Blimp User's Guide

Version 2.2

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

January 2020

Developed by Craig K. Enders, Brian T. Keller, and Han Du

Blimp was developed with funding from Institute of Educational Sciences awards R305D150056 and R305D190002.

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

Programming by Brian T. Keller, Co-P.I. Email: bkeller2@ucla.edu

Han Du, Co-P.I. Email: hdu@psych.ucla.edu

Graphical interface by Brian T. Keller (version 0.8.0 and beyond) and Behrouz NematiPour.

For support please email blimp.imputation@gmail.com.

Any published work derived from the use of this software, please cite the following sources:

Enders, C. K., Du, H., & Keller, B. (2019). A model-based imputation procedure for multilevel regression models with random coefficients, interaction effects, and non-linear terms. Psychological Methods, Advance Epub.

Enders, C. K., Keller, B. T., & Levy, R. (2018). A chained equations imputation approach for multilevel data with categorical and continuous variables. Psychological Methods, 23, 298-317.

Keller, B. T., & Enders, C. K. (2020). Blimp User's Manual (Version 2.2). Los Angeles, CA.

Contents

1 Introduction

Blimp is a software program designed to perform Bayesian estimation and multiple imputation. Its two primary algorithms are fully conditional specification (FCS; Enders, Keller, & Levy, 2018) and fully Bayesian (model-based) estimation and imputation (MBI; Enders, Du, & Keller, 2019).

FCS parallels the MICE (Multiple Imputation by Chained Equations) algorithm popularized by Stef van Buuren and colleagues. FCS is appropriate for a broad class of univariate and multivariate applications. Blimp's FCS routine uses an ordered probit and multinomial probit model (i.e., a latent variable formulation for categorical variables) to impute incomplete binary, ordinal, and nominal variables. In multilevel data sets, between-cluster associations can be modeled with latent group means (random intercepts) or with cluster means of the manifest variables. The FCS routine offers a number of specialized options, including multiple-group imputation and heterogeneous level-1 variance structures (two-level models).

MBI performs Bayesian estimation for single-level and multilevel regression models in data sets with up to three levels. MBI is designed for analysis models with interaction effects, polynomial terms, and/or random coefficients, but it is also appropriate for additive regression models. MBI accommodates binary, ordinal, and continuous outcomes, and predictors may continuous, binary, ordinal, or nominal. Categorical imputation uses a latent variable formulation for the incomplete variables (probit regression). Blimp implements two versions of MBI. The fully Bayesian (latent) approach models the full distribution of the covariates, including latent group means at the between-cluster. An approximate (manifest) approach uses manifest-variable group means to model covariates at the between level.

MBI can produce and save imputations, but it also produces Bayesian estimates of the analysis model parameters, and thus can be used as a primary analytic tool without saving imputations. Blimp's Bayesian estimation routine can accommodate a range of common analysis models, including models with (a) random intercepts and random coefficients, (b) contextual effects analyses with manifest or latent group means, (c) multiple-group regression models, (d) models with interaction or polynomial effects, and (e) post-hoc probing of interaction effects with Bayesian estimates of conditional effects, among other things. Blimp scripts and analysis examples for M*plus*, R, SAS, SPSS, and Stata are available at www.appliedmissingdata.com.

New Features in Version 2.2

- New `WALDTEST` command to perform Bayesian Wald tests

Major Features

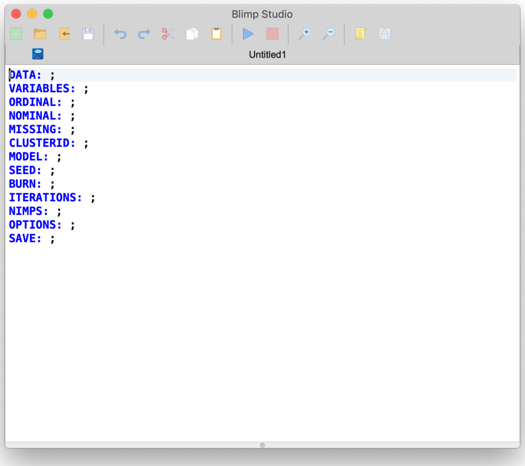
- Simplified scripting language and redesigned output
- Graphical interface with automatic updates when new features become available
- Graphical engine that creates trace plots for all model parameters
- Bayesian estimation of single-level, multilevel (up to three levels), and multiple group regression models with complete or incomplete data.
- Posterior summaries of all model parameters from Bayesian estimation (posterior mean, median, standard deviation, and credible interval).
- Single-level and multilevel regression measures of R-squared (Rights & Sterba, 2018).
- Bayesian estimation for interactive and non-linear effects with missing data.
- Bayesian estimation with grand mean centering (all models) and group mean centering (two- and three-level models).
- Post-hoc probing of interaction effects with continuous or categorical moderators.
- Bayesian estimation of conditional effects (simple slopes) in regression models with interaction effects.
- Incomplete binary, ordinal, or nominal predictor variables.
- Discrete and latent imputations for binary, ordinal, and nominal variables.
- FCS or Bayesian estimation with level-2 and level-3 cluster means modeled as latent variables.
- Contextual effects models with latent group means or manifest group means.
- Ability to specify interactions with latent group means.
- Various algorithmic and interface enhancements (e.g., random starting values, options for saving various estimates and output).

2 Quick Start Guide

This chapter provides a brief tutorial to help users quickly begin using the Blimp command language. In particular, this section gives a broad overview of Blimp's three main functions: multiple imputation with FCS, model-based multiple imputation, and Bayesian estimation of single-level and multilevel regression models. Each of these procedures has a variety of different options, and the examples in this chapter are designed to get users up and running with prototypical applications.

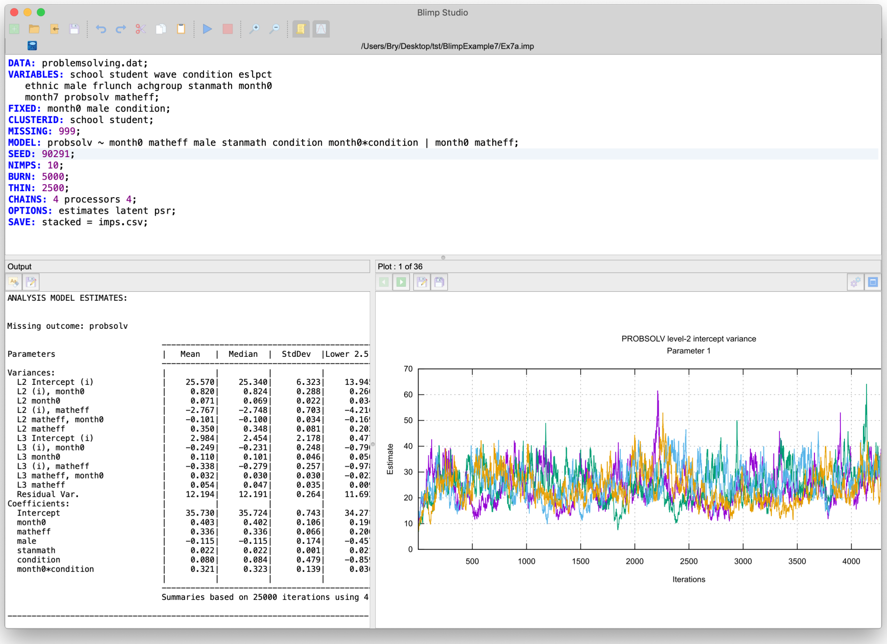
Working in Blimp

Blimp scripts consist of a small handful of commands, many of which will be familiar to users of statistical software packages (e.g., a DATA command that points to the input data set, a VARIABLES command that lists the variable names, etc.). Blimp scripts can be executed via command line arguments in macOS and Windows, but the easiest way to interact with the program is through Blimp Studio graphical user interface. More information about Blimp Studio is found with the examples in Chapter 7. For the purposes of this chapter, we need to simply spawn a blank syntax windows. To do so, choose `File` then `New` from the main application pull-down window. A new scripting window will open that is pre-populated with most of the major commands for multiple imputation. A Bayesian analysis requires only a couple minor modifications to the script. An empty Blimp syntax editor is shown below.



```
DATA: ;
VARIABLES: ;
ORDINAL: ;
NOMINAL: ;
MISSING: ;
CLUSTERID: ;
MODEL: ;
SEED: ;
BURN: ;
ITERATIONS: ;
NIMPS: ;
OPTIONS: ;
SAVE: ;
```

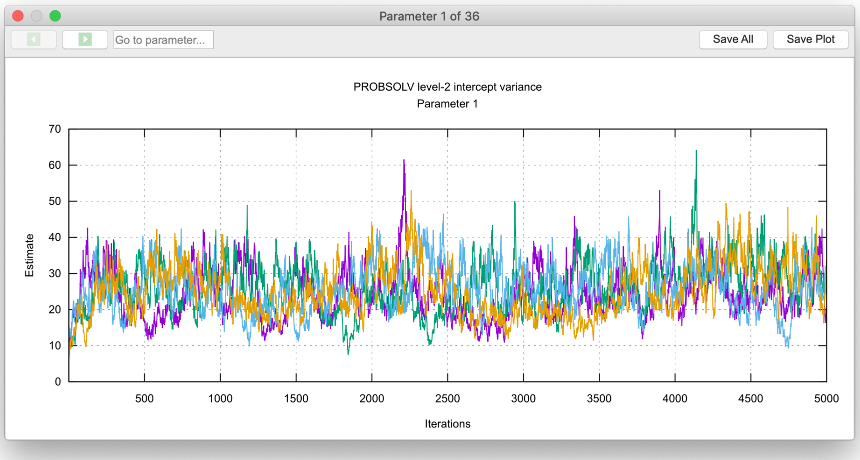
After completing a script, the script is ran by going to the `Run` menu and selecting `Run` or by clicking the blue Run icon on the menu bar. The output window will appear in the bottom pain which contains a variety of information about the script (e.g., summary of the variables, algorithmic options), diagnostic information, and possibly analysis results.



```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct
   ethnic male frlunch achgroup stanmath month0
   month7 probsolv matheff;
FIXED: month0 male condition;
CLUSTERID: school student;
MISSING: 999;
MODEL: probsolv ~ month0 matheff male stanmath condition month0*condition | month0 matheff;
SEED: 90291;
NIMPS: 10;
BURN: 5000;
THIN: 2500;
CHAINS: 4 processors 4;
OPTIONS: estimates latent psr;
SAVE: stacked = imps.csv;
```

```
ANALYSIS MODEL ESTIMATES:


Missing outcome: probsolv


                                ----------------------------------------
Parameters                      |   Mean   |  Median  |  StdDev  |Lower 2.5
                                ----------------------------------------
Variances:                      |          |          |          |
  L2 Intercept (i)              |    25.570|    25.340|     6.323|    13.94
  L2 (i), month0                |     0.820|     0.824|     0.288|     0.26
  L2 month0                     |     0.071|     0.069|     0.022|     0.03
  L2 (i), matheff               |    -2.767|    -2.748|     0.703|    -4.21
  L2 matheff, month0            |    -0.101|    -0.100|     0.034|    -0.16
  L2 matheff                    |     0.350|     0.348|     0.081|     0.20
  L3 Intercept (i)              |     2.984|     2.454|     2.178|     0.47
  L3 (i), month0                |    -0.249|    -0.231|     0.248|    -0.79
  L3 month0                     |     0.110|     0.101|     0.046|     0.05
  L3 (i), matheff               |    -0.338|    -0.279|     0.257|    -0.97
  L3 matheff, month0            |     0.032|     0.030|     0.030|    -0.02
  L3 matheff                    |     0.054|     0.047|     0.035|     0.00
  Residual Var.                 |    12.194|    12.191|     0.264|    11.69
Coefficients:                   |          |          |          |
  Intercept                     |    35.730|    35.724|     0.743|    34.27
  month0                        |     0.403|     0.402|     0.106|     0.19
  matheff                       |     0.336|     0.336|     0.066|     0.20
  male                          |    -0.115|    -0.115|     0.174|    -0.45
  stanmath                      |     0.022|     0.022|     0.001|     0.02
  condition                     |     0.080|     0.084|     0.479|    -0.85
  month0*condition              |     0.321|     0.323|     0.139|     0.03
                                |          |          |          |
                                ----------------------------------------
                                Summaries based on 25000 iterations using 4

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

Blimp Studio has a graphics engine that automatically creates trace plots of the estimated parameters during the burn-in period. The plot window opens automatically at the completion of a job. The window can be resized by dragging the vertical or horizontal bars separating the syntax, output, and plot panes. The plot window can be hidden from view by clicking the normal distribution icon on the toolbar at the top of the application window. Clicking the right or left green arrow buttons pages through the graphs one at a time. Clicking the blue popup button in the upper right corner of the plot window spawns a dedicated plot viewer window. Plots can be saved individuals, or the entire set can be saved by selecting `Save All`.



Having provided a quick tour of the Blimp Studio, the remainder of the chapter illustrates the Blimp command language for three prototypical applications. Example 2.1 illustrates general multiple imputation with FCS, Example 2.2 depicts model-based multiple imputation for a model with an interaction effect, and Example 2.3 illustrates Bayesian estimation with centering and conditional effects (simple slopes). All examples are illustrated in the context of a two-level data structure, but minor modifications give single-level or three-level versions of the applications.

Example 2.1: Fully Conditional Specification (FCS) Imputation

Blimp's FCS routine parallels the MICE (Multiple Imputation by Chained Equations) algorithm popularized by Stef van Buuren and colleagues. FCS is appropriate for a broad class of univariate and multivariate applications, perhaps with the goal of performing a variety of different statistical analyses on the same set of imputed data. FCS is not recommended for models with random slopes, interactions, polynomials or other non-linear terms. The example below depicts multiple imputation for a two-level model with random intercepts. As noted elsewhere, Blimp uses latent group means (i.e., random effects, random intercepts) to preserve between-cluster associations in multilevel model.

Example 2.1: Quick Start Example for Fully Conditional Specification

```
DATA: quickstart.dat;
VARIABLES: id av y x1 x2 x3 w1 w2;
ORDINAL: av x1 x3;
NOMINAL: w2;
CLUSTERID: id;
MISSING: 999;
FCS: y2 x1 x2 w1 w2;
SEED: 90291;
CHAINS: 4 processors 4;
BURN: 2000;
THIN: 500;
NIMPS: 10;
OPTIONS: psr latent;
SAVE: separate = imps*.dat;
```

Blimp commands can be entered in Blimp Studio as shown above or in a plain text file (raw ASCII format) with a '.imp' extension. 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), (e) a dash can be used to specify a range of variables with the same prefix and/or suffix across a range of numbers (e.g., `x1-x3` is the same as `x1 x2 x3`, `x1a-x3a` is the same as `x1a x2a x3a`, etc.), and (f) the number sign (`#`) is used to create a comment line that Blimp ignores.

The `DATA`, `VARIABLES`, `ORDINAL`, `NOMINAL`, and `MISSING` commands specify features of the raw data file. The `DATA` command specifies the input data set. The example depicts a situation where the Blimp script file is located in the same directory as the data set, in which case no file path is needed. If the data file is in a different location, `DATA` command requires a full file path to the input data set. When the file path includes spaces, the file path should not be enclosed in quotations. The input data set must be saved as a CSV (comma separated values) format or a whitespace (including tab) delimited DAT file. Note that some statistical software packages (e.g., SPSS) that save text files in Unicode format embed hidden characters that cause the import to fail. This can typically be changed in the program's preferences (e.g., in SPSS, set the output text format to Locale). The `VARIABLES` command specifies the variables in the input file. The variable names can be up to 10 characters (longer variable names might be truncated in the output), must start with an alphanumeric character, and should not contain periods or other special punctuation marks. A dash can be used to specify a range of variables with the same prefix and/or suffix across a range of numbers (e.g., `x1-x3` is the same as `x1 x2 x3`, `x1a-x3a` is the same as `x1a x2a x3a`, etc.), but dashes are limited to this case. All missing values must be assigned the same numeric code, the value of which is specified via the `MISSING` command.

The `CLUSTERID` command specifies cluster-level identifier variables needed for multilevel imputation or analysis. 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. For single-level imputation, simply omit the `CLUSTERID` command. Listing a cluster identifier automatically introduces random intercepts (i.e., latent group means) for level-1 variables listed on the `FCS` line (the same is true for level-2 variables in a three-level model). As noted elsewhere, Blimp uses latent group means to preserve between-cluster associations in a multilevel model.

The `ORDINAL` and `NOMINAL` commands designate categorical variables (unless otherwise specified, Blimp assumes a normal distribution for all variables). As mentioned elsewhere, Blimp uses a latent variable formulation for categorical variables (probit regression). To maximize computational speed, we recommend specifying binary variables on the `ORDINAL` line, although the underlying statistical model invoked by the `NOMINAL` command is equivalent in this case. Nominal variables must be represented as a single variable with numeric codes. Complete variables listed on the `NOMINAL` line are automatically recoded into a set of dummy codes during imputation. Example 2.1 depicts an imputation model with three ordinal variables (`av`, `x1`, and `x3`) and one nominal variables (`w2`).

The `FCS` command specifies the variables to be used by the imputation algorithm. The FCS algorithm cycles through incomplete variables one at a time, imputing each variable from a regression equation that conditions on all other variables listed on the `FCS` line. At a minimum, the `FCS` statement 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. FCS is broadly applicable to multilevel models with random intercepts. Blimp can also accommodate analyses with interaction effects, polynomial terms, and random coefficients, but the fully Bayesian model-based imputation via the `MODEL` is recommended for these cases.

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 number of iterations between each saved data set. In Example 2.1, imputed data sets are saved after every 500$^{\text{th}}$ iteration. The `NIMPS` command specifies the number of imputations desired. The `SEED` command specifies the seed value for Blimp's pseudo random number generator. The `OPTIONS` command specifies various miscellaneous options, such as prior distributions for variance components, and the method for modeling between-cluster associations in a multilevel model (e.g., the `latent` keyword specifies latent cluster means, or random effects, at the between-group level). Chapter 3 presents more information about the specific options and their defaults.

Finally, the `SAVE` command specifies names for the imputed data set(s). When no file path is given, imputations are saved in the same location as the Blimp script. Using a full file path directs the imputations to a specific location. Example 2.1 saves each imputed data set to a separate file in the same directory as the .imp script. This specification requires an asterisk in the file name, and this asterisk is subsequently replaced by a numeric value (e.g., imps1.dat, imps2.dat, etc.). The separate-file specification is needed for analyzing imputations in M*plus*, and Blimp also creates the necessary input file containing the data set names. Use the `stacked` keyword when analyzing imputations with R, SAS, and SPSS, and use the `stacked0` keyword for analysis in Stata.

Example 2.2: Fully Bayesian Model-Based Imputation (MBI)

The FCS algorithm cycles through incomplete variables one at a time, making no reference to a variable's role in the subsequent analysis. In contrast, MBI generates imputations that are tailored to a specific analysis model. MBI requires the user to specify the analysis model for the outcome variable, and Blimp automatically builds separate models for the covariates. Non-linear or interactive terms appear in the analysis model but not in the covariate model. MBI uses the model-implied distribution from the substantive analysis to impute incomplete outcomes, and it uses a Metropolis sampling step to generate imputations for covariates. These imputations condition on other covariates and the analysis model. Example 2.2 below illustrates a prototypical MBI setup for a two-level random coefficient with an interaction term.

Example 2.2: Quick Start Example for Fully Bayesian Model-Based Imputation

