are discussed in Chapter 6.

5.1 Multiple Chains output

When multiple chains are requested, Blimp prints slightly different output. The follow excerpts of output are based on an example of eight chains and eight imputations. Initially, each chain will display the seed value used. These seeds are randomly generated based on the seed value supplied in the `SEED` command.

```
Chain 1 online with seed of 158875
Chain 2 online with seed of 98783
Chain 3 online with seed of 172863
Chain 4 online with seed of 32484
```

During this time, no output will be displayed about the status of burn-in iterations. Instead, when a chain saves an imputation the chain number and the imputation number will be printed. It is important to note that the imputations may not finish in consecutive order.

```
    Chain 4 saving imputation 4 complete on Fri Nov 11 2016
    Chain 2 saving imputation 2 complete on Fri Nov 11 2016
    Chain 1 saving imputation 1 complete on Fri Nov 11 2016
    Chain 3 saving imputation 3 complete on Fri Nov 11 2016
```

Upon completion of all imputations required in the first set of chains, the next set of chains will start and subsequently save imputations.

```
Chain 5 online with seed of 163110
Chain 6 online with seed of 192944
Chain 7 online with seed of 106865
Chain 8 online with seed of 31813
    Chain 8 saving imputation 8 complete on Fri Nov 11 2016
    Chain 6 saving imputation 6 complete on Fri Nov 11 2016
    Chain 5 saving imputation 5 complete on Fri Nov 11 2016
    Chain 7 saving imputation 7 complete on Fri Nov 11 2016
```

This process will continue until all chains and imputations have completed.

6 Convergence Diagnostics with Blimp

The `psr` keyword of the `OPTIONS` command prints a table of potential scale reduction (PSR) factors (Gelman & Rubin, 1992). The PSR values can be used to diagnose the convergence of the iterative Gibbs sampler. Unless otherwise specified, specifying the `psr` keyword invokes two MCMC chains. PSR values are calculated after every 100 iterations of the specified burn-in period. Each time the PSR is computed, the first half of the iterations are discarded, and PSR values are computed by comparing parameter estimates from the two chains. Note that Blimp will compute the PSR factors only during the burn-in period. Therefore, when the burn-in period is finished, Blimp will continue to run until the desired number of imputed data sets are obtained without printing out the PSR factors.

Imputation for each incomplete variable is based on a regression model, the parameters of which define a distribution of plausible imputations. In a multilevel model, the parameters consist of regression coefficients, covariance matrices (level-2 and level-3), a within-cluster residual variance, and threshold parameters (ordinal variables with more than two categories). In a single-level model, the imputation model parameters are regression coefficients, a residual variance, and possibly threshold parameters. PSR values are computed for every parameter in the imputation models, and the output prints summary table that reflects the worst (highest) PSR value for the regression coefficients, variance/-covariance estimates, and threshold parameters (if applicable).

The tabled values of the PSR are divided into four categories: Fixed effects regression coefficients (labeled `Fix eff`), level-2/level-3 variance-covariance matrix parameters (labeled `Ran Var`), within-cluster residual variance parameters (labeled `Err var`), and threshold parameters for ordinal variables with more than two categories (labeled `Threshold`). Below each category label Blimp displays the maximum PSR across all imputation models for a given parameter category. An example PSR table is shown below.

```
Starting Gibbs sampler on Fri Oct  2 13:22:44 2015
-----------------------------------------------------------------

PSR: Comparing iterations 51 to 100 for 2 chains.

           |   Fix eff|   Ran Var|   Err Var| Threshold|
---------------------------------------------------------
    Max PSR:|     1.370|     1.115|     1.124|       nan|
   Variable:|         y|         x|         x|          |

-----------------------------------------------------------------
```

The example table above reflects the PSR computations at the 100th iteration of the burn-in phase. The first 50 iterations are discarded, and PSR values are computed by comparing parameter estimates from two separate MCMC chains with 50 iterations (iterations 51 through 100). As a second example, consider the PSR calculation at iteration 400. Again, the first half of the iterations are discarded, and PSR values are computed by comparing parameter estimates from two separate MCMC chains with 200 iterations (iterations 201 through 400). An example table is shown below.

```
PSR: Comparing iterations 251 to 400 with 2 chains.

           |   Fix eff|   Ran Var|   Err Var| Threshold|
---------------------------------------------------------
    Max PSR:|     1.048|     1.003|     1.027|       nan|
   Variable:|         y|         y|         y|          |

-----------------------------------------------------------------
```