```
DATA: quickstart.dat;
VARIABLES: id av y x1 x2 x3 w1 w2;
ORDINAL: av x1 x3;
NOMINAL: w2;
CLUSTERID: id;
MISSING: 999;
MODEL: y ~  x1 x2 w1 x2*w1 w2 | x2;
SEED: 90291;
CHAINS: 4 processors 4;
BURN: 2000;
THIN: 500;
NIMPS: 10;
OPTIONS: psr latent;
SAVE: separate = imps*.dat;
```

The MBI syntax is nearly identical to that of FCS, except a `MODEL` command replaces the the `FCS` command. The model command specifies the outcome variable to the left of the tilde, and all covariates and their interactions (or polynomial terms) are listed to the right of the tilde. To illustrate, the syntax in Example 2.2 includes an interaction between `x2` and `w1`, which is specified by joining these two variables with an asterisk (i.e., `x2*w1`). The `w2` variable on the `NOMINAL` line (a level-2 categorical variable) is automatically converted to dummy codes in the analysis model, with the first (lowest) code serving as the reference category. Importantly, the interaction is not imputed directly, nor is it computed passively (e.g., by imputing lower-order terms and subsequently forming the product from the imputed values). Rather, a Metropolis sampling step selects imputations that are consistent with an analysis model that includes the product. Finally, random coefficients are listed to the right of the vertical pipe. To illustrate, the syntax in Example 2.2 includes a random coefficient for `x2`. Descriptions of this procedure are found in Enders, Du, and Keller (2019), and Chapter 3 gives additional details on interactive effects. Whenever possible, we recommend model-based imputation for analyses that include interactive or polynomial terms and/or random slopes, as simulation and analytic work suggests that the procedure is far superior to FCS in these cases. In many other scenarios (e.g., additive models), we would usually expect similar performance from FCS and MBI. Note that product or polynomial terms specified on the `MODEL` command do not appear in the output data files. These variables must be computed prior to analysis.

Example 2.3: Bayesian Estimation

The previous example illustrates MBI for multiple imputation, but the procedure can also generate Bayesian estimates of the analysis model parameters. That is, Blimp can be used as a primary analytic tool without saving imputations for further analysis. Example 2.3 illustrates a prototypical Bayesian analysis with centering and post-hoc probing with conditional effect estimates. The Bayesian estimation script is similar to that of MBI with a few exceptions. First, because imputations are not the primary interest, the `NIMPS`, `THIN`, and `SAVE` commands are not needed (imputations are still created at each MCMC iteration, but they are not saved unless requested). Instead, `ITERATIONS` is used to specify the total-number of post-burn-in iterations, or the number of iterations used to generate posterior summaries. In the example above, the `CHAINS` command specifies four MCMC chains. Following the burn-in period, each chain runs for 2,500 iterations (i.e., the 10,000 iterations are spread across four chains). Second, the `estimates` keyword on the `OPTIONS` line generates summaries of the parameter estimates, including the posterior mean, median, standard deviation, and credible intervals.

Example 2.3: Quick Start Example for Fully Bayesian Estimation

```
DATA: quickstart.dat;
VARIABLES: id av y x1 x2 x3 w1 w2;
ORDINAL: av x1 x3;
NOMINAL: w2;
CLUSTERID: id;
MISSING: 999;
MODEL: y ~   x1 x2 w1 x2*w1 w2 | x2;
CENTERING: grandmean = x1 w1, groupmean = x2;
SIMPLE: x2 | w1;
SEED: 90291;
CHAINS: 4 processors 4;
BURN: 2000;
ITERATIONS: 10000;
OPTIONS: psr estimates latent;
```

Example 2.3 illustrates two useful features that would not be necessary in a multiple imputation application. First, the `CENTERING` command specifies grand mean centering or group mean centering for the predictor variables (group mean centering is only available in two- and three-level models). This command has no bearing on imputations, which are always generated and saved on the original metrics. Second, the `SIMPLE` command produces Bayesian estimates of conditional effects from the interaction (used to probe the interaction). At each MCMC iteration, conditional effects are computed by applying an appropriate contrast vector to the current coefficients from analysis specified on the `MODEL` line. These additional parameters are thus functions of the model parameters, each with its own posterior distribution. The `x2 | w1` specification requests conditional effects of the focal predictor `x2` at different levels of the moderator variable `w1`. With a continuous moderator, conditional effects are computed at the mean and at one and two standard deviations above and below the mean. Conditional effects for specific constants of interest can also be specified. For nominal moderator variables, conditional effects are reported for all groups.

Running Blimp

As noted previously, Blimp is executed by selecting `Run` within Blimp Studio. For power users who require the flexibility, a terminal emulator (e.g., Terminal.app in macOS, cmd.exe in Windows) can used to run Blimp from the command-line. See Chapter 4 for more information. The command-line specification for macOS is as follows:

```
$ /Applications/Blimp/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.

Blimp Output

As illustrated earlier in this chapter, executing Blimp launches an output window in the GUI. A header that provides the version number along with award and contact information is displayed at the top of the output. An example header is shown below.

```
                                   Blimp
                                   2.1.26

            Developed by Craig K. Enders, Brian T. Keller, and Han Du.
                       Blimp was developed with funding from
                         Institute of Educational Sciences
                         awards R305D150056 and R305D190002.
               Craig K. Enders, P.I. Email: cenders@psych.ucla.edu
        Programming by Brian T. Keller, Co-P.I. Email: bkeller2@ucla.edu
                     Han Du, Co-P.I. Email: hdu@psych.ucla.edu
                       There is 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. The previous scripts do not specify prior distributions for variance components or other special algorithm options. Appropriate default settings are invoked in this case.

```
ALGORITHMIC OPTIONS SPECIFIED:

  Imputation method:                  Fully Bayesian model-based
  MCMC algorithm:                     Full conditional Metropolis sampler
  Between-cluster imputation model:  Latent cluster means (LATENT)
  Residual variance structure:        Homogeneous level-1 variance (HOV)
  Prior for covariance matrices:      Zero matrix, df = -(p + 1) (PRIOR2)
  Prior for residual variance:        Zero sum of squares, df = -2 (PRIOR2)
  Prior for covariate model:          Unit sum of squares, df = 2 (XPRIOR1)
  Diagnostics:                        Potential scale reduction (PSR)
  Chain Starting Values:              Random starting values
```

Following the algorithmic options, Blimp displays a summary of the data and variables used in the analysis or imputation model.

```
DATA INFORMATION:

  Sample Size:            630
  Level-2 Clusters:       105
  Missing Data Code:      999.000
```

```
VARIABLES IN IMPUTATION MODEL:

  Level-2 identifier:     id
  Complete ordinal:       x1
  Incomplete continuous:  x2 w1
  Incomplete nominal:     w2
  Missing outcome:        y2 (continuous)
  CENTERED PREDICTORS
    Grand Mean Centered:  x1 w1
    Group Mean Centered:  x2
  Product Terms:          x2*w1
  Random Coefficients
    Level-2:              x2
```

In this example, `w2` is a six-category nominal variable. Because this variable is listed as a predictor on the `MODEL` line, Blimp automatically recodes the variable into five dummy codes for the analysis model, where the first group (the lowest numeric code) serves as the reference. The `MODEL` line generally invokes a set of regression models that are used to preserve associations among the predictors (these are not specified by the user). Categorical variables are treated as latent variables in the models that link the predictors to one another. Finally, in an FCS routine, nominal variables can be modeled as dummy codes or as underlying normal latent variables (the default).

Next, the output displays the progress of the MCMC algorithm. The output below corresponds to the Bayesian analysis from Example 2.3, but the printed information varies somewhat depending on the analysis (e.g., additional information and timestamps are printed if imputations are saved).

```
CHAIN HISTORY:

Chain 1 online with seed of 158875
Chain 2 online with seed of 150835
Chain 3 online with seed of 98783
Chain 4 online with seed of 104168
    Chain 2 completed burn-in on Sun Aug 25 14:55:57 2019
    Chain 4 completed burn-in on Sun Aug 25 14:55:57 2019
    Chain 1 completed burn-in on Sun Aug 25 14:55:57 2019
    Chain 3 completed burn-in on Sun Aug 25 14:55:57 2019
    Chain 2 completed iteration 1250 on Sun Aug 25 14:56:01 2019
    Chain 4 completed iteration 1250 on Sun Aug 25 14:56:01 2019
    Chain 1 completed iteration 1250 on Sun Aug 25 14:56:01 2019
    Chain 3 completed iteration 1250 on Sun Aug 25 14:56:01 2019
    Chain 2 completed iteration 2500 on Sun Aug 25 14:56:02 2019
    Chain 4 completed iteration 2500 on Sun Aug 25 14:56:02 2019
    Chain 1 completed iteration 2500 on Sun Aug 25 14:56:02 2019
    Chain 3 completed iteration 2500 on Sun Aug 25 14:56:02 2019
Chain 2 finished on Sun Aug 25 14:56:02 2019
Chain 4 finished on Sun Aug 25 14:56:02 2019
Chain 1 finished on Sun Aug 25 14:56:02 2019
Chain 3 finished on Sun Aug 25 14:56:02 2019
```

If imputations are requested on the `SAVE` command, Blimp prints the order of the variables in the imputed 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, and variables are saved in the same order as they appear on the `VARIABLES` command.

```
VARIABLE ORDER IN SAVED DATA:

   id av y x1 x2 x3 w1 w2
```

When imputations are stacked in a single file (e.g., `SAVE: stacked = imps.dat`), Blimp adds a variable, `imp#`, to the first column of the data set that indexes the imputed data sets.

```
VARIABLE ORDER IN SAVED DATA:

   imp# id av y x1 x2 x3 w1 w2
```

In a Bayesian analysis, the parameter estimates rather than the imputations are the focal interest. Whenever the `MODEL` command is invoked, parameter estimates for the specified model are displayed. This summary includes the posterior mean, median, standard deviation, and credible intervals. The analysis output from Example 2.3 is shown below. Notice that `w2` is listed several times in the coefficients table with a hash and numeric indices. Blimp automatically converts a nominal predictor variable into a set of dummy codes for the analysis model. The lowest (first) numeric code serves as the reference, and dummy codes are created for all remaining groups versus the reference. In this example, `w2` is a six-category nominal variable, and `w2#2`, `w2#3`, . . . , `w2#6` are dummy codes comparing groups two through six to the reference group (1).

```
ANALYSIS MODEL ESTIMATES:


Missing outcome: y

Grand Mean Centered: x1 w1
Group Mean Centered: x2

                   -------------------------------------------------------
Parameters         |   Mean    |  Median   |  StdDev  |Lower 2.5 |Upper 97.5|
                   -------------------------------------------------------
Variances:         |           |           |           |          |          |
  L2 Intercept (i) |      1.356|      1.283|      0.592|     0.403|     2.678|
  L2 (i), x2       |     -0.222|     -0.217|      0.155|    -0.545|     0.055|
  L2 x2            |      0.169|      0.160|      0.083|     0.036|     0.356|
  Residual Var.    |     12.493|     12.450|      0.884|    10.877|    14.326|
Coefficients:      |           |           |           |          |          |
  Intercept        |     29.246|     29.250|      0.543|    28.181|    30.315|
  x1               |      1.838|      1.839|      0.311|     1.224|     2.444|
  x2               |      0.625|      0.626|      0.080|     0.464|     0.781|
  w1               |      0.264|      0.264|      0.055|     0.154|     0.371|
  w2#2             |     -0.772|     -0.762|      0.674|    -2.110|     0.538|
  w2#3             |     -1.110|     -1.111|      0.645|    -2.361|     0.176|
  w2#4             |      0.267|      0.268|      0.808|    -1.301|     1.875|
  w2#5             |     -0.962|     -0.964|      0.786|    -2.483|     0.613|
  w2#6             |     -0.679|     -0.689|      0.826|    -2.282|     0.991|
  x2*w1            |      0.056|      0.056|      0.019|     0.020|     0.093|
                   |           |           |           |          |          |
                   -------------------------------------------------------
                   Summaries based on 10000 iterations using 4 chains
```

Conditional effects from Example 2.3 are shown below. The analysis requests the conditional effect of x1 at different levels of the moderator w1. For continuous moderators, Blimp by default reports conditional effects at zero and at one and two standard deviations above and below the zero value. We highly recommend mean centering the focal predictor and moderator such that zero represents the mean value.

```
CONDITIONAL EFFECTS ANALYSIS:


Missing outcome: y

Grand Mean Centered: x1 w1
Group Mean Centered: x2

                    --------------------------------------------------------
Conditional Effects |   Mean    |  Median   |  StdDev   |Lower 2.5  |Upper 97.5|
                    --------------------------------------------------------
  x2 | w1 @ +2 SD   |           |           |           |           |          |
    Intercept       |    31.431|    31.416|      0.678|     30.092|    32.835|
    Slope           |     1.109|     1.106|      0.179|      0.767|     1.467|
                    |           |           |           |           |          |
  x2 | w1 @ +1 SD   |           |           |           |           |          |
    Intercept       |    30.329|    30.321|      0.560|     29.216|    31.467|
    Slope           |     0.866|     0.866|      0.112|      0.652|     1.092|
                    |           |           |           |           |          |
  x2 | w1 @ 0       |           |           |           |           |          |
    Intercept       |    29.227|    29.223|      0.531|     28.186|    30.279|
    Slope           |     0.623|     0.625|      0.083|      0.456|     0.788|
                    |           |           |           |           |          |
  x2 | w1 @ -1 SD   |           |           |           |           |          |
    Intercept       |    28.125|    28.120|      0.604|     26.919|    29.324|
    Slope           |     0.381|     0.382|      0.122|      0.127|     0.611|
                    |           |           |           |           |          |
  x2 | w1 @ -2 SD   |           |           |           |           |          |
    Intercept       |    27.023|    27.021|      0.751|     25.571|    28.483|
    Slope           |     0.138|     0.139|      0.192|     -0.249|     0.494|
                    |           |           |           |           |          |
                    --------------------------------------------------------
                    Summaries based on 10000 iterations using 4 chains
```

3 Blimp Command Language

This Chapter gives a more detailed account of the Blimp command language illustrated previously in the Chapter 2. 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 below.

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), (e) a dash can be used to specify a range of variables with the same prefix and/or suffix across a range of numbers (e.g., `x1-x3` is the same as `x1 x2 x3`, `x1a-x3a` is the same as `x1a x2a x3a`, etc.), and (f) the number sign (`#`) is used to create a comment line that Blimp ignores.

Reading Data

`DATA`. The `DATA` command specifies the input data set. No file path is needed if the Blimp script (the .imp file) is located in the same directory as the data set. If the data file is in a different location, `DATA` command requires a full file path to the input data set. When the file path includes spaces, the file path should not be enclosed in quotations. The input data set must be saved as a CSV (comma separated values) format or a whitespace (including tab) delimited DAT file. It is important to note that Blimp currently accepts only numerical values in the data sets. Some statistical software packages that save text files in Unicode format (e.g., SPSS) embed hidden characters that cause the import to fail. This can typically be changed in the program's preferences (e.g., in SPSS, set the output text format to Locale). These packages may also save variable names as column headers, which Blimp also prohibits.

`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. A dash can be used to specify a range of variables with the same prefix and/or suffix across a range of numbers (e.g., `x1-x3` is the same as `x1 x2 x3`, `x1a-x3a` is the same as `x1a x2a x3a`, etc.), but dashes are limited to this case.

`ORDINAL`. The `ORDINAL` command specifies incomplete ordinal (including binary) variables. For computational efficiency, we recommend listing binary variables on the `ORDINAL` line, but these variables could also be treated as nominal. A dash can be used to specify a range of variables with the same prefix and/or suffix across a range of numbers (e.g., `x1-x3` is the same as `x1 x2 x3`, `x1a-x3a` is the same as `x1a x2a x3a`, etc.), but dashes are limited to this case.

`NOMINAL`. The `NOMINAL` command specifies nominal variables, incomplete or complete. Nominal variables must be represented as a single variable with numeric codes. Nominal variables are automatically recoded into a set of dummy codes during imputation when needed (e.g., when listed as a predictor on the `MODEL` line). In some situations (e.g., FCS with the `latent` keyword), nominal variables are represented as latent variables rather than as dummy codes. A dash can be used to specify a range of variables with the same prefix and/or suffix across a range of numbers (e.g., `x1-x3` is the same as `x1 x2 x3`, `x1a-x3a` is the same as `x1a x2a x3a`, etc.), but dashes are limited to this case.

`CLUSTERID`. The `CLUSTERID` command specifies cluster-level identifier variables needed for multilevel imputation or analysis. 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. For single-level imputation, simply omit the `CLUSTERID` command.

`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., 999).

The FCS Command

Fully conditional specification is one of two major imputation algorithms in Blimp. The FCS algorithm cycles through incomplete variables one at a time, imputing each variable from a regression equation that conditions on all other variables listed on the `FCS` line. This algorithm makes no distinction between the outcome and covariates in the subsequent analysis model. At a minimum, the `FCS` command should include all variables and effects of interest in the analysis model(s), but the list may also include additional auxiliary variables. Blimp's FCS procedure parallels the MICE (Multiple Imputation by Chained Equations) algorithm popularized by Stef van Buuren and colleagues. FCS is appropriate for a broad class of univariate and multivariate applications. Blimp's FCS routine uses a latent variable formulation for categorical variables (i.e., ordered probit and multinomial probit models) to impute incomplete binary, ordinal, and nominal variables. In multilevel data sets, between-cluster associations can be modeled with latent group means (random intercepts) or with the group means of the manifest variables. The FCS routine offers a number of specialized options, including multiple-group imputation and heterogeneous level-1 variance structures (two-level models).

The MODEL Command

The `MODEL` command invokes Bayesian estimation and model-based multiple imputation. This procedure is designed for regression models with interaction effects, polynomial terms, and/or random coefficients, but it is also appropriate for additive regression models as well. The outcome variable may be binary, ordinal, or continuous (normal). Explanatory variables may be binary, ordinal, nominal, or continuous (normal). Categorical variables are modeled as underlying normal latent variables, and are converted to discrete values when necessary (e.g., when represented as a dummy code in the analysis model). Interactions can include any combination of categorical and continuous predictors.

Additive Regression Models

The `MODEL` command specifies a regression model that corresponds to the substantive analysis. The outcome variable is listed to the left of the tilde, and all covariates are listed to the right of the tilde. Blimp automatically determines the level at which a variable is measured in a multilevel data set, so the user need only provide a basic model specification. The `MODEL` statement for a multiple regression model with an outcome and two predictors is shown below.

```
MODEL: y ~ x1 x2;
```

If the `CLUSTERID` command includes identifier variables for a multilevel analysis, then random intercepts are automatically added to the analysis model at all higher levels. By default, these group (cluster) means are modeled as latent variables (i.e., random effects). For example, in a two-level model, the level-2 association between `x1` and `x2` is modeled with latent group means (i.e., random intercepts). The means can also be modeled as arithmetic or manifest group averages using the `manifest` keyword on the `OPTIONS` line.

Specifying Random Slopes

Random coefficients are specified by listing random predictors to the right of the optional vertical pipe on the `MODEL` statement. This option is only available with a multilevel model with cluster-level identifiers specified on the `CLUSTERID` line.

```
MODEL: y ~ x1 x2 w1 w2 | x1 x2;
```

Blimp automatically includes random slopes at all higher levels of the data hierarchy. For example, suppose that `x1` and `x2` are level-1 and level-2 predictors in a three-level model. The specification above would include a random slope for `x1` at level-2 and level-3, and it would include a random slope for `x2` at level-3.

Specifying Interaction and Polynomial Terms

Traditional imputation schemes typically force the user to impute incomplete product or polynomial terms (e.g., the so-called "just another variable" approach). This strategy is not ideal because it imposes incorrect distributional assumptions on the product term and is prone to bias. Fully Bayesian (model-based) estimation offers an alternative approach that appears to be far superior. Rather than imputing the product directly, a Metropolis sampling step selects imputations that are consistent with an analysis model that includes the non-linear terms listed on the on the `MODEL` line (including random coefficients).

To illustrate the specification of a product term, consider a two-level analysis that includes a random slope for a level-1 predictor `x1` and a cross-level interaction involving `x1` and a level-2 predictor `w1`.

```
MODEL: y ~ x1 x2 w1 w2 x1*w1 | x1;
```

Notice that the variables forming the interaction are joined with an asterisk with no spaces separating them i.e. `x1*w1`. This specification extends to accommodate higher-order interaction terms as well. For example, consider a single-level analysis with a three-way interaction and all corresponding lower-order terms. The `MODEL` statement would appear as follows.

```
MODEL: y ~ x z w x*z x*w z*w x*z*w;
```

Polynomial terms are specified using a similar convention. For example, an analysis that expresses the outcome variable as a quadratic function of a predictor would have the following `MODEL` command.

```
MODEL: y ~ x x*x;
```

Importantly, interaction and polynomial terms are not imputed directly, nor are they computed passively (e.g., by imputing lower-order terms and subsequently forming the product from the imputed values). Similarly, these special variables do not appear in the output data sets and must be computed prior to analysis. Descriptions of the Bayesian estimation procedure for interactions are found in Enders et al. (2019) and Zhang and Wang (2016). Examples of interactive effects along with M*plus*, SAS, SPSS, and Stata analysis scripts are available from www.appliedmissingdata.com.

Adding Group Means (Contextual Effects)

In a multilevel model, group means can be added as between-cluster predictors at level-2 or level-3 of the data hierarchy. A common application is a contextual effects model that examines whether regression slopes differ across levels of the data hierarchy. Group means are added by attaching `.mean` to a variable on the `MODEL` line. The following code adds the `x1` group means to the analysis model.

```
MODEL: y ~ x1 x1.mean x2 w1 w2 | x1 x2;
```

By default, Blimp models group means for continuous variables as normally distributed latent variables (i.e., random effects). Also, group means are automatically included at all higher levels. For example, if `x1` is a level-1 predictor in a three-level model, then the above specification would add level-2 and level-3 latent group means to the model. If `x1` is a level-1 predictor in a two-level model, then the above specification would add the group means only at the second level. The latent group means of level-2 variables can also be included as predictors in the three-level model.

The CENTERING Command

The `CENTERING` command is used in conjunction with Bayesian estimation. This option has no bearing on imputations generated by the `SAVE` command. Thus, centering would only be invoked when using Blimp as the primary analytic tool. Blimp's default estimation routine models the multivariate distribution of the predictor variables (i.e., all predictors are random variables rather than fixed by design, as in an ordinary least squares analysis). This specification treats grand means and group means as random variables to be estimated at each MCMC iteration (e.g., level-2 and level-3 cluster means are normally distributed latent variables, or random effects). The `CENTERING` command invokes a Metropolis-Hastings step that draws grand or group means from a distribution that accounts for their presence and role in the analysis model. Product terms specified on the `MODEL` line automatically reflect the centering method.

Two forms of centering are available: grand mean centering and group mean centering (the latter is for multilevel models). To illustrate, suppose that `x1` and `w2` are level-1 and level-2 predictors in a two-level model.

```
MODEL: y ~ x1 x2 w1 w2 x1*w2 | x1;
```

The specification below centers `x2` and `w2` at their grand means, and it centers `x1` at its group means.

```
CENTERING: grandmean = x2 w2, groupmean = x1;
```

Importantly, group mean centering reflects deviations between `x1` scores and the latent group means. Further, group mean centering is always performed by subtracting the latent group means at the next level of the data hierarchy. For example, if the previous analysis was a three-level model, the centering procedure would subtract `x1` scores from the level-2 latent group means. The group means can be included in the analysis model as predictors if desired.

Categorical variables can also be centered. To illustrate, consider a binary variable with a 50/50 split in the categories. Centering a binary dummy code at its grand mean (.50, the proportion of ones) gives an effect-type code with values of -.5 and +.5. As mentioned elsewhere, categorical predictors (binary, ordinal, or nominal) are modeled as underlying normal latent variables. The grand means and group means of these predictors are also modeled on the latent (z-score) metric. Listing categorical variables on the `CENTERING` command invokes a transformation that converts the latent mean to the metric required by the analysis model. For example, the aforementioned dummy variable would represented as an underlying normal latent variable in the model, such that the grand mean controls the proportion of zeros and ones. Centering the binary predictor converts the latent grand mean to a 'manifest' mean that is interpreted as model-implied proportion of ones in the data. Applying centering to nominal variables with three or more categories can be computationally intensive because the latent mean conversion requires Monte Carlo integration at each MCMC step.

The SIMPLE Command

The `SIMPLE` command is used in a Bayesian analysis to request conditional effects (e.g., simple intercepts and simple slopes) from a regression model with an interaction effect. At each MCMC iteration, conditional effects are computed by applying an appropriate contrast vector to the current coefficients from analysis specified on the `MODEL` line. These additional parameters are thus functions of the model parameters, each with its own posterior distribution. To illustrate, consider the following random coefficient model with an interaction effect.

```
MODEL: y ~ x1 x2 w1 w2 x1*w2 | x1;
```

The `x1 | w2` specification requests conditional effects of the focal predictor `x1` at different levels of the moderator variable `w2`.

```
SIMPLE: x1 | w2;
```

With a continuous moderator, Blimp by default reports conditional effects at zero and at one and two standard deviations above and below the zero value. We highly recommend mean centering the focal predictor and moderator such that zero represents the mean value. The exact standard deviation used depends on the moderator's level and potentially on it's centering. For example, suppose that `w2` is a level-2 variable. In this case, the between-cluster standard deviation is used to compute conditional effects. As a second example, suppose `w2` is a level-1 moderator centered at its group means. In this case, centering removes the between-group variation from the moderator, and the within-cluster standard deviation is used. As a final example, suppose `w2` is a level-1 moderator centered at the grand mean. In this case, the total standard deviation that combines within- and between-group variability is used for the pick-a-point approach. For nominal moderator variables, conditional effects are reported for the reference group and for each of the dummy coded groups. Users must specify an exact value for ordinal moderator variables (including binary variables listed as ordinal) in the process described below.

Multiple conditional effects can be separated by commas, as shown below, or `SIMPLE` commands can be used in succession.

```
SIMPLE: x1 | w2, w2 | x1;
```

Conditional effects for specific constants of interest can also be specified. This specification can be used for continuous variables, but it must be used for ordinal variables. For example, suppose `w2` is a binary moderator variable on the `ORDINAL` line with codes of 0 and 1. The conditional effects for this example are as follows.

```
SIMPLE: x1 | w2@0, x1 | w2@1;
```

Finally, for continuous moderators, standard deviation units can also be specified. For example, suppose `w2` is now a continuous moderator variable. One can specify plus and minus one and a half standard deviation units as follows.

```
SIMPLE: x1 | w2@1.5SD, x1 | w2@-1.5SD;
```

The WALDTEST Command

The `WALDTEST` (or `WALD/TEST`) command is used to perform a Bayesian Wald test on a nested model (Asparouhov & Muthén, 2019). The test statistic is computed based on the posterior draws of the full model and is asymptotically equivalent to the maximum likelihood version. To use the `WALDTEST` command, specify the nested model that is to be compared to the full model estimated in the `MODEL` command by excluding predictors or random effects that are to be fixed to zero. Alternatively, one can set these predictors explicitly to zero (or any value) by following the predictor with an @ symbol and a 0 (or any value).

To illustrate the `WALDTEST` command, consider the following random coefficient model with a cross-level interaction effect.

$$y_{ij} = \beta_0 + \beta_1 x_{ij} + \beta_2 w_j + \beta_3 x_{ij} w_j + u_{0_j} + u_{1_j} x_{ij} + e_{ij}$$

The corresponding `MODEL` command is given as follows.

```
MODEL: y ~ x w x*w | x;
```

To compute a Wald test comparing the above model to an unconditional intercept only model, specify the following command:

```
WALDTEST: y ~ 1 | 1;
```

where the `1` represents an intercept. Alternatively, one can give the equivalent and more explicit specification using constarints.

```
WALDTEST: y ~ x@0 w@0 x*w@0 | x@0;
```

The above command will result in a 5 degree of freedom (three constraints on fixed effects, one constraint on the variance, and one constraint on the covariance) χ^2 test.

Commands for Algorithmic Options

`FIXED`. The `FIXED` command is used in conjunction with the `latent` estimation option (described below) to identify predictor variables that should be treated as fixed by design during estimation. This option is particularly useful for regression models with complete categorical predictors. By default, Blimp models the full distribution of the predictors (or the latent variable counterparts, in the case of categorical predictors). This specification can be computationally intensive, particularly when there is no substantive interest in the underlying latent variables. As an example, in a regression model where gender is a complete dummy code, setting this variable to fixed would eliminate the need to estimate its underlying latent scores. Where appropriate, this specification can speed convergence and improve efficiency.

Variables listed on the `FIXED` line will have all their variance assigned to the level at which they are measured and will not have their means and variances estimated. This specification is useful, for example, when a level-1 predictor has no variance at higher levels by design (e.g., a growth curve model where a temporal predictor such as data collection wave is the same for all participants; a repeated measures experiment where the number of stimuli coded 0 and 1 are the same for all individuals). Prior to estimation, Blimp attempts to identify variables that have estimated intraclass correlations close to zero and assign these predictors as fixed, but this command allows users to manually identify such variables. Doing so can mitigate computational problems that arise when level-1 variables have no variation at level-2 or level-3.

`BYGROUP`. The `BYGROUP` command is used to perform estimation or imputation separately for observed subgroups in the data. Among other things, this specification is useful when the analysis model is a multiple group model. Two applications are as follows. First, using `BYGROUP` command in conjunction with `FCS` might be appropriate for psychometric analyses that posit unique covariance matrices or factor structures across a set of demographic groups (e.g., analyses investigating measurement invariance). Second, using `BYGROUP` command in conjunction with `MODEL` estimates a regression model separately at each level of the grouping variable. This might be useful for modeling higher-order (e.g., three-way) interactions where the moderator is a complete categorical variable. Only a single categorical variable is allowed on the `BYGROUP` command, although higher-order associations can be preserved by coding multiple categorical variables into a single variable, sample size permitting. Importantly, the variable listed on the `BYGROUP` command should not be listed on the `ORDINAL`, `NOMINAL`, or `MODEL` lines. Also, multiple MCMC chains (see discussion of the `CHAINS` command) are not allowed when `BYGROUP` is used.

`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 MCMC iterations separating each output data set. For example, a value of 1000 would save 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.

ITERATIONS. The ITERATIONS command is used to specify the number of post-burn-in iterations in a Bayesian analysis. For example, specifying a value of 10,000 would produce 10,000 MCMC cycles and a posterior summary based on 10,000 parameter values. The ITERATIONS command is usually used in lieu of the THIN and NIMPS commands.

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 MCMC chains. The default number of chains is one, unless the psr keyword is specified on the OPTIONS command, in which case two chains are used. When multiple chains are specified, Blimp attempts to distribute the chains across physical cores, resulting in faster computation. To override this default behavior, one can modify the number of processors with the processors keyword, as follows.

```
CHAINS: 10 processors 2;
```

The above line specifies 10 chains using two processors at a time. It is important to note that each chain will have a different seeding value along with different randomly perturbed start values for the variance parameters in the imputation model. The random starts can be turned off by specifying norandomstarts on the OPTIONS line.

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 Chapter 2).

`OPTIONS` Keywords

This section describes the keywords available with the `OPTIONS` command. Note that uppercase keywords in bold typeface are default settings that will be used unless otherwise specified.

`GIBBS`/`MICE`. This option is applicable only to the `manifest` FCS imputation method. Blimp offers two versions of the fully conditional specification algorithm. The `mice` keyword is consistent with the original formulation of fully conditional specification described by van Buuren et al. (2011), where estimates are derived using only those cases with observed data on the variable to be imputed. The `gibbs` keyword triggers a traditional Gibbs sampler described in Bayesian analysis texts whereby parameter estimates are derived from the full imputed data set. The `gibbs` option is the default because it can avoid computational problems that occur when some clusters in a multilevel data set have few cases (e.g., singleton clusters) or very sparse data. With continuous variables, the two options give nearly identical results, although the `gibbs` option tends to converge more slowly.

`LATENT`/`MANIFEST`/`NOCLMEAN`. This option specifies how covariates are modeled in FCS and MBI. The `latent` keyword defines all variables as random variables with a distribution (e.g., normal distribution or normal underlying latent distribution in the case of categorical variables). In single-level models, this specification tends to make very little difference. In this case, complete variables are effectively treated as though they are incomplete in the sense that their means, variances, etc. are estimated at each iteration. In contrast, the `manifest` and `noclmean` keywords treats complete variables as fixed (i.e., their means and variances are not estimated). The key use of this keyword occurs in multilevel models. In this context, the default `latent` keyword invokes a model that uses latent group means (i.e., random effects, random intercepts) to represent the between-cluster parts of the covariates, whereas the `manifest` keyword computes group means as arithmetic averages of the scores within each cluster. Finally, the `noclmean` keyword will not use manifest cluster means in the regression models of variables with two or more levels. Although the procedures often give very similar results, the `latent` keyword is generally preferable because it accommodates unequal cluster sizes, whereas the `manifest` keyword assumes equal group sizes and the `noclmean` assumes there are no contextual effects.

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

`HOV`/`HEV`. In a two-level data set, the `hov` (`homogenous`) keyword specifies a common residual variance for all clusters, which is in line with the standard representation of the multilevel model. 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.

`PRIOR1`/**`PRIOR2`**/`PRIOR3`. This command is used to set the prior distributions for the substantive model parameters in a fully Bayesian (model-based) application. For all regression coefficients in the imputation model, Blimp implements a Jeffreys (uniform) prior distribution, and this prior cannot be modified. The program offers three 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 choice of prior distribution is applied to all variance components in the substantive analysis model. 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 parameters in the covariance matrix. The `prior2` keyword specifies an inverse Wishart prior, $W^{-1}(0, -p-1)$, where p is the number of variance parameters in the covariance matrix. Finally, the `prior3` keyword specifies a Jeffreys prior. The technical details for these prior distributions are given in (Enders et al., 2019). For a fully Bayesian analysis, `prior2` is the default for the substantive analysis model. This command also sets the priors when the `manifest` keyword is used, although we anticipate that most users will opt for the the `latent` estimation method.

`XPRIOR1`/`XPRIOR2`/`XPRIOR3`. This command is used to set the prior distributions for (a) covariates in a fully Bayesian (model-based) application (`latent` commmand only), or (b) FCS imputation with the `latent` estimation method (the default). For all regression coefficients in the imputation model, Blimp implements a Jeffreys (uniform) prior distribution, and this prior cannot be modified. The program offers the same three choices of prior distribution described above. We set `prior1` as the default because it appears to mitigate computational problems that can occur when a level-1 variable has very little between-cluster variation at level-2 (or level-3).

`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 MCMC algorithm. 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 across all parameters. This table includes a numeric label that identifies the parameter with the highest PSR, and the numeric indices are linked to the parameters in the labels file that saved as optional output by the the `SAVE` command. The `nopsr` keyword is the default, meaning that no diagnostic information is printed. See Chapter 6 for more information.

`ESTIMATES`/`NOESTIMATES`. When the `MODEL` command is used for Bayesian estimation or model-based imputation, Blimp prints posterior summaries of the analysis model parameters. Covariate model parameters are not printed, but can be saved to a text file using the `SAVE` command. The `estimates` keyword is not available with `FCS`, although the `SAVE` command can be used to output the posterior summaries of the various imputation models.

The **`SAVE`** Command

The `SAVE` command is used to specify the location to save various outputs (including imputations) in Blimp. A correct `SAVE` command is specified in two parts

```
SAVE: keyword = filepath;
```

where 'keyword' is replaced with one of the keywords specifying a particular object to be saved (e.g., imputations, paremter draws from burn-in or post-burn-in periods, starting values). The 'filepath' term is replaced with the full system file path to where the output will be saved. When no file path is given, objects are saved in the same location as the Blimp script. Using a full file path directs the imputations to a different location. Additionally, multiple keywords can be chained together by separating each statement by a comma.

```
SAVE: keyword1 = filepath1, keyword2 = filepath2;
```

To save imputations there are three different options: `stacked`, `stacked0`, and `separate`. The `stacked` specification stacks imputed data sets in a single file with an identifier variable that indexes the imputations from $m = 1$ to M. This format is ideal for analyzing the imputations in R, SAS, and SPSS, among others. The `stacked0` specification stacks imputed data sets in a single file but additional appends the original incomplete data with an index $m = 0$. This format is ideal for analyzing the imputations in Stata. Finally, the `separate` specification writes imputations as separate files. When saving imputations to separate files, an asterisk (⋆) is required in the file path (e.g., see Example 2.1 in the Chapter 2 of the document). This asterisk will be replaced with an integer in the file name (e.g., specifying myimps*.dat would produce imputed data sets named myimps1.dat, myimps2.dat, etc.). Specifying separate files is required to analyze imputations in M*plus*. Note that the separate specification also generates a listing file that contains the names of the files (this listing file functions as the input file for an M*plus* analysis).

In addition to saving imputed data sets, the `SAVE` command can also be used to save the parameter draws during the burn-in iterations (`burn` keyword), save the parameter draws during the post burn-in iterations (`iterations` keyword), save summaries of the post burn-in parameter draws (`estimates` keyword), save starting values used by Blimp (`starts` keyword), save the key that links numeric labels of parameters given by Blimp (`labels` keyword), and the psr values of all parameters (`psr` keyword). To illustrate how this can be useful, specifying `burn` keyword saves parameters in a format that can be used to create trace plots in R. An R program for creating trace plots from the burn-in output is available from www.appliedmissingdata.com that can then be used to create trace plots for monitoring convergence and mixing. The typical imputation run invokes several regression models (e.g., in the case of MBI, one for the substantive analysis, and one for each covariate), and all model parameters are saved in matrices for subsequent processing in a general-use data analysis package. To understand the structure of these files, it is necessary to save a file that contains numeric labels for all parameters. These labels will be useful for identifying the meaning of each parameter when generating traceplots.

When using a single MCMC chains, the burn-in and post burn-in parameter draws are saved as follows.

```
SAVE: stacked = imps.dat, burn = burn.dat;
```

When using multiple MCMC chains, the burn-in and post burn-in parameter values are saved in separate files, and an asterisk (⋆) is required in the file path, as below. This asterisk will be replaced with an integer in the file name (e.g., specifying burn*.dat would produce data sets named params1.dat, params2.dat, etc.).

```
SAVE: stacked = imps.dat, burn = params⋆.dat;
```

Finally, the `latent` keyword can be used to save imputations of the latent means, random effects from the analysis model, and latent scores for ordinal and nominal variables. The random effects and latent means can be used for diagnostic purposes (e.g., to check for distributional assumptions or omitted variables, as outlined in multilevel texts), and the latent scores can be analyzed as multiple imputations outside of Blimp. For example, a set of categorical indicators could be converted to their underlying normal latent variables, and these latent scores could be used as multiply imputed data in a confirmatory factor analysis to avoid modeling discrete variables.

The latent quantities are saved with each imputed data set, and number of imputations is specified by the `NIMPS` command. The following example stacks the latent data in a single file in the same way as `stacked =` does for imputed data.

```
SAVE: latent = latent.dat;
```

By including an asterisk in the file name, the latent quantities will be saved as separate files. An example of this specification is as follows, where the asterisk is replaced by the

```
SAVE: latent = latent*.dat;
```

The output data file has the necessary identifier variables to merge the latent data with the original data, if desired, as the latent data sets have the same structure (i.e., same number of rows) as the original data.

4 Running Blimp

In Blimp Studio, Blimp is executed by going to the `Run` menu and selecting `Run` or by clicking the blue Run icon on the menu bar. Double-clicking a Blimp syntax file (a text file ending in the .imp extension) will automatically open the file in Blimp Studio. 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 external Monte Carlo computer simulations.

To run Blimp from a command line, type the filepath to the blimp executable included in the installation folder followed by the full file path to the syntax file. For example, the command line specification for macOS is

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

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

5 Blimp Output

After executing Blimp (via Blimp Studio or command-line), 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
                                       2.1.30

              Developed by Craig K. Enders, Brian T. Keller, and Han Du.
                         Blimp was developed with funding from
                           Institute of Educational Sciences
                          awards R305D150056 and R305D190002.
                Craig K. Enders, P.I. Email: cenders@psych.ucla.edu
         Programming by Brian T. Keller, Co-P.I. Email: bkeller2@ucla.edu
                      Han Du, Co-P.I. Email: hdu@psych.ucla.edu
                         There is 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. The previous scripts do not specify prior distributions for variance components or other special algorithm options. Appropriate default settings are invoked in this case.

```
ALGORITHMIC OPTIONS SPECIFIED:

  Imputation method:                    Fully Bayesian model-based
  MCMC algorithm:                       Full conditional Metropolis sampler
  Between-cluster imputation model:     Latent cluster means (LATENT)
  Residual variance structure:          Homogeneous level-1 variance (HOV)
  Prior for covariance matrices:        Zero matrix, df = -(p + 1) (PRIOR2)
  Prior for residual variance:          Zero sum of squares, df = -2 (PRIOR2)
  Prior for covariate model:            Unit sum of squares, df = 2 (XPRIOR1)
  Diagnostics:                          Potential scale reduction (PSR)
  Chain Starting Values:                Random starting values
```

Following the algorithmic options, Blimp displays a summary of the data and variables used in the analysis or imputation model.