As noted above, PSR values are computed for every parameter in the imputation models, and the output prints summary table that reflects the worst (highest) PSR value for the regression coefficients, variance/covariance estimates, and threshold parameters (if applicable). The labels in the Variable row reflect the incomplete variable that generated each PSR value. For example, in the table above, the PSR values are all from the regression model used to impute variable `Y`. PSR values less than 1.05 to 1.10 are typically viewed as acceptable.

7 Running a Simulation

Blimp has utilities to run simulations with two different methods: (1) internal Blimp simulations and (2) external Blimp simulations. Internal simulations use Blimp to automate the imputation of multiple replicates with a single Blimp syntax file. External Blimp simulations allow users to impute replicates with additional scripting from outside programs (e.g., R, Bash, etc.). Both methods are designed to easily automate Blimp for the purposes of methodological simulations.

7.1 Internal Blimp Simulations

Internal Blimp simulations offer users an easy interface to quickly run Blimp on a collection of simulated data sets. Example 7.1 below demonstrates an internal simulation. For an internal simulation, a set of artificial data sets must have a common file name and a consecutive integer ranging from 1 to the desired number of replications (e.g., myrep1.dat, myrep2.dat, . . . , myrep1000.dat). The `SIMULATE` command is added to the first line of the Blimp syntax file, and the command has two keywords. The `replications` keyword must be followed by a space and the number of artificial data sets. Blimp will start from 1 and increment the number by 1 in the file path until reaching the total number of replications. In addition, a range of replications can be ran by specifying a starting replication, the keyword `to` and the ending replication. For example, `replications 200 to 500`. The `processors` keyword must be followed by a space and the number of total processors used. Blimp will restrict the number of processors based on computer hardware specifications (and allow no more than eight total). Specifying multiple processors will run each replication on a separate processor. For example, specifying: `processors 4` will run 4 replications at a time until all are finished.

Example 7.1: Internal Simulation Syntax

```
SIMULATE: replications 1000 processors 4;

DATA: /Users/name/Desktop/myrep*.dat;

VARIABLES: idvar x y z;

MODEL: idvar ~ x y z;

BURN: 500;

THIN: 100;

NIMPS: 10;

MISSING: -9999;

SEED: 38203;

CHAINS: 1;

OUTFILE: /Users/name/Desktop/myimps*.csv;

OPTIONS: csv;
```

When invoking the `SIMULATE` command, the `DATA` command is used to specify the file name prefix for the artificial data sets, and an asterisk is used as a placeholder for the replication number. To illustrate, consider Example 7.1. The `SIMULATE` command specifies 1000 replications, and the `DATA` command indicates that the file names are myrep1.dat, myrep2.dat, . . . , myrep1000.dat. The `OUTFILE` command follows a similar convention. In Example 7.1, the imputed data sets are saved to files named myimps1.csv, myimps2.csv, . . . , myimps1000.csv. Note that one cannot specify the `separate` keyword when using the `SIMULATE` command, as the imputed data sets for each replication will always be saved in the stacked format described in Section 3.4. In addition, the `psr` keyword is not available with the SIMULATE command.

7.2 Output from **SIMULATE** command

The `SIMULATE` command produces the same screen output if it is a single processor simulation or a multiprocessor simulation. Adjusting the input in Example 7.1 to run five replications would produce the following output after Blimp's standard header.

```
WARNING: Entering simulation mode.



-----------------------------------------------------------------

Algorithmic Options Specified:
        hov, clmean, Raneff Prior 1, Resvar Prior 1.
-----------------------------------------------------------------


Starting simulation on Mon Oct  5 13:49:51 2015
-----------------------------------------------------------------

   Finished replication 1 on Mon Oct  5 13:49:52 2015
   Finished replication 4 on Mon Oct  5 13:49:52 2015
   Finished replication 2 on Mon Oct  5 13:49:52 2015
   Finished replication 3 on Mon Oct  5 13:49:52 2015
   Finished replication 5 on Mon Oct  5 13:49:52 2015
-----------------------------------------------------------------

Finished simulation on Mon Oct  5 13:49:52 2015
-----------------------------------------------------------------

Variable Order: imp# x y z
-----------------------------------------------------------------
```