```
DATA INFORMATION:

  Sample Size:            630
  Level-2 Clusters:       105
  Missing Data Code:      999.000
```

```
VARIABLES IN IMPUTATION MODEL:

  Level-2 identifier:     id
  Complete ordinal:       x1
  Incomplete continuous:  x2 w1
  Incomplete nominal:     w2
  Missing outcome:        y2 (continuous)
  CENTERED PREDICTORS
    Grand Mean Centered:  x1 w1
    Group Mean Centered:  x2
  Product Terms:          x2*w1
  Random Coefficients
    Level-2:              x2
```

Next, the output displays the progress of the MCMC algorithm. Blimp issues a message when it has completed 25%, 50%, 75%, and 100% of the burn-in iterations. Following the burn-in phase, Blimp prints a timestamp as it saves imputed data sets or the iteration history.

```
ITERATION HISTORY:

Starting Burn-in on Sun Aug 25 18:48:03 2019
    Burn-in iteration 500 completed on Sun Aug 25 18:48:03 2019
    Burn-in iteration 1000 completed on Sun Aug 25 18:48:03 2019
    Burn-in iteration 1500 completed on Sun Aug 25 18:48:03 2019
    Burn-in iteration 2000 completed on Sun Aug 25 18:48:04 2019
Burn-in completed on Sun Aug 25 18:48:04 2019
    Iteration 1000 completed on Sun Aug 25 18:48:04 2019
    Iteration 2000 completed on Sun Aug 25 18:48:05 2019
    Iteration 3000 completed on Sun Aug 25 18:48:05 2019
    Iteration 4000 completed on Sun Aug 25 18:48:06 2019
    Iteration 5000 completed on Sun Aug 25 18:48:07 2019
    Iteration 6000 completed on Sun Aug 25 18:48:07 2019
    Iteration 7000 completed on Sun Aug 25 18:48:08 2019
    Iteration 8000 completed on Sun Aug 25 18:48:08 2019
    Iteration 9000 completed on Sun Aug 25 18:48:09 2019
    Iteration 10000 completed on Sun Aug 25 18:48:09 2019
```

Finally, if imputations are requested on the SAVE command, Blimp prints the order of the variables in the imputed 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 IN SAVED DATA:

   id av y x1 x2 x3 w1 w2
```

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 (e.g., `SAVE: stacked = imps.dat`), the first variable in the file is an identifier variable that indexes the imputed data sets. In this case, the variable order appears as follows.

```
VARIABLE ORDER IN SAVED DATA:

   imp# id av y x1 x2 x3 w1 w2
```

Multiple Chains

When multiple chains are requested, Blimp prints slightly different output. The follow excerpts of output are based on an example of four MCMC chains. 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 HISTORY:

Chain 1 online with seed of 158875
Chain 2 online with seed of 150835
    Chain 2 completed burn-in on Sun Aug 25 14:55:57 2019
    Chain 1 completed burn-in on Sun Aug 25 14:55:57 2019
    Chain 2 completed iteration 1250 on Sun Aug 25 14:56:01 2019
    Chain 1 completed iteration 1250 on Sun Aug 25 14:56:01 2019
    Chain 2 completed iteration 2500 on Sun Aug 25 14:56:02 2019
    Chain 1 completed iteration 2500 on Sun Aug 25 14:56:02 2019
Chain 2 finished on Sun Aug 25 14:56:02 2019
Chain 1 finished on Sun Aug 25 14:56:02 2019
```

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. Upon completion of all imputations required in the first set of chains, the next set of chains will start and subsequently save imputations. 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 MCMC algorithm. PSR values less than 1.05 to 1.10 are typically viewed as acceptable. 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 second half of the iterations.

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, between-cluster variance/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 include regression coefficients, a residual variance, and possibly threshold parameters. PSR values are computed for every parameter in the analysis or imputation model, and the output prints summary tables that reflect the highest (worst) PSR value across all parameters. The PSR table below reflects the PSR computations from a four-chain MCMC process with random starting values for the means and variance parameters. In the first row of the table, 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). Across all regression model parameters, the highest (worst) PSR value is 1.697. In the second row, the first 100 iterations are discarded, and PSR values are computed by comparing parameter estimates from two separate MCMC chains with 100 iterations (iterations 101 through 201). The highest PSR value across all regression model parameters is 1.396.

```
POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

  Comparing iterations across 4 chains   | Highest PSR | Parameter # |
                         51 to 100       |       1.697 |          42 |
                        101 to 200       |       1.396 |          50 |
                        151 to 300       |       1.222 |          49 |
                        201 to 400       |       1.317 |          42 |
                        251 to 500       |       1.094 |          44 |
                        301 to 600       |       1.112 |          44 |
                        351 to 700       |       1.085 |          44 |
                        401 to 800       |       1.099 |           3 |
                        451 to 900       |       1.070 |           3 |
                        501 to 1000      |       1.079 |           2 |
                        551 to 1100      |       1.079 |           2 |
                        601 to 1200      |       1.057 |           2 |
                        651 to 1300      |       1.075 |           3 |
                        701 to 1400      |       1.059 |           3 |
                        751 to 1500      |       1.067 |           3 |
                        801 to 1600      |       1.076 |           2 |
                        851 to 1700      |       1.101 |           3 |
                        901 to 1800      |       1.080 |           3 |
                        951 to 1900      |       1.079 |           3 |
                       1001 to 2000      |       1.054 |           3 |
                       1051 to 2100      |       1.052 |          41 |
                       1101 to 2200      |       1.047 |          41 |
                       1151 to 2300      |       1.043 |          41 |
                       1201 to 2400      |       1.032 |          41 |
                       1251 to 2500      |       1.032 |          41 |
                       1301 to 2600      |       1.039 |          41 |
                       1351 to 2700      |       1.041 |          41 |
                       1401 to 2800      |       1.046 |          41 |
                       1451 to 2900      |       1.034 |          41 |
                       1501 to 3000      |       1.032 |          41 |
```

Bayesian texts suggest that PSR values below 1.05 to 1.10 are acceptable. In the table above, the PSR falls below 1.05 when comparing two chains of 1,100 iterations (iterations 1,101 through 2,200). This suggests that the burn-in interval should be at least 1,000. Threshold parameters for ordinal variables with more than two categories tend to converge very slowly, and we have observed no negative consequences from adopting a slightly less stringent criterion for these parameters (e.g., PSR values below 1.10). Finally, the right-most column in the table lists a numeric label associated with the parameter producing the PSR value displayed in the table. As described in the Command Language chapter, a text file containing the labels can be saved using the `SAVE` command. This file is automatically saved when using the `SAVE` command to output estimates from the burn-in period.

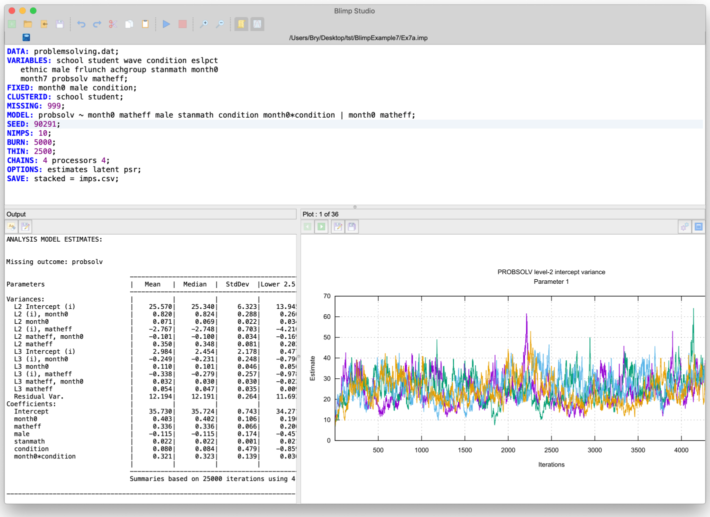
In addition to the PSR values, Blimp Studio has a graphing engine that automatically creates trace plots of the estimated parameters during the burn-in period. If one would like to manually include plots without the auto-plotting feature, this can be done by including the following specification in the syntax.

```
PLOT: traceplots;
```

Note, this command only works using Blimp Studio. Currently, Blimp only supplies trace plots and other diagnostic plots must manually be created by saving the burn-in iterations using the save command. In the future, we hope to include other diagnostic plots.

Trace plots can be useful for identifying individual parameters that are contributing to slow convergence or poor mixing. This can happen, for example, in a multilevel analysis that posits random coefficients that are not supported by the data. In this case, the trace plots could be very noisy, displaying unacceptably long periods of drift. This might be an indication that the model should be simplified. A second example of slow convergence that could be identified with trace plots occurs when using ordinal variables with very few responses in a category. In this case, the threshold parameters will be very noisy and sparse categories may need to be collapsed to stabilize estimation.

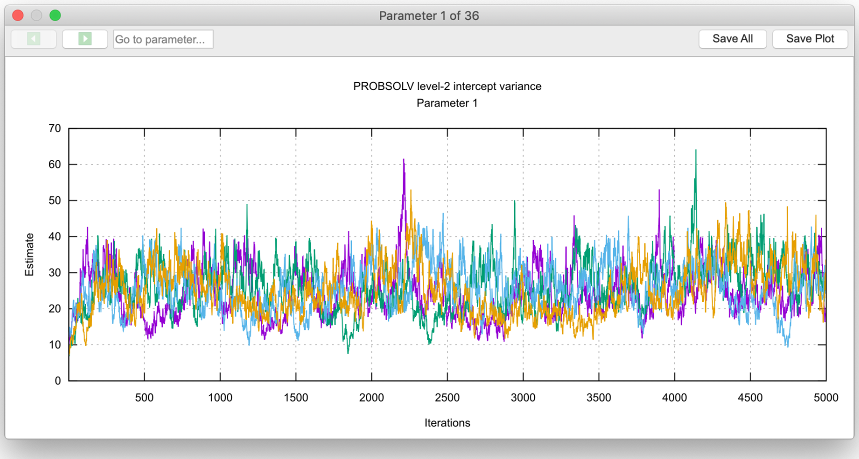
The plot window opens automatically at the completion of the script in Blimp Studio. The window can be resized by dragging the vertical or horizontal bars separating the syntax, output, and plot panes. The plot window can be hidden from view by clicking the normal distribution icon on the toolbar at the top of the application window.



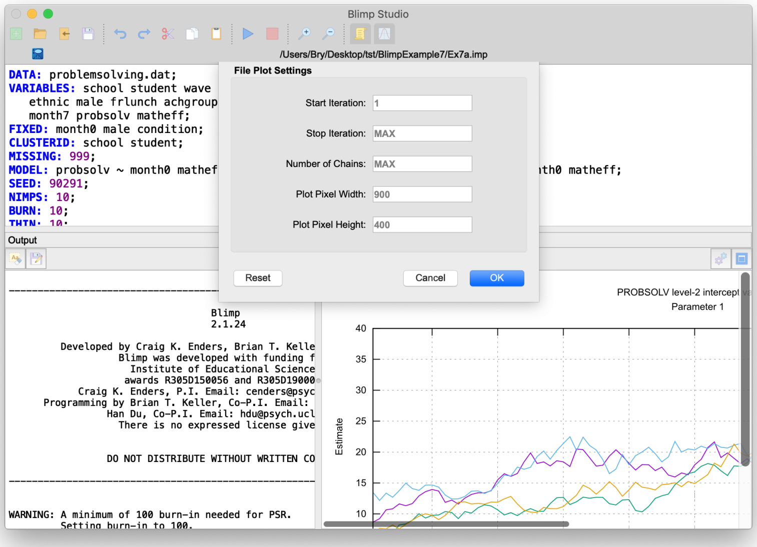
```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct
   ethnic male frlunch achgroup stanmath month0
   month7 probsolv matheff;
FIXED: month0 male condition;
CLUSTERID: school student;
MISSING: 999;
MODEL: probsolv ~ month0 matheff male stanmath condition month0*condition | month0 matheff;
SEED: 90291;
NIMPS: 10;
BURN: 5000;
THIN: 2500;
CHAINS: 4 processors 4;
OPTIONS: estimates latent psr;
SAVE: stacked = imps.csv;
```

```
ANALYSIS MODEL ESTIMATES:


Missing outcome: probsolv

                              ------------------------------------------
Parameters                    |   Mean    |  Median   |  StdDev  |Lower 2.5
                              ------------------------------------------
Variances:                    |           |           |          |
  L2 Intercept (i)            |     25.570|     25.340|     6.323|    13.94
  L2 (i), month0              |      0.820|      0.824|     0.288|     0.26
  L2 month0                   |      0.071|      0.069|     0.022|     0.03
  L2 (i), matheff             |     -2.767|     -2.748|     0.703|    -4.21
  L2 matheff, month0          |     -0.101|     -0.100|     0.034|    -0.16
  L2 matheff                  |      0.350|      0.348|     0.081|     0.20
  L3 Intercept (i)            |      2.984|      2.454|     2.178|     0.47
  L3 (i), month0              |     -0.249|     -0.231|     0.248|    -0.79
  L3 month0                   |      0.110|      0.101|     0.046|     0.05
  L3 (i), matheff             |     -0.338|     -0.279|     0.257|    -0.97
  L3 matheff, month0          |      0.032|      0.030|     0.030|    -0.02
  L3 matheff                  |      0.054|      0.047|     0.035|     0.00
  Residual Var.               |     12.194|     12.191|     0.264|    11.69
Coefficients:                 |           |           |          |
  Intercept                   |     35.730|     35.724|     0.743|    34.27
  month0                      |      0.403|      0.402|     0.106|     0.19
  matheff                     |      0.336|      0.336|     0.066|     0.20
  male                        |     -0.115|     -0.115|     0.174|    -0.45
  stanmath                    |      0.022|      0.022|     0.001|     0.02
  condition                   |      0.080|      0.084|     0.479|    -0.85
  month0*condition            |      0.321|      0.323|     0.139|     0.03
                              |           |           |          |
                              ------------------------------------------
                              Summaries based on 25000 iterations using 4

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

Clicking the right or left green arrow buttons pages through the graphs one at a time. Clicking the blue expand button in the upper right corner of the plot window spawns a dedicated plot viewer window. Plots can be saved individuals, or the entire set can be saved by selecting `Save All`.



Plots based on many chains and iterations may cause the viewer to respond slowly because the computer is required to render a large amount of information. The plotting parameters can be modified to display fewer chains or interaction by clicking the gear icon (preferences) in the upper right corner of the plot window. A window appears that allows the user to enter the number of chains to plot and the iteration range.



Blimp Studio
/Users/Bry/Desktop/tst/BlimpExample7/Ex7a.imp
Output
PROBSOLV level-2 intercept
Parameter 1
Estimate

```
DATA: problemsolving.dat;
VARIABLES: school student wave
   ethnic male frlunch achgroup
   month7 probsolv matheff;
FIXED: month0 male condition;
CLUSTERID: school student;
MISSING: 999;
MODEL: probsolv ~ month0 mathef                                   nth0 matheff;
SEED: 90291;
NIMPS: 10;
BURN: 10;
THIN: 10;
```

File Plot Settings

Start Iteration: 1
Stop Iteration: MAX
Number of Chains: MAX
Plot Pixel Width: 900
Plot Pixel Height: 400

Reset | Cancel | OK

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

                              Blimp
                              2.1.24

        Developed by Craig K. Enders, Brian T. Kelle
                 Blimp was developed with funding f
                   Institute of Educational Science
                  awards R305D150056 and R305D19000
          Craig K. Enders, P.I. Email: cenders@psyc
      Programming by Brian T. Keller, Co-P.I. Email:
                 Han Du, Co-P.I. Email: hdu@psych.ucl
                   There is no expressed license give

                DO NOT DISTRIBUTE WITHOUT WRITTEN CO

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

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

These default settings can be changed globally in Blimp Studio's Preferences in the File menu. Note, the keyword MAX is used for iterations and number of chains to represent them max number allowed by the Blimp script.

If desired, the SAVE command can output the parameter values sampled during the burn-in period. These parameters can be used to create trace plots or other diagnostics outside of Blimp. The following syntax illustrates how to save parameters from multiple chains, where the burn-in estimates are saved in separate files, and an asterisk (⋆) is required in the file path. This asterisk will be replaced with an integer in the file name (e.g., specifying params*.dat would produce data sets named params1.dat, params2.dat, etc.).

```
SAVE: burn =  /desktop/params⋆.dat;
```

The SAVE command outputs model parameters as matrices for subsequent processing (each parameter occupies a column, one row per burn-in iteration). We have provided an R program that produces trace plots similar to the ones produced by Blimp's internal graphics engine. We provide the code so that users might co-opt parts of the script and product autocorrelation or other types of plots not available in Blimp. The R program is available at www.appliedmissingdata.com. The actual plotting function is stored in a file named `blimptraceplotfunction.R`, and it is this function that could be edited to produce output other than line graphs. The following input below generate trace plots based on the output generated by the previous SAVE command.

```
# FILE PATH TO PLOTTING FUNCTION
# WINDOWS FILEPATHS MUST HAVE \\ OR / INSTEAD OF \
source("~/desktop/blimptraceplotfunction.R")

# PLOTTING PARAMETERS
ChainsToPlot <- 4
IterationsToPlot <- 1000
BurnFiles <- "~/desktop/params⋆.dat"
ChainColors <- c("blue","red","green","orange")

# CALL PLOTTING FUNCTION
traceplots(ChainsToPlot, IterationsToPlot, BurnFiles, ChainColors)
```

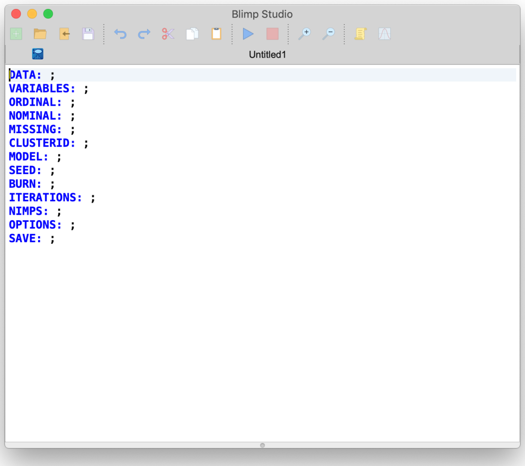
7 Analysis Examples

This chapter illustrates the application of Blimp to a variety of real-world analysis problems. The examples are inspired by an Institute of Educational Sciences-funded project featuring a cluster-randomized trial of a novel math problem-solving intervention (Montague, Krawec, Enders, & Dietz, 2014). The data structure consists of three levels: repeated measurements at level-1 nested in students at level-2, and students in turn nested in schools at level-3. Schools were randomly assigned to an intervention or control condition, such that all students within a given school received the same treatment. For simplicity, the analysis examples in this chapter use the same data set but ignore the nesting structure to illustrate imputation with single-level, 2-level, and 3-level regression analyses. The variables in the file are given in below, and the data and scripts for the examples are available in the Examples folder in the Blimp installation or from www.appliedmissingdata.com.

```
SCHOOL: School identifier variable
STUDENT: Student identifier variable
WAVE: Data collection wave, 1 to 7
CONDITION: Intervention code (0 = control, 1 = intervention)
ESLPCT: Percentage of students for whom English is a second language
ETHNIC: Ethnicity (1 = Caucasian, 2 = African American, 3 = Latina/Latino)
MALE: Gender dummy code (0 = female, 0 = male)
FRLUNCH: Lunch assistance code (0 = no, 1 = free/reduced lunch)
ACHGROUP: Three-group diagnostic classification (1 = learning disability, 2 =
    low-achieving, 3 = typically achieving)
STANMATH: Standardized math achievement scores
MONTH0: Months since the beginning of the school year
MONTH7: Months until the end of the school year
PROBSOLV: Math problem-solving
MATHEFF: Math self-efficacy scale
```

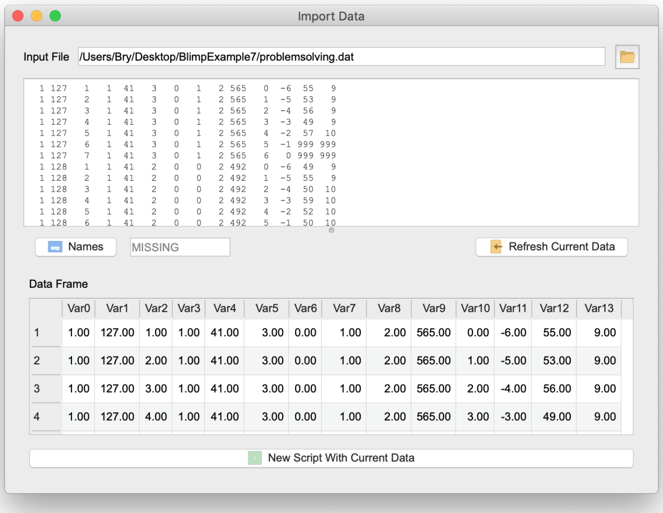
Working in Blimp Studio

Blimp scripts can be executed via command line arguments in macOS and Windows, but the easiest way to interact with the program is through Blimp Studio graphical user interface. One way to generate a script is to simply spawn a blank syntax window. To do so, choose `File` then `New` from the main application pull-down window. A new scripting window will open that is pre-populated with most of the major commands for multiple imputation. A Bayesian analysis requires only a couple minor modifications to the script. An empty Blimp syntax editor is shown below.



```
DATA: ;
VARIABLES: ;
ORDINAL: ;
NOMINAL: ;
MISSING: ;
CLUSTERID: ;
MODEL: ;
SEED: ;
BURN: ;
ITERATIONS: ;
NIMPS: ;
OPTIONS: ;
SAVE: ;
```

Blimp Studio also features a data import function that allows users to specify a file path, enter variable names, and specify a missing value code. The data import function is accessed by selecting `File` then `Import Data` from the application's main pull-down menus. This section illustrates the import process for the math problem-solving data. Blimp Studio first prompts the user to select the location of the input data file, after which it displays an initial preview of the data.



Input File /Users/Bry/Desktop/BlimpExample7/problemsolving.dat

```
 1 127   1   1  41   3   0   1   2 565   0  -6  55   9
 1 127   2   1  41   3   0   1   2 565   1  -5  53   9
 1 127   3   1  41   3   0   1   2 565   2  -4  56   9
 1 127   4   1  41   3   0   1   2 565   3  -3  49   9
 1 127   5   1  41   3   0   1   2 565   4  -2  57  10
 1 127   6   1  41   3   0   1   2 565   5  -1 999 999
 1 127   7   1  41   3   0   1   2 565   6   0 999 999
 1 128   1   1  41   2   0   0   2 492   0  -6  49   9
 1 128   2   1  41   2   0   0   2 492   1  -5  55   9
 1 128   3   1  41   2   0   0   2 492   2  -4  50  10
 1 128   4   1  41   2   0   0   2 492   3  -3  59  10
 1 128   5   1  41   2   0   0   2 492   4  -2  52  10
 1 128   6   1  41   2   0   0   2 492   5  -1  50  10
```

Data Frame

	Var0	Var1	Var2	Var3	Var4	Var5	Var6	Var7	Var8	Var9	Var10	Var11	Var12	Var13
1	1.00	127.00	1.00	1.00	41.00	3.00	0.00	1.00	2.00	565.00	0.00	-6.00	55.00	9.00
2	1.00	127.00	2.00	1.00	41.00	3.00	0.00	1.00	2.00	565.00	1.00	-5.00	53.00	9.00
3	1.00	127.00	3.00	1.00	41.00	3.00	0.00	1.00	2.00	565.00	2.00	-4.00	56.00	9.00
4	1.00	127.00	4.00	1.00	41.00	3.00	0.00	1.00	2.00	565.00	3.00	-3.00	49.00	9.00

Clicking the `Names` button launches a text box where variable names can be entered, and the numeric missing value code can be specified in the `MISSING` text box. The final data preview pane shows the column names and highlights missing values.

Clicking the `New Script With Current Data` button creates a new syntax editor with the data file and variable information populated with information from the input process.

Example 7.1: Single-Level Regression Analysis

Example 7.1 is a single-level regression analysis where standardized math scores, math self-efficacy, gender, and a three-category achievement classification variable predict problem-solving scores. The substantive analysis model is as follows.

$$\begin{aligned} probsolv_i = \gamma_0 + \gamma_1\left(stanmath_i\right) + \gamma_2\left(matheff_i\right) + \gamma_3\left(male_i\right) \\ + \gamma_4\left(achgroup1_i\right) + \gamma_5\left(achgroup2_i\right) + \epsilon_i \end{aligned}$$

For simplicity, the analysis example ignores the nesting structure and treats each row in the data set as an independent observation.

Because the analysis model does not include interaction or non-linear effects, the `FCS` command invokes an appropriate imputation procedure. The default `latent` specification treats `male` and `achgroup` as underlying normal latent variables throughout the entire imputation process (`achgroup` consists of three nominal categories that are modeled as two latent difference scores). That is, imputation is on the latent metric, and the latent imputes serve as predictors in other imputation models. This formulation is consistent with a so-called joint model specification that treats continuous variables and latent scores as multivariate normal. The `manifest` keyword triggers a routine where imputation is on the latent metric, but discrete imputes serve as predictors in other imputation models. In both cases, discrete imputes are saved to the output data files, and the categorical responses are generated by a function that links the latent and discrete variables.

In addition to estimating the substantive analysis model, the Bayesian analysis automatically models the distributions of the predictors with no user specification required. The default `latent` specification represents the 3-category nominal variable `achgroup` as two normally distributed latent difference scores. Continuous predictors are also assumed to be normal. Complete categorical variables such as `male` can be modeled as underlying normal latent variables, or they can be treated as fixed by design (i.e., no distribution specified, in line with conventional regression models). The analysis treats the gender dummy code as fixed by listing this variable on the `FIXED` line. The imputation step for the missing predictors uses a Metropolis sampler that simultaneously accounts for the associations among the continuous and latent predictors as well as the relations between the predictors and the outcome. Variables listed on the `NOMINAL` line (complete or incomplete) are automatically dummy coded, with the first group serving as the reference category. If desired, imputations can be saved using the `NIMPS` and `SAVE` commands, as in the FCS example.

Example 7.1a: Single-Level Imputation Script

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
NOMINAL: achgroup;
ORDINAL: male;
MISSING: 999;
FCS: probsolv stanmath matheff male achgroup;
SEED: 90291;
NIMPS: 10;
BURN: 5000;
THIN: 2500;
CHAINS: 4 processors 4;
OPTIONS: psr;
SAVE: stacked =  imps.csv; # R, SAS, or SPSS;
# use stacked0 = for Stata and separate = for Mplus;
```

Example 7.1b: Single-Level Bayesian Analysis Script

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
NOMINAL: achgroup;
FIXED: male;
MISSING: 999;
MODEL: probsolv ~  stanmath matheff male achgroup;
SEED: 90291;
BURN: 4000;
ITERATIONS: 10000;
CHAINS: 4 processors 4;
OPTIONS: psr;
```

All analysis examples use a burn-in period guided by the potential scale reduction factors, obtained by `psr` on the `OPTIONS` command line. An excerpt of the PSR output is shown below. The PSR indicates that a thinning interval of 2500 is sufficient because this value yields diagnostic value below 1.05, as shown in the last row of the output below.

Example 7.1b Multiple Imputation Output

```
POTENTIAL SCALE REDUCTION (PSR) OUTPUT:

Comparing iterations across 4 chains   | Highest PSR | Parameter # |
                        51 to 100      |       1.348 |          13 |
                       101 to 200      |       1.260 |          36 |
                       151 to 300      |       1.144 |          39 |
                       201 to 400      |       1.154 |          31 |
                       251 to 500      |       1.126 |          31 |
                       301 to 600      |       1.150 |          31 |
                       351 to 700      |       1.133 |          13 |
                       401 to 800      |       1.080 |           8 |
                       ...
                      2401 to 4800     |       1.075 |          13 |
                      2451 to 4900     |       1.056 |          13 |
                      2501 to 5000     |       1.033 |          13 |
```

Example 7.1b Bayesian Analysis Output

```
ANALYSIS MODEL ESTIMATES:

Missing outcome: probsolv
                                ---------------------------------------------------------
Parameters                      |   Mean   |  Median  |  StdDev  |Lower 2.5 |Upper 97.5|
                                ---------------------------------------------------------
Variances:                      |          |          |          |          |          |
  Residual Var.                 |    20.671|    20.668|     0.377|    19.949|    21.425|
Coefficients:                   |          |          |          |          |          |
  Intercept                     |    36.511|    36.509|     0.490|    35.542|    37.473|
  stanmath                      |     0.022|     0.022|     0.001|     0.020|     0.023|
  matheff                       |     0.499|     0.498|     0.038|     0.424|     0.574|
  male                          |    -0.052|    -0.052|     0.121|    -0.288|     0.184|
  achgroup#2                    |    -0.224|    -0.224|     0.222|    -0.659|     0.211|
  achgroup#3                    |     0.637|     0.637|     0.265|     0.117|     1.150|
                                |----------|----------|----------|----------|----------|
Standardized Coefficients:      |          |          |          |          |          |
  stanmath                      |     0.412|     0.412|     0.014|     0.385|     0.439|
  matheff                       |     0.148|     0.148|     0.011|     0.126|     0.170|
  male                          |    -0.005|    -0.005|     0.011|    -0.027|     0.017|
  achgroup#2                    |    -0.020|    -0.020|     0.019|    -0.058|     0.019|
  achgroup#3                    |     0.051|     0.051|     0.021|     0.009|     0.092|
                                |----------|----------|----------|----------|----------|
Proportion Variance Explained   |          |          |          |          |          |
  by Fixed Effects              |     0.267|     0.267|     0.009|     0.249|     0.284|
  by Residual Variation         |     0.733|     0.733|     0.009|     0.716|     0.751|
                                |          |          |          |          |          |
                                ---------------------------------------------------------
                                Summaries based on 10000 iterations using 4 chains}}
```

A Bayesian analysis automatically generates parameter estimates when the `NIMPS` command is omitted (as it is in the above script), but these quantities can always be requested with the `estimates` keyword on the `OPTIONS` line. The posterior means and standard deviations are analogous to frequentist point estimates and standard errors (e.g., a one-unit increase in self-efficacy predicts a .49 point increase in problem-solving, holding other variables constant). The 95% credible intervals rule out zero as a plausible parameter value for `stanmath`, `matheff`, and the dummy code comparing typically achieving and learning disabled students (the reference). Standardized coefficients and variance explained measures are automatically printed with the unstandardized posterior summaries. These additional quantities are functions of the model parameters from each MCMC iteration and thus have their own unique posterior distributions.

Example 7.2: Single-Level Regression with Interaction Effect

Example 7.2 is a single-level regression analysis that features a gender by self-efficacy interaction. The substantive analysis model is as follows.

$$\begin{aligned} probsolv_i = \gamma_0 + \gamma_1 \left(stanmath_i\right) + \gamma_2 \left(frlunch_i\right) + \gamma_3 \left(male_i\right) \\ + \gamma_4 \left(matheff_i\right) + \gamma_5 \left(male_i\right) \left(matheff_i\right) + \epsilon_i \end{aligned}$$

For simplicity, the analysis example ignores the nesting structure and treats each row in the data set as an independent observation. Because the analysis model includes and interaction effect, the `MODEL` command is used to invoke model-based imputation with Bayesian parameter estimates. The `FCS` command is not recommended for analyses with interactive effects.

In addition to estimating the substantive analysis model, model-based imputation and the Bayesian analysis automatically model the distributions of the predictors with no user specification required. The default `latent` specification treats `male` and `frlunch` as underlying normal latent variables, and continuous predictors are also assumed to be normal. However, both variables are discrete in the analysis model specified on the `MODEL` line. The imputation step for the missing predictors uses a Metropolis sampler that simultaneously accounts for the associations among the continuous and latent predictors as well as the relations between the predictors and the outcome.

The multiple imputation script creates imputations that are consistent with the hypothesized interaction effect, but the procedure does not output the product term. Researchers can center the lower-order variables post-imputation and compute the product (and potentially simple effects) in the same manner as a complete-data analysis. One impetus for using multiple imputation is that auxiliary variables can be included in the imputation process and ignored when analyzing the data outside of Blimp. To illustrate this possibility, the imputation script uses the nominal variable `achgroup` as an auxiliary variable. As noted in Example 7.1, this variable's associations with other predictors are modeled using underlying normal latent variables, and it appears as two dummy codes in the moderated regression model.

The Bayesian analysis features centered covariates and conditional effects that can be used to probe the interaction effect. In this framework, grand means are random variables to be estimated at each MCMC iteration. The `CENTERING` command invokes a Metropolis-Hastings step that draws grand or group means from a distribution that accounts for their presence and role in the analysis model. In this way, the analysis is accounting for the uncertainty in the centering constants. In order to model the grand means for centering, any complete predictors involved in an interaction effect, e.g., `male`, should not be listed on the `FIXED` line, as doing so would treat the mean as a known constant. Any product terms specified on the `MODEL` line automatically reflect the desired centering method. The `SIMPLE` command requests self-efficacy slopes for males and females (i.e., conditional effects, simple slopes). When the moderator variable to the right of the vertical pipe is nominal, conditional effects are automatically computed for each group. Conditional effects are computed as functions of the model parameters at each MCMC iteration, and thus have unique posterior distributions separate from the corresponding slopes in the analysis model.

Example 7.2a: Fully-Bayesian Model-Based Imputation Script

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
      stanmath month0 month7 probsolv matheff;
ORDINAL: frlunch male;
NOMINAL: achgroup;
MISSING: 999;
MODEL: probsolv ~  stanmath frlunch male matheff male*matheff achgroup;
SEED: 90291;
NIMPS: 10;
BURN: 4000;
THIN: 2000;
CHAINS: 4 processors 4;
OPTIONS: psr;
SAVE: stacked =  imps.csv; # R, SAS, or SPSS;
# use stacked0 = for Stata and separate = for Mplus;
```

Example 7.2b: Bayesian Analysis Script

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
ORDINAL: frlunch;
NOMINAL: male;
MISSING: 999;
MODEL: probsolv ~  stanmath frlunch male matheff male*matheff;
CENTERING: grandmean = stanmath frlunch matheff;
SIMPLE: matheff | male;
SEED: 90291;
BURN: 4000;
ITERATIONS: 10000;
CHAINS: 4 processors 4;
OPTIONS: psr;
```

Example 7.2b Bayesian Analysis Output

```
ANALYSIS MODEL ESTIMATES:

Missing outcome: probsolv

Grand Mean Centered: stanmath frlunch matheff

                                 -------------------------------------------------------
Parameters                       |   Mean   |  Median  |  StdDev  |Lower 2.5 |Upper 97.5|
                                 -------------------------------------------------------
Variances:                       |          |          |          |          |          |
  Residual Var.                  |    20.609|    20.608|     0.376|    19.884|    21.378|
Coefficients:                    |          |          |          |          |          |
  Intercept                      |    51.938|    51.938|     0.085|    51.773|    52.103|
  stanmath                       |     0.024|     0.024|     0.001|     0.023|     0.025|
  frlunch                        |    -0.169|    -0.168|     0.151|    -0.462|     0.128|
  male#1                         |    -0.064|    -0.064|     0.119|    -0.297|     0.170|
  matheff                        |     0.542|     0.542|     0.051|     0.442|     0.642|
  male#1*matheff                 |    -0.111|    -0.111|     0.077|    -0.257|     0.040|
                                 |----------|----------|----------|----------|----------|
Standardized Coefficients:       |          |          |          |          |          |
  stanmath                       |     0.456|     0.456|     0.010|     0.436|     0.476|
  frlunch                        |    -0.013|    -0.013|     0.011|    -0.035|     0.010|
  male#1                         |    -0.006|    -0.006|     0.011|    -0.028|     0.016|
  matheff                        |     0.161|     0.161|     0.015|     0.131|     0.190|
  male#1*matheff                 |    -0.021|    -0.021|     0.015|    -0.049|     0.008|
                                 |----------|----------|----------|----------|----------|
Proportion Variance Explained    |          |          |          |          |          |
by Fixed Effects                 |     0.269|     0.269|     0.009|     0.251|     0.287|
by Residual Variation            |     0.731|     0.731|     0.009|     0.713|     0.749|
                                 |          |          |          |          |          |
                                 -------------------------------------------------------
                                 Summaries based on 10000 iterations using 4 chains
```

The posterior means and standard deviations are analogous to frequentist point estimates and standard errors. The Bayesian analysis suggests that the gender by efficacy interaction could be zero because this null values falls within the 95% credible interval. The conditional effects output gives estimates of the male and female slopes. These estimates are valid regardless of whether the interaction is deemed important.

Example 7.2b Bayesian Analysis Output

```
CONDITIONAL EFFECTS ANALYSIS:

Missing outcome: probsolv

Grand Mean Centered: stanmath frlunch matheff

                                ---------------------------------------------------------
Conditional Effects             |   Mean    |  Median   |  StdDev   |Lower 2.5 |Upper 97.5|
                                ---------------------------------------------------------
  matheff | male#1 @ 0          |           |           |           |          |          |
    Intercept                   |    51.938|     51.938|      0.085|    51.773|    52.103|
    Slope                       |     0.542|      0.542|      0.051|     0.442|     0.642|
                                |           |           |           |          |          |
  matheff | male#1 @ 1          |           |           |           |          |          |
    Intercept                   |    51.874|     51.874|      0.095|    51.687|    52.055|
    Slope                       |     0.431|      0.431|      0.059|     0.315|     0.548|
                                |           |           |           |          |          |
                                ---------------------------------------------------------
                                Summaries based on 10000 iterations using 4 chains

                                NOTE: Intercepts are computed by setting all predictors
                                      not involved in the conditional effect to zero.
```

Example 7.3: Two-Level Regression with Random Intercepts

Example 7.3 is a two-level regression analysis with random intercepts, where math self-efficacy, gender, lunch assistance, and the achievement grouping variable predict problem-solving scores. The substantive analysis model is as follows.

$$\begin{aligned} probsolv_{ij} = \gamma_0 + \gamma_1 \left(matheff_{ij}\right) + \gamma_2 \left(male_j\right) + \gamma_3 \left(frlunch_j\right) + \gamma_4 \left(achgroup1_j\right) \\ + \gamma_5 \left(achgroup2_j\right) + u_{0_j} + \epsilon_{ij} \end{aligned}$$

For simplicity, the analysis example ignores nesting within schools and models observations (level-1) within students (level-2). Listing `student` on the `CLUSTERID` line automatically adds random intercepts and latent group means at the between-cluster level.

Because the analysis model does not include interaction or non-linear effects, the `FCS` command invokes an appropriate imputation procedure. Listing the student-level identifier on the `CLUSTERID` line automatically invokes random intercepts for all variables. Between-cluster associations are modeled uniquely from within-cluster associations using latent means (i.e., random intercepts, random effects). This specification is consistent with a so-called joint model that allows for unrestricted within- and between-cluster associations. The default `latent` specification treats `male`, `frlunch`, and `achgroup` as underlying normal latent variables, as described in Example 7.1. Discrete imputes are generated by a function that links the latent and discrete variables.

In addition to estimating the substantive analysis model, the Bayesian analysis automatically models the distributions of the predictors with no user specification required. The default `latent` specification represents nominal and ordinal variables as underlying normal latent variables, although they appear as discrete variables in the analysis. Continuous predictors are also assumed to be normal. Complete categorical variables such as `male` can be modeled as underlying normal latent variables, or they can be treated as fixed by design (i.e., no distribution specified, in line with conventional regression models). The analysis treats the gender dummy code as fixed by listing this variable on the `FIXED` line. Variables listed on the `NOMINAL` line (complete or incomplete) are automatically dummy coded, with the first group serving as the reference category. If desired, imputations can be saved using the `NIMPS` and `SAVE` commands, as in the FCS example.