Note that the replications will not necessarily finish in numerical order when more than one processor is used. Also note that if any errors occur that force a replication to exit, the replication will not have any imputations saved. That is, if the MCMC algorithm encounters computational problems for a particular replication after the 1st imputation is complete, the program does not produce an output data set that contains a partial set of imputations. Instead, Blimp will print out an error message that lists the failed replication number. Convergence problems can often be solved by specifying a different seed for the failed replicate, and failed replications can be addressed with the external simulation procedure described below.

7.3 External Blimp Simulations

An external Blimp simulation uses a common Blimp syntax file but changes specific commands in that script via command-line arguments. The use of external simulations allows advanced users to circumvent some of the limiting factors of internal simulations (e.g., inability to run diagnostics, replication numbers must increment by one, etc.). The command-line arguments for an external Blimp simulation are demonstrated by the following terminal commands.

```
$ /path/to/blimp/Blimp /path/to/input/InputExample.imp
       -d /path/to/data/data1.dat
       -o /path/to/output/output1.dat
       -s 287123
```

As before, the terminal command requires two file paths, the path to the executable, and the path to the Blimp syntax file. In the above example, three command-line arguments modify the file "InputExample.imp" (e.g., the script file that will be applied to a number of artificial data sets).

`-d` or `--data`

First, the `-d` or `--data` argument will override the file path given by the `DATA` command in the syntax file. A warning is displayed when this argument is used.

```
WARNING: Overriding DATA command in input.
```

`-o` or `--outfile`

The `-o` or `--outfile` argument will override the file path given by the `OUTFILE` command in the syntax file. Again, a warning is displayed when this argument is used.

```
WARNING: Overriding OUTFILE command in input.
```

`-s` or `--seed`

The `-s` or `--seed` argument will override the `SEED` command in the syntax file. A warning is displayed when this argument is used.

```
WARNING: Overriding SEED command in input.
```

In order to automate a simulation, a scripting or programming language must be employed to submit the command-line arguments to the operating system. For convenience, we have included a script to do this in Example 7.2 using the R statistical programming language the `system()` function.

Example 7.2: R script to automate simulation external of Blimp

```
## Running a simulation via External Blimp method.
#    Note no spaces should be in the directory paths
#    All output from Blimp Is saved in a list: outputs
#    Does not work with 'sep' subcommand

# Set a Seed
seed       <- 37298372

# Path to Blimp
blimpPath  <- '~/Desktop/Blimp'

# Specify Path to Syntax File
inputPath  <- '~/Desktop/myInput.imp'

# Specify Path to Data Folder
#   Only the data files should be in folder.
dataPath   <- '~/Desktop/myDataFolder'

# Specify Path to Output Folder
outputPath <- '~/Desktop/myImpsFolder'

# Specify Names of Imputation Data
#   Use a * to represent where the name of data file.
#   E.g., Data1.csv will give you impData1.csv
impsData   <- 'imp*.csv'

#######################################################
## PROGRAM BEGINS HERE
# Get file names
dataFiles <- list.files(path = dataPath)
dataFilesPath <-list.files(path = dataPata, full.names = T)
# Calculate total number of reps.
repNumber <- length(dataFiles)
# Set seed
set.seed(seed)
# Generate list of seeds
seeds <- sample.int(1e10, repNumber,replace = F)
# Execute Simulation
outputs <- lapply(seq_along(seeds),function(x){
  # Construct path names
  repla <- gsub('\\..+','',dataFiles[x],perl=T)
  fileP <- gsub('.+\\.','',dataFilesPath[x],perl=T)
  outfile  <- paste0(fileP,gsub('\\*',repla,impsData,perl=T))
  out <- system(paste(blimpPath,inputPath,'-o',outfile,'-s',
              seeds[x],'-d',dataFilesPath[x]),intern = T
  return(list(dataFiles[x]),out)
})
```

8 Error and Warning Message Reference

Error messages (denoted with `ERROR` in the Blimp output) are issued when problems arise due to computational or syntactical issues and cause Blimp to stop running and exit. In contrast, warning messages (denoted with `WARNING` in the Blimp output) will only warn the user that some options may have been adjusted, but Blimp will continue to run. The following sections detail Blimp error and warning messages and possible resolutions for each error.