Example 7.3a: Two-level Regression with Random Intercept Imputation Script

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
NOMINAL: achgroup;
ORDINAL: frlunch male;
CLUSTERID: student;
MISSING: 999;
FCS: probsolv matheff frlunch male achgroup ;
SEED: 90291;
NIMPS: 10;
BURN: 4000;
THIN: 1000;
CHAINS: 4 processors 4;
OPTIONS: psr;
SAVE: stacked =  imps.csv; # R, SAS, or SPSS;
# use stacked0 = for Stata and separate = for Mplus;
```

Example 7.3b: Bayesian Analysis for a Two-level Regression with Random Intercept

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
NOMINAL: achgroup;
ORDINAL: frlunch;
FIXED: male;
CLUSTERID: student;
MISSING: 999;
MODEL: probsolv ~  matheff frlunch male achgroup;
SEED: 90291;
BURN: 4000;
ITERATIONS: 10000;
CHAINS: 4 processors 4;
OPTIONS: psr;
```

Example 7.3b Bayesian Analysis Output

```
ANALYSIS MODEL ESTIMATES:

Missing outcome: probsolv

                                -------------------------------------------------------
Parameters                      |   Mean   |  Median  |  StdDev  |Lower 2.5 |Upper 97.5|
                                -------------------------------------------------------
Variances:                      |          |          |          |          |          |
  L2 Intercept (i)              |     8.615|     8.595|     0.513|     7.662|     9.678|
  Residual Var.                 |    15.003|    14.996|     0.299|    14.430|    15.610|
Coefficients:                   |          |          |          |          |          |
  Intercept                     |    45.951|    45.942|     0.626|    44.731|    47.184|
  matheff                       |     0.607|     0.607|     0.048|     0.512|     0.701|
  frlunch                       |    -0.687|    -0.685|     0.274|    -1.227|    -0.150|
  male                          |     0.210|     0.210|     0.213|    -0.211|     0.631|
  achgroup#2                    |    -0.089|    -0.088|     0.414|    -0.899|     0.718|
  achgroup#3                    |     3.788|     3.785|     0.448|     2.937|     4.681|
                                |----------|----------|----------|----------|----------|
Standardized Coefficients:      |          |          |          |          |          |
  matheff                       |     0.181|     0.181|     0.014|     0.153|     0.208|
  frlunch                       |    -0.052|    -0.051|     0.020|    -0.092|    -0.011|
  male                          |     0.020|     0.020|     0.020|    -0.020|     0.059|
  achgroup#2                    |    -0.008|    -0.008|     0.036|    -0.079|     0.063|
  achgroup#3                    |     0.305|     0.305|     0.035|     0.237|     0.374|
                                |----------|----------|----------|----------|----------|
Proportion Variance Explained   |          |          |          |          |          |
  by Fixed Effects              |     0.160|     0.160|     0.013|     0.136|     0.187|
  by Level-2 Random Intercepts  |     0.306|     0.306|     0.014|     0.279|     0.335|
  by Level-1 Residual Variation |     0.534|     0.534|     0.014|     0.507|     0.561|
                                |          |          |          |          |          |
                                -------------------------------------------------------
                                Summaries based on 10000 iterations using 4 chains
```

The posterior means and standard deviations are analogous to frequentist point estimates and standard errors. The Bayesian analysis results are similar to those in Example 7.1.b. The key difference is that this analysis introduces a random intercept and corresponding variance component that captures between-student differences. This residual between-cluster variation is denoted as `L2 Intercept (i)` in the output. The proportion variance explained effect sizes are based on those from Rights & Sterba (2018).

Example 7.4: Two-Level Regression with Random Slopes

Example 7.4 is a two-level regression analysis where math self-efficacy, gender, lunch assistance, and standardized math scores predict problem-solving scores. Math self-efficacy has a random coefficient that varies across students. The substantive analysis model is as follows.

$$\begin{aligned} probsolv_{ij} &= \gamma_0 + \gamma_1 \left(matheff_{ij}\right) + \gamma_2 \left(male_j\right) + \gamma_3 \left(frlunch_j\right) \\ &+ \gamma_4 \left(stanmath_j\right) + u_{0_j} + u_{1_j} \left(matheff_{ij}\right) + \epsilon_{ij} \end{aligned}$$

For simplicity, the analysis example ignores nesting within schools and models observations (level-1) within students (level-2). Because the analysis model includes a random coefficient, the `FCS` command would invoke an inappropriate imputation model that could introduces bias. Instead, the `MODEL` command is used to tailor imputations to this specific analysis model.

The random slope predictor is specified to the right of the vertical pipe on the `MODEL` line. In addition to estimating the substantive analysis model, the Bayesian analysis automatically models the distributions of the predictors with no user specification required. The default `latent` specification represents nominal and ordinal variables (e.g., `frlunch`) as underlying normal latent variables, although they appear as discrete variables in the analysis. Continuous predictors are also assumed to be normal. Complete categorical variables such as `male` can be modeled as underlying normal latent variables, or they can be treated as fixed by design (i.e., no distribution specified, in line with conventional regression models). The analysis treats the gender dummy code as fixed by listing this variable on the `FIXED` line. The multiple imputation script creates imputations that are consistent with the hypothesized random coefficient model.

The multiple imputation analysis does not invoke centering (imputations are always output on their original raw score metric), but the imputations can be centered outside of Blimp, as they would have been had the data been complete. The Bayesian analysis is a direct estimation method that gives the posterior distribution summaries without need for further analysis. To illustrate the `CENTERING` command, the repeated measures variable `matheff` is centered at its group mean (i.e., centered at each individual's latent mean). Group-mean centering creates a pure within-cluster predictor that is orthogonal to all person-level predictors. The latent group means can be added to the analysis model, if desired (see Example 7.8).

Example 7.4a: Model-Based Imputation Script for a Two-Level Regression with Random Slope

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
ORDINAL: frlunch;
CLUSTERID: student;
FIXED: male;
MISSING: 999;
MODEL: probsolv ~  matheff male frlunch stanmath | matheff;
SEED: 90291;
NIMPS: 10;
BURN: 4000;
THIN: 1000;
CHAINS: 4 processors 4;
OPTIONS: psr;
SAVE: stacked =  imps.csv, latent = latentdata.dat; # R, SAS, or SPSS;
# use stacked0 = for Stata and separate = for Mplus;
```

Example 7.4b: Bayesian Analysis for a Two-Level Regression with Random Slope

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
ORDINAL: frlunch;
CLUSTERID: student;
FIXED: male;
MISSING: 999;
MODEL: probsolv ~  matheff male frlunch stanmath | matheff;
CENTERING: grandmean = stanmath, groupmean = matheff;
SEED: 90291;
BURN: 4000;
ITERATIONS: 10000;
CHAINS: 4 processors 4;
OPTIONS: psr;
```

The multiple imputation script uses the `latent` keyword on the `SAVE` line to save the random intercept and slope residuals from the analysis model, the latent group means of the level-1 predictors, and the underlying latent variable scores for the categorical variables. The residuals and latent means (random intercepts) can be used for diagnostic purposes, as outlined in multilevel textbooks.

The Blimp output gives the order of variables in the latent data as follows, with `L2` and `L3` (three-level models) denoting the level at which a residual or latent mean appears in the model.

Example 7.4a Latent Data Variable List

```
VARIABLE ORDER IN SAVED DATA:

  imp# school student wave condition eslpct ethnic male frlunch achgroup
    stanmath month0 month7 probsolv matheff

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

VARIABLE ORDER IN LATENT MEANS DATA SET:

  imp# student L2:probsolv.residual L2:matheff.residual L2:matheff.mean
    frlunch.latent
```

Example 7.4b Bayesian Analysis Output

```
ANALYSIS MODEL ESTIMATES:

Missing outcome: probsolv

Grand Mean Centered: stanmath
Group Mean Centered: matheff

                                --------------------------------------------------------
Parameters                      |   Mean   |  Median  |  StdDev  |Lower 2.5 |Upper 97.5|
                                --------------------------------------------------------
Variances:                      |          |          |          |          |          |
  L2 Intercept (i)              |     5.965|     5.954|     0.398|     5.219|     6.778|
  L2 (i), matheff               |     0.547|     0.544|     0.199|     0.153|     0.940|
  L2 matheff                    |     0.654|     0.640|     0.167|     0.357|     1.010|
  Residual Var.                 |    14.442|    14.440|     0.308|    13.849|    15.072|
Coefficients:                   |          |          |          |          |          |
  Intercept                     |    52.077|    52.075|     0.251|    51.575|    52.559|
  matheff                       |     0.636|     0.635|     0.068|     0.505|     0.772|
  male                          |     0.010|     0.009|     0.191|    -0.361|     0.387|
  frlunch                       |    -0.255|    -0.254|     0.247|    -0.738|     0.234|
  stanmath                      |     0.026|     0.026|     0.001|     0.024|     0.028|
                                |----------|----------|----------|----------|----------|
Standardized Coefficients:      |          |          |          |          |          |
  matheff                       |     0.111|     0.111|     0.012|     0.089|     0.135|
  male                          |     0.001|     0.001|     0.018|    -0.034|     0.036|
  frlunch                       |    -0.019|    -0.019|     0.018|    -0.055|     0.017|
  stanmath                      |     0.493|     0.493|     0.015|     0.463|     0.522|
                                |----------|----------|----------|----------|----------|
Proportion Variance Explained   |          |          |          |          |          |
  by Fixed Effects              |     0.259|     0.259|     0.014|     0.231|     0.287|
  by Level-2 Random Intercepts  |     0.211|     0.210|     0.012|     0.187|     0.235|
  by Level-2 Random Slopes      |     0.020|     0.020|     0.005|     0.011|     0.031|
  by Level-1 Residual Variation |     0.510|     0.510|     0.014|     0.483|     0.537|
                                |          |          |          |          |          |
                                --------------------------------------------------------
                                Summaries based on 10000 iterations using 4 chains
```

The posterior means and standard deviations are analogous to frequentist point estimates and standard errors. The Bayesian analysis estimates an unrestricted covariance matrix for the random intercepts and random slopes. Group-mean centering the random slope predictor gives the pure within-student association between the repeated self-efficacy and problem-solving scores. However, this effect does not control for other variables because the group centered predictor is orthogonal to all level-2 variables, by definition. Centering decisions should be made on substantive grounds, and the choices here are strictly for illustration. The random slope variation is listed as `L2 matheff` in the variances section of the output. The proportion variance explained effect sizes are based on those from Rights & Sterba (2018).

Example 7.5: Two-Level Growth Model with a Cross-Level Interaction Effect

Example 7.5 is a two-level regression analysis that features a random slope for a temporal predictor (months since the start of the school year), and a cross-level group-by-time interaction involving intervention indicator.

$$\begin{aligned} probsolv_{ij} &= \gamma_0 + \gamma_1 \left(month0_{ij}\right) + \gamma_2 \left(matheff_{ij}\right) + \gamma_3 \left(male_j\right) + \gamma_4 \left(stanmath_j\right) \\ &\quad + \gamma_5 \left(condition_j\right) + \gamma_6 \left(month0_{ij}\right)\left(condition_j\right) + u_{0_j} + u_{1_j}\left(month0_{ij}\right) + u_{2_j}\left(matheff_{ij}\right) + \epsilon_{ij} \end{aligned}$$

For simplicity, the analysis example ignores nesting within schools and models observations (level-1) within students (level-2). Because the analysis model includes an interaction and random coefficients, the `FCS` command would invoke an inappropriate imputation model that could introduces bias. Instead, the `MODEL` command is used to tailor imputations to this specific analysis model.

Two random slope predictors are listed to the right of the vertical pipe. In addition to estimating the substantive analysis model, the imputation routine and Bayesian analysis automatically model the distributions of the predictors with no user specification required. The default `latent` specification treats nominal and ordinal variables as underlying normal latent variables, and continuous predictors are also assumed to be normal. However, categorical variables are discrete in the analysis specified on the `MODEL` line. The imputation step for the missing predictors uses a Metropolis sampler that simultaneously accounts for the associations among the continuous and latent predictors as well as the relations between the predictors and the outcome.

Complete categorical variables such as `male` can be modeled as underlying normal latent variables, or they can be treated as fixed by design (i.e., no distribution specified, in line with conventional regression models). The multiple imputation analysis lists three complete variables on the `FIXED` line: `month0` (the temporal predictor, months since baselines), `condition`, and `male`. The multiple imputation script creates imputations that are consistent with the hypothesized interaction effect, but the procedure does not output the product term. Further, the multiple imputation analysis does not invoke centering (imputations are always output on their original raw score metric), but the imputations can be centered outside of Blimp, as they would have been had the data been complete. One impetus for using multiple imputation is that auxiliary variables can be included in the imputation process and ignored when analyzing the data outside of Blimp. To illustrate this possibility, the imputation script uses the nominal variable `achgroup` as an auxiliary variable. As noted elsewhere, this variable's associations with other predictors are modeled using underlying normal latent variables, and it appears as two dummy codes in the analysis model.

The Bayesian analysis features centered covariates and conditional effects that can be used to probe the group-by-time interaction effect. In this framework, grand means and group means are random variables to be estimated at each MCMC iteration. The `CENTERING` command invokes a Metropolis-Hastings step that draws grand or group means from a distribution that accounts for their presence and role in the analysis model. In this way, the analysis is accounting for the uncertainty in the centering constants. In order to model the grand means for centering, any complete predictors, e.g., `condition` and `male`, should not be listed on the `FIXED` line, as doing so would treat the means as a known constant. Any product terms specified on the `MODEL` line automatically reflect the desired centering method. Note that group mean centering creates a pure within-cluster variable that is orthogonal to all level-2 variables. Centering decisions should be made on substantive grounds, and the choices here are just for illustration. The `SIMPLE` command requests growth rates for intervention and comparison groups (i.e., conditional effects, simple slopes). When the moderator variable to the right of the vertical pipe is ordinal, the desired values of the moderator must be specified with the @ symbol. This specification is not necessary if `condition` is specified on the `NOMINAL` line, in which case simple slopes are automatically computed for all groups. Conditional effects are computed as functions of the model parameters at each MCMC iteration, and thus have unique posterior distributions separate from the corresponding slopes in the analysis model.

Example 7.5a: Two-Level Regression with Random Slopes and a Cross-Level Interaction Imputation Script

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
NOMINAL: achgroup;
FIXED: month0 condition male;
CLUSTERID: student;
MISSING: 999;
MODEL: probsolv ~  month0 matheff male stanmath condition month0*condition |
   month0 matheff;
SEED: 90291;
NIMPS: 10;
BURN: 4000;
THIN: 2000;
CHAINS: 4 processors 4;
OPTIONS: psr;
SAVE: stacked =  imps.csv; # R, SAS, or SPSS;
# use stacked0 = for Stata and separate = for Mplus;
```

Example 7.5b: Bayesian Analysis for a Two-Level Regression with Random Slopes and a Cross-Level Interaction

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
ORDINAL: male condition;
FIXED: month0;
CLUSTERID: student;
MISSING: 999;
MODEL: probsolv ~  month0 matheff male stanmath condition month0*condition |
   month0 matheff;
CENTERING: grandmean = male stanmath, groupmean = matheff;
SIMPLE: month0 | condition@0, month0 | condition@1;
SEED: 90291;
BURN: 4000;
ITERATIONS: 10000;
CHAINS: 4 processors 4;
OPTIONS: psr;
```

The posterior means and standard deviations are analogous to frequentist point estimates and standard errors. Group-mean centering `matheff` yields a pure within-student association, controlling for growth. However, efficacy is orthogonal to the level-2 (student-level) predictors due to group mean centering. Centering decisions should be made on substantive grounds, and the choices here are strictly for illustration. Because the covariates are centered, the `intercept` and `month0` coefficients reflect the baseline mean and monthly growth rate for the comparison group, controlling for covariates. The positive interaction coefficient indicates the intervention group's growth rate is more positive by approximately .338. A slope difference of zero with well outside the credible interval. The `SIMPLE` command requests the growth rates (i.e., conditional effects, simple slopes) for the intervention and comparison groups. Consistent with the interaction effect from the main analysis model, the monthly growth rate coefficient for the intervention condition is .753, as compared to .416 for the comparison group. The proportion variance explained effect sizes are based on those from Rights & Sterba (2018).

Example 7.5b Bayesian Analysis Output

```
ANALYSIS MODEL ESTIMATES:

Missing outcome: probsolv

Grand Mean Centered: male stanmath
Group Mean Centered: matheff

                                 --------------------------------------------------------
Parameters                       |   Mean    |  Median   |  StdDev   |Lower 2.5 |Upper 97.5|
                                 --------------------------------------------------------
Variances:                       |           |           |           |          |          |
  L2 Intercept (i)               |     5.443|     5.412|     0.572|     4.398|     6.670|
  L2 (i), month0                 |    -0.024|    -0.023|     0.108|    -0.248|     0.182|
  L2 month0                      |     0.122|     0.120|     0.032|     0.063|     0.190|
  L2 (i), matheff                |     0.458|     0.456|     0.223|     0.035|     0.897|
  L2 matheff, month0             |    -0.070|    -0.070|     0.048|    -0.166|     0.026|
  L2 matheff                     |     0.480|     0.469|     0.143|     0.229|     0.797|
  Residual Var.                  |    12.192|    12.189|     0.282|    11.648|    12.754|
Coefficients:                    |           |           |           |          |          |
  Intercept                      |    50.247|    50.245|     0.200|    49.858|    50.644|
  month0                         |     0.416|     0.416|     0.042|     0.332|     0.499|
  matheff                        |     0.317|     0.316|     0.063|     0.194|     0.440|
  male                           |     0.004|     0.005|     0.189|    -0.362|     0.378|
  stanmath                       |     0.026|     0.026|     0.001|     0.024|     0.028|
  condition                      |    -0.252|    -0.251|     0.230|    -0.709|     0.189|
  month0*condition               |     0.338|     0.338|     0.053|     0.234|     0.442|
                                 |----------|----------|----------|----------|----------|
Standardized Coefficients:       |           |           |           |          |          |
  month0                         |     0.156|     0.156|     0.016|     0.124|     0.188|
  matheff                        |     0.055|     0.055|     0.011|     0.034|     0.077|
  male                           |     0.000|     0.000|     0.018|    -0.034|     0.035|
  stanmath                       |     0.493|     0.493|     0.015|     0.463|     0.521|
  condition                      |    -0.023|    -0.023|     0.021|    -0.065|     0.017|
  month0*condition               |     0.136|     0.136|     0.021|     0.094|     0.178|
                                 |----------|----------|----------|----------|----------|
Proportion Variance Explained    |           |           |           |          |          |
  by Fixed Effects               |     0.313|     0.313|     0.015|     0.283|     0.342|
  by Level-2 Random Intercepts   |     0.192|     0.191|     0.017|     0.160|     0.228|
  by Level-2 Random Slopes       |     0.065|     0.064|     0.020|     0.031|     0.106|
  by Level-1 Residual Variation  |     0.431|     0.431|     0.018|     0.395|     0.465|
                                 |           |           |           |          |          |
                                 --------------------------------------------------------
                                 Summaries based on 10000 iterations using 4 chains
```

Example 7.5b Bayesian Analysis Output

```
CONDITIONAL EFFECTS ANALYSIS:

Missing outcome: probsolv

Grand Mean Centered: male stanmath
Group Mean Centered: matheff

                                ---------------------------------------------------------
Conditional Effects             |   Mean    |  Median   |  StdDev  |Lower 2.5 |Upper 97.5|
                                ---------------------------------------------------------
  month0 | condition @ 0        |           |           |          |          |          |
    Intercept                   |     50.247|     50.245|     0.200|    49.858|    50.644|
    Slope                       |      0.416|      0.416|     0.042|     0.332|     0.499|
                                |           |           |          |          |          |
  month0 | condition @ 1        |           |           |          |          |          |
    Intercept                   |     49.995|     49.993|     0.164|    49.674|    50.315|
    Slope                       |      0.753|      0.754|     0.032|     0.691|     0.817|
                                |           |           |          |          |          |
                                ---------------------------------------------------------
                                Summaries based on 10000 iterations using 4 chains

                                NOTE: Intercepts are computed by setting all predictors
                                      not involved in the conditional effect to zero.
```

Example 7.6: Three-Level Regression with Random Intercepts

Example 7.6 is a three-level regression analysis with random intercepts, where math self-efficacy, gender, lunch assistance, the achievement grouping variable, and percentage of non-English speaking students in the school predict problem-solving scores. The example is similar to 7.3 but incorporates a third level of nesting within schools as well as a level-3 covariate.

$$\begin{aligned} probsolv_{ijk} &= \gamma_0 + \gamma_1\left(matheff_{ijk}\right) + \gamma_2\left(male_{jk}\right) + \gamma_3\left(frlunch_{jk}\right) + \gamma_4\left(achgroup1_{jk}\right) \\ &\quad + \gamma_5\left(achgroup2_{jk}\right) + \gamma_6\left(eslpct_k\right) + u_{0jk} + u_{0k} + \epsilon_{ijk} \end{aligned}$$

Consistent with the previous examples, the achievement classification variable is represented as two dummy codes. Listing `student` and `school` on the `CLUSTERID` line automatically adds random intercepts and latent group means at both between-cluster levels.

Because the analysis model does not include interaction or non-linear effects, the `FCS` command invokes an appropriate imputation procedure. Listing the student- and school-level identifiers on the `CLUSTERID` line automatically invokes random intercepts for all variables. The level-1 predictor `matheff` has latent group means at the second and third levels, and the student-level predictors have latent group means (random intercepts) at level-3. Between-cluster associations are modeled uniquely from within-cluster associations using latent means (i.e., random intercepts, random effects). This specification is consistent with a so-called joint model that allows for unrestricted within- and between-cluster associations. The default `latent` specification treats `male`, `frlunch`, and `achgroup` as underlying normal latent variables, as described elsewhere. Discrete imputes are generated by a function that links the latent and discrete variables.

In addition to estimating the substantive analysis model, the Bayesian analysis automatically models the distributions of the predictors with no user specification required. The default `latent` specification represents nominal and ordinal variables as underlying normal latent variables, although they appear as discrete variables in the analysis. Continuous predictors are also assumed to be normal. Complete categorical variables such as `male` can be modeled as underlying normal latent variables, or they can be treated as fixed by design (i.e., no distribution specified, in line with conventional regression models). The analysis treats the gender dummy code as fixed by listing this variable on the `FIXED` line. Variables listed on the `NOMINAL` line (complete or incomplete) are automatically dummy coded, with the first group serving as the reference category. If desired, imputations can be saved using the `NIMPS` and `SAVE` commands, as in the FCS example.

Example 7.6a: Three-Level Regression with Random Intercepts Imputation Script

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
NOMINAL: achgroup;
ORDINAL: frlunch male;
CLUSTERID: school student;
MISSING: 999;
FCS: probsolv matheff frlunch male achgroup eslpct;
SEED: 90291;
NIMPS: 10;
BURN: 4000;
THIN: 2000;
CHAINS: 4 processors 4;
OPTIONS: psr;
SAVE: stacked =  imps.csv; # R, SAS, or SPSS;
# use stacked0 = for Stata and separate = for Mplus;
```

Example 7.6b: Bayesian Analysis of a Three-Level Regression with Random Intercepts

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
NOMINAL: achgroup;
ORDINAL: frlunch;
FIXED: male;
CLUSTERID: school student;
MISSING: 999;
MODEL: probsolv ~  matheff frlunch male achgroup eslpct;
SEED: 90291;
BURN: 4000;
ITERATIONS: 10000;
CHAINS: 4 processors 4;
OPTIONS: psr;
```

Example 7.6b Bayesian Analysis Output

```
ANALYSIS MODEL ESTIMATES:

Missing outcome: probsolv

                                ---------------------------------------------------------
Parameters                      |   Mean   |  Median  |  StdDev  |Lower 2.5 |Upper 97.5|
                                ---------------------------------------------------------
Variances:                      |          |          |          |          |          |
  L2 Intercept (i)              |     6.621|     6.604|     0.423|     5.831|     7.503|
  L3 Intercept (i)              |     2.298|     2.157|     0.809|     1.161|     4.276|
  Residual Var.                 |    15.005|    15.001|     0.297|    14.436|    15.604|
Coefficients:                   |          |          |          |          |          |
  Intercept                     |    44.203|    44.190|     0.929|    42.440|    46.042|
  matheff                       |     0.592|     0.592|     0.047|     0.501|     0.686|
  frlunch                       |    -0.294|    -0.296|     0.278|    -0.830|     0.266|
  male                          |     0.187|     0.186|     0.200|    -0.203|     0.582|
  achgroup#2                    |     0.319|     0.321|     0.377|    -0.417|     1.054|
  achgroup#3                    |     3.503|     3.510|     0.417|     2.672|     4.317|
  eslpct                        |     0.024|     0.024|     0.013|    -0.000|     0.048|
                                |----------|----------|----------|----------|----------|
Standardized Coefficients:      |          |          |          |          |          |
  matheff                       |     0.178|     0.178|     0.014|     0.150|     0.206|
  frlunch                       |    -0.022|    -0.022|     0.021|    -0.063|     0.020|
  male                          |     0.018|     0.018|     0.019|    -0.019|     0.055|
  achgroup#2                    |     0.028|     0.028|     0.033|    -0.037|     0.093|
  achgroup#3                    |     0.284|     0.285|     0.033|     0.218|     0.348|
  eslpct                        |     0.094|     0.094|     0.049|    -0.002|     0.185|
                                |----------|----------|----------|----------|----------|
Proportion Variance Explained   |          |          |          |          |          |
  by Fixed Effects              |     0.136|     0.136|     0.016|     0.107|     0.168|
  by Level-2 Random Intercepts  |     0.239|     0.239|     0.014|     0.211|     0.268|
  by Level-3 Random Intercepts  |     0.082|     0.078|     0.026|     0.043|     0.145|
  by Level-1 Residual Variation |     0.542|     0.544|     0.020|     0.499|     0.578|
                                |          |          |          |          |          |
                                ---------------------------------------------------------
                                Summaries based on 10000 iterations using 4 chains
```

The posterior means and standard deviations are analogous to frequentist point estimates and standard errors. The Bayesian analysis results are similar to those in Example 7.3b. The key difference is that random intercepts and their corresponding variance components are now modeled at level-2 and level-3. The random intercept variation is listed as `L2 Intercept (i)` and `L3 Intercept (i)` on the output. The proportion variance explained effect sizes are based on those from Rights & Sterba (2018).

Example 7.7: Three-Level Regression with a Cross-Level Interaction

Example 7.7 is a three-level regression analysis that features a random slope for a temporal predictor (months since the start of the school year), and a cross-level group-by-time interaction involving intervention indicator. The example is similar to 7.5 but incorporates a third level of nesting within schools. The substantive analysis model is as follows.

$$\begin{aligned} probsolv_{ijk} = {} & \gamma_0 + \gamma_1 \left(month0_{ijk}\right) + \gamma_2 \left(matheff_{ijk}\right) + \gamma_3 \left(male_{jk}\right) + \gamma_4 \left(stanmath_{jk}\right) \\ & + \gamma_5 \left(condition_k\right) + \gamma_6 \left(month0_{ijk}\right) \left(condition_k\right) + u_{0jk} + u_{0k} + u_{1jk} \left(month0_{ijk}\right) \\ & + u_{1k} \left(month0_{ijk}\right) + \epsilon_{ijk} \end{aligned}$$

Because the analysis model includes an interaction and random coefficients, the `FCS` command would invoke an inappropriate imputation model that could introduces bias. Instead, the `MODEL` command is used to tailor imputations to this specific analysis model.

Two random slope predictors are specified to the right of the vertical pipe on the `MODEL` line. Random coefficients are automatically added at level-2 and at level-3. In addition to estimating the substantive analysis model, the imputation routine and Bayesian analysis automatically model the distributions of the predictors with no user specification required. The default `latent` specification treats nominal and ordinal variables as underlying normal latent variables, and continuous predictors are also assumed to be normal. However, categorical variables are discrete in the analysis specified on the `MODEL` line. The imputation step for the missing predictors uses a Metropolis sampler that simultaneously accounts for the associations among the continuous and latent predictors as well as the relations between the predictors and the outcome.

Complete categorical variables such as `male` can be modeled as underlying normal latent variables, or they can be treated as fixed by design (i.e., no distribution specified, in line with conventional regression models). The multiple imputation analysis lists three complete variables on the `FIXED` line: `month0` (the temporal predictor, months since baselines), `condition`, and `male`. The multiple imputation script creates imputations that are consistent with the hypothesized interaction effect, but the procedure does not output the product term. Further, the multiple imputation analysis does not invoke centering (imputations are always output on their original raw score metric), but the imputations can be centered outside of Blimp, as they would have been had the data been complete. One impetus for using multiple imputation is that auxiliary variables can be included in the imputation process and ignored when analyzing the data outside of Blimp. To illustrate this possibility, the imputation script uses the nominal variable `achgroup` as an auxiliary variable. As noted elsewhere, this variable's associations with other predictors are modeled using underlying normal latent variables, and it appears as two dummy codes in the analysis model.

The Bayesian analysis features centered covariates and conditional effects that can be used to probe the group-by-time interaction effect. In this framework, grand means and group means are random variables to be estimated at each MCMC iteration. The `CENTERING` command invokes a Metropolis-Hastings step that draws grand or group means from a distribution that accounts for their presence and role in the analysis model. In this way, the analysis is accounting for the uncertainty in the centering constants. In order to model the grand means for centering, any complete predictors, e.g., `condition` and `male`, should not be listed on the `FIXED` line, as doing so would treat the means as a known constant. Any product terms specified on the `MODEL` line automatically reflect the desired centering method. Note that group mean centering creates a pure within-cluster variable that is orthogonal to all level-2 variables. Centering decisions should be made on substantive grounds, and the choices here are just for illustration. The `SIMPLE` command requests growth rates for intervention and comparison groups (i.e., conditional effects, simple slopes). When the moderator variable to the right of the vertical pipe is nominal, simple slopes are automatically computed for all groups. Conditional effects are computed as functions of the model parameters at each MCMC iteration, and thus have unique posterior distributions separate from the corresponding slopes in the analysis model.

Example 7.7a: Three-Level Regression with Random Slopes and Cross-Level Interaction Imputation Script

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
NOMINAL: achgroup;
FIXED: month0 male condition;
CLUSTERID: school student;
MISSING: 999;
MODEL: probsolv ~  month0 matheff male stanmath condition month0*condition |
   month0 matheff;
SEED: 90291;
NIMPS: 10;
BURN: 5000;
THIN: 2500;
CHAINS: 4 processors 4;
OPTIONS: psr;
SAVE: stacked =  imps.csv; # R, SAS, or SPSS;
# use stacked0 = for Stata and separate = for Mplus;
```

Example 7.7b: Bayesian Analysis of a Three-Level Regression with Random Slopes and Cross-Level Interaction

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
   stanmath month0 month7 probsolv matheff;
NOMINAL: condition;
FIXED: month0 male;
CLUSTERID: school student;
MISSING: 999;
MODEL: probsolv ~  month0 matheff male stanmath condition month0*condition |
   month0 matheff;
CENTERING: grandmean = male stanmath, groupmean = matheff;
SIMPLE: month0 | condition;
SEED: 90291;
BURN: 10000;
ITERATIONS: 10000;
CHAINS: 4 processors 4;
OPTIONS: psr;
```

The posterior means and standard deviations are analogous to frequentist point estimates and standard errors. The Bayesian analysis estimates an unrestricted covariance matrix for the random intercepts and random slopes at level-2 and level-3. Because the covariates are centered, the `intercept` and `month0` coefficients reflect the baseline mean and monthly growth rate for the comparison group, controlling for covariates. The positive interaction coefficient indicates the intervention group's growth rate is more positive by approximately .280, controlling for other variables. A slope difference of zero with well outside the credible interval. The proportion variance explained effect sizes are based on those from Rights & Sterba (2018).

Example 7.7b Bayesian Analysis Output

```
ANALYSIS MODEL ESTIMATES:

Missing outcome: probsolv

Grand Mean Centered: stanmath male
Group Mean Centered: matheff

                                ---------------------------------------------------------
Parameters                      |   Mean   |  Median  |  StdDev  |Lower 2.5 |Upper 97.5|
                                ---------------------------------------------------------
Variances:                      |          |          |          |          |          |
  L2 Intercept (i)              |     4.727|     4.702|     0.501|     3.797|     5.776|
  L2 (i), month0                |    -0.038|    -0.031|     0.085|    -0.228|     0.110|
  L2 month0                     |     0.055|     0.053|     0.023|     0.014|     0.104|
  L3 Intercept (i)              |     1.042|     0.958|     0.465|     0.404|     2.178|
  L3 (i), month0                |     0.046|     0.047|     0.090|    -0.138|     0.229|
  L3 month0                     |     0.097|     0.090|     0.038|     0.045|     0.190|
  Residual Var.                 |    12.511|    12.508|     0.266|    11.987|    13.034|
                                |----------|----------|----------|----------|----------|
Coefficients:                   |          |          |          |          |          |
  Intercept                     |    50.015|    50.020|     0.682|    48.798|    51.287|
  month0                        |     0.448|     0.446|     0.094|     0.273|     0.643|
  matheff                       |     0.252|     0.252|     0.055|     0.146|     0.361|
  male                          |    -0.087|    -0.088|     0.178|    -0.441|     0.262|
  stanmath                      |     0.024|     0.024|     0.001|     0.022|     0.026|
  condition#1                   |    -0.118|    -0.118|     0.424|    -0.955|     0.729|
  month0*condition#1            |     0.280|     0.283|     0.125|     0.022|     0.512|
                                |----------|----------|----------|----------|----------|
Standardized Coefficients:      |          |          |          |          |          |
  month0                        |     0.174|     0.173|     0.036|     0.106|     0.247|
  matheff                       |     0.045|     0.045|     0.010|     0.026|     0.065|
  male                          |    -0.008|    -0.008|     0.017|    -0.043|     0.025|
  stanmath                      |     0.465|     0.466|     0.016|     0.432|     0.496|
  condition#1                   |    -0.011|    -0.011|     0.040|    -0.090|     0.068|
  month0*condition#1            |     0.116|     0.118|     0.051|     0.009|     0.211|
                                |----------|----------|----------|----------|----------|
Proportion Variance Explained   |          |          |          |          |          |
  by Fixed Effects              |     0.289|     0.289|     0.017|     0.255|     0.324|
  by Level-2 Random Intercepts  |     0.178|     0.177|     0.017|     0.146|     0.211|
  by Level-2 Random Slopes      |     0.008|     0.008|     0.003|     0.002|     0.015|
  by Level-3 Random Intercepts  |     0.039|     0.036|     0.016|     0.015|     0.079|
  by Level-3 Random Slopes      |     0.015|     0.013|     0.006|     0.007|     0.028|
  by Level-1 Residual Variation |     0.471|     0.471|     0.018|     0.437|     0.506|
                                |          |          |          |          |          |
                                ---------------------------------------------------------
                                Summaries based on 10000 iterations using 4 chains
```

The `SIMPLE` command requests the growth rates (i.e., conditional effects, simple slopes) for the intervention and comparison groups. Consistent with the interaction effect from the main analysis model, the monthly growth rate coefficient for the intervention condition is .728, as compared to .448 for the comparison group.

Example 7.7b Bayesian Analysis Output

```
CONDITIONAL EFFECTS ANALYSIS:

Missing outcome: probsolv

Grand Mean Centered: stanmath male
Group Mean Centered: matheff

                                --------------------------------------------------------
Conditional Effects             |   Mean    |  Median   |  StdDev  |Lower 2.5 |Upper 97.5|
                                --------------------------------------------------------
  month0 | condition#1 @ 0      |           |           |          |          |          |
    Intercept                   |    50.015|    50.020|     0.682|    48.798|    51.287|
    Slope                       |     0.448|     0.446|     0.094|     0.273|     0.643|
                                |           |           |          |          |          |
  month0 | condition#1 @ 1      |           |           |          |          |          |
    Intercept                   |    49.897|    49.881|     0.652|    48.688|    51.116|
    Slope                       |     0.728|     0.729|     0.077|     0.575|     0.879|
                                |           |           |          |          |          |
                                --------------------------------------------------------
                                Summaries based on 10000 iterations using 4 chains

                                NOTE: Intercepts are computed by setting all predictors
                                      not involved in the conditional effect to zero.
```

Example 7.8: Two-Level Regression Model with a Latent Contextual Effect

Example 7.8 is a two-level regression analysis where math self-efficacy, gender, lunch assistance, and standardized math scores predict problem-solving scores. Math self-efficacy has a random coefficient that varies across students. In addition, the math self-efficacy group means are included in the analysis (i.e., a contextual effect analysis. The substantive analysis model is as follows.

$$\begin{aligned} probsolv_{ij} &= \gamma_0 + \gamma_1\left(matheff_{ij}\right) + \gamma_2\left(male_j\right) + \gamma_3\left(frlunch_j\right) \\ &+ \gamma_4\left(stanmath_j\right) + \gamma_5\left(matheff.mean_j\right) + u_{0_j} + u_{1_j}\left(matheff_{ij}\right) + \epsilon_{ij} \end{aligned}$$

For simplicity, the analysis example ignores nesting within schools and models observations (level-1) within students (level-2). Because the analysis model includes a random coefficient, the `MODEL` command invokes model-based imputation with Bayesian parameter estimates. The random slope predictor is specified to the right of the vertical pipe on the `MODEL` line.

This analysis is a so-called latent contextual effects model that examines whether the influence of self-efficacy on problem-solving scores differs within student and between students. Group-mean centering the level-1 `matheff` predictor deviates this variable around its level-2 latent group means, which are treated as random variables during estimation. This form of centering estimates the pure within-student association between the repeated self-efficacy and problem-solving assessments. The `matheff.mean` specification adds the latent group means as a level-2 predictor. Complete categorical variables such as `male` can be modeled as underlying normal latent variables, or they can be treated as fixed by design (i.e., no distribution specified, in line with conventional regression models). The analysis treats the gender dummy code as fixed by listing this variable on the `FIXED` line.