8.1 Error Messages

```
ERROR: Unable to open syntax file.
```

Blimp was unable to open the Blimp syntax file. This usually indicates that the file path supplied to the Blimp cannot be found.

```
ERROR: Must specify SIMULATE command before specifying data
   command.
```

The `SIMULATE` command must be specified before the `DATA` command in the Blimp syntax file. For more information on how to properly set up a simulation input see Section 7.1.

```
ERROR: Must specify more than 0 replications.
```

Blimp read the number of replications specified in the `SIMULATE` command as 0. Check input to verify that the number is greater than 0. For more information on how to properly set up a simulation input see Section 7.1.

```
ERROR: Must specify more than 0 processors.
```

Blimp read the number of processors to use in the `SIMULATE` command as 0. Check input to verify that the number is greater than 0. For more information on how to properly set up a simulation input see Section 7.1.

```
ERROR: Chains commmand not currently available in simulation
   mode.
```

The `CHAINS` command was used with simulation mode. Only single chains are allowed with internal simulations. To specify multiple chains in a simulation you must run an external simulation. For more information on external simulations see Section 7.3.

```
ERROR: Simulation mode already specified, psr command not
   available.
```

Currently the `psr` keyword is not available in simulation mode. To obtain PSR factors in a simulation you must run an external simulation. For more information on external simulations see Section 7.3.

```
ERROR: Missing a DATA, VARIABLES, or MODEL command.
```

Either the `DATA`, `VARIABLES`, and/or `MODEL` are/is missing or not read correctly by Blimp. Double check that all lines end in a semicolon (`;`). For more information on specifying a Blimp syntax file see Chapter 3.

```
ERROR: Missing imputation parameters.
```

Either the `BURN`, `THIN`, `MISSING`, `SEED`, and/or `NIMPS` command(s) are/is missing or not read correctly by Blimp. Double check that all lines end in a semicolon (;). For more information on specifying a Blimp syntax file see Chapter 3.

```
ERROR: No output file given.
```

The `OUTFILE` command is missing or not read correctly by Blimp. Double check that all lines end in a semicolon (`;`). For more information on specifying a Blimp syntax file see Chapter 3.

```
ERROR: Please place ONE asterisk (*) in OUTFILE command.
```

Blimp did not detect an asterisk (*) in the `OUTFILE` command or Blimp detected more than one asterisk. This is required when the `SIMULATE` command is used or the `separate` keyword is specified in the `OPTIONS` command. For more information on specifying a Blimp syntax file see Chapter 3.

```
ERROR: More processors than simulation files requested.
```

More processors were requested than the number of replications specified in the `SIMULATE` command. For more information on setting up a simulation see Chapter 7.

```
ERROR: More processors than chains requested.
```

More processors were requested than the number of chains specified in the `CHAINS` command. For more information on specifying a Blimp syntax file see Chapter 3.

```
ERROR: More chains than imputations requested.
```

More chains than the number of imputations were requested in the syntax file. For more information on specifying a Blimp syntax file see Chapter 3.

```
ERROR: Requested seed of --- must be greater than 0.
```

All seeds must be positive integers.

```
ERROR: DATA command and OUTFILE command have the same file path.
```

The file path specified in the `DATA` command and in the `OUTFILE` command are the same. These two paths must differ.

```
ERROR: File /MY/FILE/PATH cannot be found.
```

The file specified in the DATA command (labeled as '/MY/FILE/PATH' in the example error message) does not exist. Check to see if the line is terminated by a semicolon (;).

```
ERROR: File /MY/FILE/PATH cannot be created.
       Please check that the file path is correct.
```

The file specified in the OUTFILE command (labeled as '/MY/FILE/PATH' in the example error message) cannot be created. Check to see if the line is terminated by a semicolon (;).

```
ERROR: Currently no more than two identifier variables are
   supported.
```

Blimp has interpreted more than two identifier variables in the MODEL command. Blimp currently only allows a maximum of three-levels (i.e., two identifier variables).

```
ERROR: Variable X1 has only one observed category.
       Ordinal and nominal variables need
       a minimum of two observed categories.
```

The variable 'X1' (where 'X1' could be any variable name) was listed on the ORDINAL or NOMINAL command, but has only one observed category. Blimp requires more than one category to be observed in order to impute the variable.

```
ERROR: Variable: X1 not in data.
```

The variable 'X1' (where 'X1' could be any variable name) was listed in the MODEL command and not in the VARIABLES command.

```
ERROR: Identifier variable: SUBJID not in data.
```

The identifier variable SUBJID (where SUBJID could be any identifier variable name) listed prior to the tilde in the MODEL command does not appear in the VARIABLES command.

```
ERROR: Identifier variables have the same number of clusters.
       Cross-classified models are currently unsupported.
```

Blimp has detected the identifier variables have the same number of clusters. At this time, Blimp does not support cross-classified models.

```
ERROR: Identifier’s appear to be a cross-classified model.
       Cross-classified models are currently unsupported.
```

Blimp has detected the identifier variables do not follow the proper nesting structure. At this time, Blimp does not support cross-classified models.

```
ERROR: Please place ONE asterisk (*) in DATA command.
```

More than one asterisk (*) was placed in the file path given in the `DATA` command. This can also be triggered when no asterisk is found. An asterisk is required for the SIMULATE command.

```
ERROR: Failed to read the data in,
       please use a comma separated file
       or space separated file.
```

Blimp was unable to read the data. This could be due to the file path specified not being correct or due to the file type not being recognized.

```
ERROR: The number of variables listed does not equal data
   columns.
```

The number of variables listed in the `VARIABLES` command does not equal the number of columns the data matrix read in by Blimp.

```
ERROR: No missing variables in MODEL command.
```

There were no missing variables listed in the `MODEL` command.

```
ERROR: A matrix is numerically positive indefinite
       Either:
              (1) Try another seed.
              (2) Specify fewer random effects.
              (3) Specify fewer variables in model.
```

During the imputation process, a matrix became numerically positive indefinite and Blimp was unable to continue because of an inversion problem. It is recommended to first try another seed. If this continues, then try to specify fewer random effects. Finally, try specifying fewer variables.

```
   Chain i failed with status 2.
```

Chain `i`, where `i` is a number, had a matrix become computationally positive indefinite. If this is the case, it is recommended to first try another seed. If this continues, then try to specify fewer random effects. Finally, try specifying fewer variables.

```
   Replication i return an error code of 1.
```

During the simulation, the replication `i`, where `i` is a number, had an error and returned the code of error code of 1. An error code of 1 indicates a failure to load the data set.

```
   Replication i return an error code of 2.
```

During the simulation, the replication `i`, where `i` is a number, had an error and returned the code of error code of 2. An error code of 2 indicates that a matrix became numerically positive indefinite. If this is the case, it is recommended to first try another seed.

8.2 Warning Messages

```
WARNING: Overriding OUTFILE command in input.
```

An argument from the command-line is overriding the file path given in the `OUTFILE` command (e.g., when implementing an external simulation). If this is not desired, see Chapter 4.

```
WARNING: Overriding SEED command in input.
```

An argument from the command-line is overriding the value given in the `SEED` command (e.g., when implementing an external simulation). If this is not desired, see Chapter 4.

```
WARNING: Overriding DATA command in input.
```

An argument from the command-line is overriding the file path given in the `DATA` command (e.g., when implementing an external simulation). If this is not desired, see Chapter 4.

```
WARNING: Entering simulation mode.
```

Simulation mode was enabled with a `SIMULATE` command.

```
WARNING: Separate data files are not available for simulation
   mode.
```

The `separate` keyword was specified with the `SIMULATE` command. The `SIMULATE` command will override the `separate` keyword.

```
WARNING: Zero or less imputations requested. Defaulting to one
   imputation.
```

The number of imputations requested were read in to be zero or a negative number. Blimp is setting the number of imputations to one.

```
WARNING: Less than zero thinning interval requested. Defaulting
   to zero.
```