Example 7.8: Bayesian Analysis for a Two-Level Regression with Random Slope and Latent Contextual Effect

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct ethnic male frlunch achgroup
      stanmath month0 month7 probsolv matheff;
FIXED: male;
ORDINAL: frlunch;
CLUSTERID: student;
MISSING: 999;
MODEL: probsolv ~  matheff male frlunch stanmath matheff.mean | matheff;
CENTERING: grandmean = male frlunch stanmath matheff.mean, groupmean = matheff
   ;
SEED: 90291;
BURN: 5000;
ITERATIONS: 10000;
CHAINS: 4 processors 4;
OPTIONS: psr;
```

Centering self-efficacy at the latent group means yields pure within- and between-cluster regression slopes. The results suggest that the within-student influence of self-efficacy on problem-solving is somewhat stronger than the between-student influence of average self-efficacy on average problem-solving. Centering `matheff` at its grand mean would change this level-2 coefficient to reflect the difference between the within- and between-cluster associations, which is the latent contextual effect.

Example 7.8b Bayesian Analysis Output

```
ANALYSIS MODEL ESTIMATES:

Missing outcome: probsolv

Grand Mean Centered: frlunch stanmath male matheff.mean
Group Mean Centered: matheff

                                 --------------------------------------------------------
Parameters                       |   Mean   |  Median  |  StdDev  |Lower 2.5 |Upper 97.5|
                                 --------------------------------------------------------
Variances:                       |          |          |          |          |          |
  L2 Intercept (i)               |     5.722|     5.713|     0.380|     4.998|     6.505|
  L2 (i), matheff                |     0.461|     0.457|     0.191|     0.094|     0.853|
  L2 matheff                     |     0.658|     0.650|     0.176|     0.349|     1.028|
  Residual Var.                  |    14.445|    14.447|     0.304|    13.861|    15.051|
                                 |----------|----------|----------|----------|----------|
Coefficients:                    |          |          |          |          |          |
  Intercept                      |    51.872|    51.871|     0.126|    51.634|    52.122|
  matheff                        |     0.605|     0.605|     0.068|     0.473|     0.737|
  male                           |    -0.030|    -0.028|     0.185|    -0.398|     0.327|
  frlunch                        |    -0.179|    -0.183|     0.237|    -0.632|     0.280|
  stanmath                       |     0.024|     0.024|     0.001|     0.022|     0.026|
  L2: matheff.mean               |     0.450|     0.450|     0.082|     0.286|     0.612|
                                 |----------|----------|----------|----------|----------|
Standardized Coefficients:       |          |          |          |          |          |
  matheff                        |     0.106|     0.106|     0.012|     0.083|     0.128|
  male                           |    -0.003|    -0.003|     0.017|    -0.037|     0.030|
  frlunch                        |    -0.013|    -0.014|     0.018|    -0.047|     0.021|
  stanmath                       |     0.459|     0.459|     0.016|     0.427|     0.489|
  L2: matheff.mean               |     0.108|     0.108|     0.020|     0.069|     0.146|
                                 |----------|----------|----------|----------|----------|
Proportion Variance Explained    |          |          |          |          |          |
  by Fixed Effects               |     0.269|     0.269|     0.014|     0.242|     0.295|
  by Level-2 Random Intercepts   |     0.202|     0.202|     0.012|     0.179|     0.226|
  by Level-2 Random Slopes       |     0.020|     0.020|     0.005|     0.011|     0.031|
  by Level-1 Residual Variation  |     0.510|     0.510|     0.013|     0.484|     0.535|
                                 |          |          |          |          |          |
                                 --------------------------------------------------------
                                 Summaries based on 10000 iterations using 4 chains
```

Example 7.9: Multiple-Group Growth Model with an Interaction Effect

Example 7.9 is a multiple-group growth model that features a random slope for a temporal predictor (months since the start of the school year), and a cross-level group-by-time interaction involving intervention indicator. The multiple-group aspect fits the model separately for males and females.

$$probsolv_{ij(group)} = \gamma_0 + \gamma_1 \left(matheff_{ij}\right) + \gamma_2 \left(stanmath_j\right) + \gamma_3 \left(month0_{ij}\right) \gamma_4 \left(condition_j\right)$$
$$+ \gamma_5 \left(month0_{ij}\right) \left(condition_j\right) + u_{0_j} + u_{1_j} \left(month0_{ij}\right) + \epsilon_{ij}$$

For simplicity, the analysis example ignores nesting within schools and models observations (level-1) within students (level-2). Because the analysis model includes an interaction and random coefficients, the `FCS` command would invoke an inappropriate imputation model that could introduces bias. Instead, the `MODEL` command is used to tailor imputations to this specific analysis model.

The random slope predictor (the temporary variable, months since baseline assessment) is listed to the right of the vertical pipe. In addition to estimating the substantive analysis model, the imputation routine and Bayesian analysis automatically model the distributions of the predictors with no user specification required. The imputation step for the missing predictors uses a Metropolis sampler that simultaneously accounts for the associations among the continuous (and latent predictors, when applicable) as well as the relations between the predictors and the outcome. The `BYGROUP` command specifies a complete grouping variable. In this case, listing `male` as the grouping variable estimates the growth model separately for males and females. Note that plotting is not available with multiple-group models, and the `SAVE` command is limited to imputations. In this example, the script performs a Bayesian analysis and also saves imputations (e.g., for analysis in an SEM package). By default, estimates are not printed whenever `NIMPS` is specified, and the script below overrides this default behavior by listing the `estimates` keyword on the `OPTIONS` line.

The Bayesian analysis features centered covariates and conditional effects that can be used to probe the group-by-time interaction effect separately for males and females. In this framework, grand means and group means are random variables to be estimated at each MCMC iteration. The `CENTERING` command invokes a Metropolis-Hastings step that draws grand or group means from a distribution that accounts for their presence and role in the analysis model. In this way, the analysis is accounting for the uncertainty in the centering constants. The `SIMPLE` command requests growth rates for intervention and comparison groups (i.e., conditional effects, simple slopes). When the moderator variable to the right of the vertical pipe is nominal, simple slopes are automatically computed for all groups. Conditional effects are computed as functions of the model parameters at each MCMC iteration, and thus have unique posterior distributions separate from the corresponding slopes in the analysis model.

Example 7.9: Multiple-Group Growth Model Analysis and Imputation Script

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct
   ethnic male frlunch achgroup stanmath month0
   month7 probsolv matheff;
NOMINAL: condition;
FIXED: month0;
CLUSTERID: student;
MISSING: 999;
BYGROUP: male;
MODEL: probsolv ~   matheff stanmath month0 condition month0*condition |
   month0;
CENTERING: grandmean =  matheff stanmath;
SIMPLE: month0 | condition;
SEED: 90291;
BURN: 2000;
ITERATIONS: 10000;
CHAINS: 4 processors 4;
OPTIONS: estimates psr;
SAVE: stacked =  imps.csv; # R, SAS, or SPSS;
# use stacked0 = for Stata and separate = for Mplus;
```

The posterior means and standard deviations are analogous to frequentist point estimates and standard errors. Grand-mean centering `matheff` and `stanmath` gives growth rate estimates controlling for these variables. Centering decisions should be made on substantive grounds, and the choices here are strictly for illustration. Because the covariates are centered, the `intercept` and `month0` coefficients reflect the baseline mean and monthly growth rate for the comparison group, controlling for covariates. The group-by-time coefficient's credible interval does not contain zero, but the female interaction coefficient is stronger than the male interaction, suggesting that the impact of the intervention was more salient for females. The proportion variance explained effect sizes are based on those from Rights & Sterba (2018).

Example 7.9 Bayesian Analysis Group 0 Output

```
ANALYSIS MODEL ESTIMATES:

Missing outcome: probsolv

Grand Mean Centered: matheff stanmath

                                 -------------------------------------------------------
Parameters                       |   Mean   |  Median  |  StdDev  |Lower 2.5 |Upper 97.5|
                                 -------------------------------------------------------
Variances:                       |          |          |          |          |          |
  L2 Intercept (i)               |     4.089|     4.067|     0.651|     2.910|     5.461|
  L2 (i), month0                 |     0.040|     0.046|     0.127|    -0.222|     0.286|
  L2 month0                      |     0.123|     0.122|     0.042|     0.046|     0.210|
  Residual Var.                  |    12.520|    12.514|     0.365|    11.821|    13.258|
Coefficients:                    |          |          |          |          |          |
  Intercept                      |    50.174|    50.176|     0.259|    49.661|    50.677|
  matheff                        |     0.388|     0.388|     0.058|     0.274|     0.503|
  stanmath                       |     0.024|     0.024|     0.001|     0.022|     0.026|
  month0                         |     0.356|     0.355|     0.058|     0.245|     0.469|
  condition#1                    |    -0.386|    -0.386|     0.286|    -0.949|     0.177|
  month0*condition#1             |     0.416|     0.416|     0.071|     0.275|     0.552|
                                 |----------|----------|----------|----------|----------|
Standardized Coefficients:       |          |          |          |          |          |
  matheff                        |     0.120|     0.120|     0.018|     0.085|     0.156|
  stanmath                       |     0.485|     0.485|     0.018|     0.449|     0.521|
  month0                         |     0.137|     0.137|     0.022|     0.095|     0.181|
  condition#1                    |    -0.036|    -0.036|     0.027|    -0.089|     0.017|
  month0*condition#1             |     0.172|     0.172|     0.029|     0.114|     0.229|
                                 |----------|----------|----------|----------|----------|
Proportion Variance Explained    |          |          |          |          |          |
  by Fixed Effects               |     0.363|     0.362|     0.018|     0.327|     0.398|
  by Level-2 Random Intercepts   |     0.152|     0.152|     0.021|     0.112|     0.195|
  by Level-2 Random Slopes       |     0.018|     0.018|     0.006|     0.007|     0.031|
  by Level-1 Residual Variation  |     0.467|     0.467|     0.020|     0.430|     0.507|
                                 |          |          |          |          |          |
                                 -------------------------------------------------------
                                 Summaries based on 10000 iterations using 4 chains

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

CONDITIONAL EFFECTS ANALYSIS:

Missing outcome: probsolv

Grand Mean Centered: matheff stanmath

                                 -------------------------------------------------------
Conditional Effects              |   Mean   |  Median  |  StdDev  |Lower 2.5 |Upper 97.5|
                                 -------------------------------------------------------
  month0 | condition#1 @ 0       |          |          |          |          |          |
    Intercept                    |    50.174|    50.176|     0.259|    49.661|    50.677|
    Slope                        |     0.356|     0.355|     0.058|     0.245|     0.469|
                                 |          |          |          |          |          |
  month0 | condition#1 @ 1       |          |          |          |          |          |
    Intercept                    |    49.788|    49.795|     0.208|    49.369|    50.180|
    Slope                        |     0.771|     0.771|     0.043|     0.689|     0.855|
                                 |          |          |          |          |          |
                                 -------------------------------------------------------
                                 Summaries based on 10000 iterations using 4 chains

                                 NOTE: Intercepts are computed by setting all predictors
                                       not involved in the conditional effect to zero.
```

Example 7.9 Bayesian Analysis Group 1 Output

```
ANALYSIS MODEL ESTIMATES:

Missing outcome: probsolv

Grand Mean Centered: matheff stanmath

                                ---------------------------------------------------------
Parameters                      |   Mean    |  Median   |  StdDev   |Lower 2.5 |Upper 97.5|
                                ---------------------------------------------------------
Variances:                      |           |           |           |          |          |
  L2 Intercept (i)              |     6.449|     6.389|     0.902|     4.844|     8.367|
  L2 (i), month0                |    -0.027|    -0.020|     0.157|    -0.368|     0.246|
  L2 month0                     |     0.109|     0.106|     0.044|     0.030|     0.202|
  Residual Var.                 |    12.598|    12.583|     0.413|    11.818|    13.434|
Coefficients:                   |           |           |           |          |          |
  Intercept                     |    50.389|    50.385|     0.310|    49.803|    51.009|
  matheff                       |     0.253|     0.254|     0.068|     0.117|     0.384|
  stanmath                      |     0.025|     0.025|     0.002|     0.022|     0.028|
  month0                        |     0.489|     0.489|     0.064|     0.364|     0.613|
  condition#1                   |    -0.087|    -0.085|     0.370|    -0.822|     0.626|
  month0*condition#1            |     0.232|     0.231|     0.081|     0.075|     0.393|
                                |----------|----------|----------|----------|----------|
Standardized Coefficients:      |           |           |           |          |          |
  matheff                       |     0.074|     0.075|     0.020|     0.034|     0.113|
  stanmath                      |     0.459|     0.461|     0.025|     0.410|     0.505|
  month0                        |     0.186|     0.186|     0.024|     0.139|     0.233|
  condition#1                   |    -0.008|    -0.008|     0.034|    -0.077|     0.058|
  month0*condition#1            |     0.095|     0.094|     0.033|     0.031|     0.160|
                                |----------|----------|----------|----------|----------|
Proportion Variance Explained   |           |           |           |          |          |
  by Fixed Effects              |     0.294|     0.295|     0.023|     0.249|     0.337|
  by Level-2 Random Intercepts  |     0.233|     0.232|     0.026|     0.184|     0.287|
  by Level-2 Random Slopes      |     0.016|     0.015|     0.006|     0.004|     0.029|
  by Level-1 Residual Variation |     0.457|     0.457|     0.023|     0.414|     0.504|
                                |           |           |           |          |          |
                                ---------------------------------------------------------
                                Summaries based on 10000 iterations using 4 chains

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

CONDITIONAL EFFECTS ANALYSIS:

Missing outcome: probsolv

Grand Mean Centered: matheff stanmath

                                ---------------------------------------------------------
Conditional Effects             |   Mean    |  Median   |  StdDev   |Lower 2.5 |Upper 97.5|
                                ---------------------------------------------------------
  month0 | condition#1 @ 0      |           |           |           |          |          |
    Intercept                   |    50.389|    50.385|     0.310|    49.803|    51.009|
    Slope                       |     0.489|     0.489|     0.064|     0.364|     0.613|
                                |           |           |           |          |          |
  month0 | condition#1 @ 1      |           |           |           |          |          |
    Intercept                   |    50.302|    50.304|     0.261|    49.780|    50.806|
    Slope                       |     0.722|     0.722|     0.049|     0.627|     0.818|
                                |           |           |           |          |          |
                                ---------------------------------------------------------
                                Summaries based on 10000 iterations using 4 chains

                                NOTE: Intercepts are computed by setting all predictors
                                      not involved in the conditional effect to zero.
```

Example 7.10: Three-Level Regression with External Analysis of Latent Scores

Example 7.10 is a three-level regression analysis that illustrates a contextual effects analysis for a binary variable, lunch assistance. The coefficients for the binary predictor and its group means (proportion of students receiving lunch assistance in the school) are not comparable because of scaling differences. The script saves the underlying normal latent variable and the latent group means of the binary variable, and these scores are subsequently analyzed as multiple imputations outside of Blimp. The ultimate analysis model is as follows, where `latentfrlunch` is the underlying normal latent variable, and `latentfrlunchmean` is the school-level group mean of the latent variable.

$$\begin{aligned} probsolv_{ijk} &= \gamma_0 + \gamma_1 \left(stanmath_{jk}\right) + \gamma_2 \left(male_{jk}\right) + \gamma_3 \left(latentfrlunch_{jk}\right) + \gamma_4 \left(latentfrlunchmean_k\right) \\ &\quad + u_{0jk} + u_{0k} + \epsilon_{ijk} \end{aligned}$$

Listing `student` and `school` on the `CLUSTERID` line automatically adds random intercepts and latent group means at both between-cluster levels.

Listing the student- and school-level identifiers on the `CLUSTERID` line automatically invokes random intercepts for all variables. The level-2 predictor `frlunch` has latent group means at the third level, and the student-level predictors have latent group means (random intercepts) at level-3. The default `latent` specification treats `frlunch` as an underlying normal latent variable, as described elsewhere. Discrete imputes are generated by a function that links the latent and discrete variables. Complete categorical variables such as `male` can be modeled as underlying normal latent variables, or they can be treated as fixed by design (i.e., no distribution specified, in line with conventional regression models). The analysis treats the gender dummy code as fixed by listing this variable on the `FIXED` line. Imputations and the latent data (random intercepts and slopes from the substantive analysis, latent group means of the predictors, and underlying latent variable scores for categorical variables) are saved to separate files using the `NIMPS` and `SAVE` commands. Both files are saved in a stacked format and can be merged outside of Blimp. Latent variable scores can be generated and saved for a variety of different analyses. For example, the `FCS` command could be used to generate latent imputations for subsequent use in a confirmatory factor analysis. The Blimp output gives the order of variables in the latent data as follows, with `L2` and `L3` (three-level models) denoting the level at which a residual or latent mean appears in the model.

Example 7.10: Three-Level Regression with Latent Imputations Script

```
DATA: problemsolving.dat;
VARIABLES: school student wave condition eslpct
   ethnic male frlunch achgroup stanmath month0
   month7 probsolv matheff;
ORDINAL: frlunch;
FIXED: male;
CLUSTERID: school student;
MISSING: 999;
MODEL: probsolv ~  stanmath  male frlunch frlunch.mean;
SEED: 90291;
BURN: 2000;
THIN: 1000;
NIMPS: 10;
CHAINS: 4 processors 4;
OPTIONS: psr;
SAVE: stacked = imps.csv, latent = latentdata.csv; # R, SAS, or SPSS;
# use stacked0 = for Stata and separate = for Mplus;
```

Example 7.10 Latent Data Variable List

```
VARIABLE ORDER IN SAVED DATA:

  imp# school student wave condition eslpct ethnic male frlunch achgroup
    stanmath month0 month7 probsolv matheff

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

VARIABLE ORDER IN LATENT MEANS DATA SET:

  imp# student school L2:probsolv.residual L3:probsolv.residual L3:stanmath.mean
    frlunch.latent L3:frlunch.mean
```

The `R` analysis script (below) uses the `lme4` and `mitml` packages to analyze the imputed data and pool the estimates. The lunch assistance latent variable (a level-2, or student-level variable) and its school-level (level-3) group mean are both scaled as z-scores. With no centering specified, the student-level lunch assistance variable represents the pure within-student association, and the latent group mean's coefficient reflects the net effect of the school effect above and beyond the student effect (i.e., the contextual effect). Alternatively, group-mean centering the student-level variable would give the level-2 and level-3 slopes. The analysis results suggest that the influence of lunch assistance is almost entirely at the school level.

Example 7.10: R Analysis of a Three-Level Regression with Latent Variable Contextual Effect

```
library(mitml)
library(lme4)

# Read imputation data
impdata <- read.csv("filepath/imps.csv", header = F)
names(impdata) <- c("imputation", "school", "student", "wave", "condition", "
   eslpct", "ethnic", "male", "frlunch", "achgroup", "stanmath", "month0", "
   month7", "probsolv", "matheff")

# Read latent data
latentdata <- read.csv("filepath/latentdata.csv", header = F)
names(latentdata) <- c("imputation", "student", "school", "probsolv.residual.
   l2", "probsolv.residual.l3", "stanmath.mean", "frlunch.latent","frlunch.
   latent.mean")

alldata <- cbind(impdata, latentdata)

# Analyze data and pool estimates
model <- "probsolv ~ stanmath + male + frlunch.latent + frlunch.latent.mean +
    (1|student:school) + (1|school)"
implist <-  as.mitml.list(split(alldata, alldata$imputation))
analysis <- with(implist, lmer(model, REML = F))
estimates <- testEstimates(analysis, var.comp = T, df.com = NULL)
estimates
```

Example 7.10 R Analysis Output

```
Final parameter estimates and inferences obtained from 10 imputed data sets.

                     Estimate Std.Error   t.value        df   P(>|t|)       RIV       FMI
(Intercept)            40.764     0.781    52.220   230.912     0.000     0.246     0.204
stanmath                0.024     0.001    24.378   930.932     0.000     0.109     0.100
male                   -0.060     0.177    -0.342  5049.533     0.732     0.044     0.043
frlunch.latent          0.040     0.105     0.380    69.950     0.705     0.559     0.376
frlunch.latent.mean    -0.893     0.588    -1.518    52.764     0.135     0.704     0.434

                                    Estimate
Intercept~~Intercept|student:school    4.610
Intercept~~Intercept|school            1.396
Residual~~Residual                    15.166

Unadjusted hypothesis test as appropriate in larger samples.
```

8 Running a Simulation

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).

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.
```

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.
```

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 8.1 using the R statistical programming language the `system()` function.

Example 8.1: 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 = dataPath, 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)
  outfile <- paste0(outputPath,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))
})
```

9 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 common Blimp error and warning messages and possible resolutions for each of them.

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: Unable to expand syntax. The prefixes for
       c1 and x3 do not match.
```

A variable list that uses a dash to indicate a sequence of variables on the `VARIABLE`, `NOMINAL`, `ORDINAL`, or `MODEL` commands is incorrectly specified, most likely because the alphanumeric prefixes provided do not match. In the example above, the variable list `c1-x3` would generate the error.

```
ERROR: Unable to expand syntax. The suffixes for
        x1 and x3a do not match.
```

A variable list that uses a dash to indicate a sequence of variables on the `VARIABLE`, `NOMINAL`, `ORDINAL`, or `MODEL` commands is incorrectly specified, most likely because the numeric suffixes provided do not match. In the example above, the variable list `x1-x3a` would generate the error.

```
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 Chapter 8.

```
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 Chapter 8.

```
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 Chapter 8.

```
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 Chapter 8.

```
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 Chapter 8.

```
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 8.

```
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.

```
ERROR: A non-numeric value of '.' was given in MISSING command.
       Currently only numeric missing codes are allowed.
```

Blimp only accepts numerical values as missing data codes at this time.

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 Chapter 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

Asparouhov, T., & Muthén, B. (2019). Advances in bayesian model fit evaluation for structural equation models. Retrieved from `http://www.statmodel.com/download/BayesFit.pdf`

Enders, C., Du, H., & Keller, B. (2019). A model-based imputation procedure for multilevel regression models with random coefficients, interaction effects, and non-linear terms. *Psychological methods*.

Enders, C., Keller, B., & Levy, R. (2017). A fully conditional specification approach to multilevel imputation of categorical and continuous variables. *Psychological methods*.

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.

Montague, M., Krawec, J., Enders, C., & Dietz, S. (2014). The effects of cognitive strategy instruction on math problem solving of middle-school students of varying ability. *Journal of Educational Psychology*, *106*(2), 469.

Rights, J. D., & Sterba, S. K. (2018). A framework of r-squared measures for single-level and multilevel regression mixture models. *Psychological methods*, *23*(3), 434.

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

Zhang, Q., & Wang, L. (2016). Moderation analysis with missing data in the predictors. *Psychological Methods, Advanced online publication.*