The thinning interval must be greater than or equal to zero. Blimp is defaulting to zero.

```
WARNING: Less than zero burn-in requested. Defaulting to zero.
```

The burn-in interval must be greater than or equal to zero. Blimp is defaulting to zero.

```
WARNING: Maximum number of processors allowed is 4.
         This is based on your hardware specifications.
         Setting the processors to 4.
```

Blimp will not allow more processors requested than number of physical CPU cores in the computer (note in the above warning 4 may change depending on the hardware). Blimp will default to the maximum number of allowed processors.

```
WARNING: Multithreading not enabled.
         Setting the processors to 1.
```

The distribution of Blimp being used does not have multithreading enabled. Therefore, Blimp is setting itself to one processor.

```
WARNING: A minimum of 100 burn-in needed for PSR.
         Setting burn-in to 100.
```

The `psr` keyword was specified in the `OPTIONS` command and the burn-in requested was less than 100. Blimp prints the PSR statistic for every 100 burn-in iterations. Therefore, Blimp is defaulting to a minimum of 100 burn-in iterations.

```
NOTE: Setting number of imputations to 2
      to match number of chains requested.
```

More chains were requested than imputations. A minimum of one imputation per chain must be requested. Blimp is defaulting the number of imputations to the number of chains requested.

```
WARNING: Multithreading not enabled.
         Reverting to single thread simulation.
```

The distribution of Blimp being used does not have multithreading enabled. Therefore, Blimp is using a single-threaded simulation.

```
WARNING: Variable "X1" in ORDINAL command was not used.
         Cannot be found in VARIABLE command.
```

The variable 'X1' (note this will be the name of variable causing the warning message) was used in the `ORDINAL` command, but is not listed in the `VARIABLE` command. It will be ignored.

```
WARNING: Ignoring variable "X1" in ORDINAL command is not in
   MODEL command.
         Cannot be found in MODEL statement.
```

The variable 'X1' (note this will be the name of variable causing the warning message) was used in the `ORDINAL` command, but is not listed in the `MODEL` command. It will be left out of the model and ignored.

```
WARNING: Variable "X1" in NOMINAL command was not used.
         Cannot be found in VARIABLE command.
```

The variable 'X1' (note this will be the name of variable causing the warning message) was used in the `NOMINAL` command, but is not listed in the `VARIABLE` command. It will be ignored.

```
WARNING: Ignoring variable "X1" in NOMINAL command is not in
   MODEL command.
         Cannot be found in MODEL statement.
```

The variable 'X1' (note this will be the name of variable causing the warning message) was used in the `NOMINAL` command, but is not listed in the `MODEL` command. It will be left out of the model and ignored.

```
WARNING: No identifier variable specified.
         Defaulting to single-level imputation.
```

Blimp did not interpret an identifier variable in the `MODEL` command. Blimp will treat the file as a single-level imputation. If this is desired, ignore this warning. Otherwise, check the syntax's `MODEL` command. See Section 3.3 on specifying a `MODEL` command.

```
WARNING: 5 observations have all variables in the imputation
         model missing. They have been dropped from data set.
```

Blimp has dropped 5 observations, where 5 is the number specific to the data set. Blimp will not impute variables with missing observations on all variables in the `MODEL` command.

```
WARNING: Excluding the following variables as predictors at the
   listed level.
           Variable "X1" excluded from level 2 imputation.

         These variables are either orthogonal to all variables
            at that level
         (e.g., because they lack variation at that level) or
            their cluster
         mean is linearly dependent with another variable at
            that level.
```

The variable 'X1' (note this will be the name of variable causing the warning message) is being excluded from the imputation model at the level specified in the message (i.e., 2 in the above example message).

References

Enders, C. K., Keller, B. T., & Levy, R. (2016). A chained equations imputation approach for multilevel data with categorical and continuous variables. Manuscript submitted for publication.

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical science*, 457–472.

Kasim, R. M., & Raudenbush, S. W. (1998). Application of gibbs sampling to nested variance components models with heterogeneous within-group variance. *Journal of Educational and Behavioral Statistics*, *23*(2), 93–116.

Van Buuren, S., et al. (2011). Multiple imputation of multilevel data. *Handbook of advanced multilevel analysis*, 173–196.