BLImP Software Manual
(Version beta 8.62)

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

October 2015

THIS IS BETA SOFTWARE WITH NO EXPRESSED LICENSE GIVEN

DO NOT DISTRIBUTE WITHOUT WRITTEN CONSENT

Contents

1 Introduction

BLImP is a software program designed to perform multiple imputation via fully conditional specification. It has the capabilities to perform both single-level and two-level imputation with continuous, binary/ordinal, and nominal variables. Moreover, BLImP has the capabilities to model random-slopes between both incomplete and complete variables. BLImP also has the capabilities to handle heterogenous residual variance structures in two-level models.

1.1 DISCLAIMER

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

Keller, B. T., & Enders, C. K. (2015). *BLImP Software Manual (Version beta 8.62)*. Los Angeles, CA.

INSERT FORTHCOMING CHAINED EQUATION PAPER REFERENCE?

Keller, B. T., & Enders, C. K. (2014, May). A Latent Variable Chained Equations Approach for Multilevel Multiple Imputation. Paper presented at the Modern Modeling Methods Conference, Storrs, Connecticut.

1.2 Quick Start

This section is designed to quickly set up BLImP and start using it via a small tutorial. All of the information presented is repeated in more detail in the subsequent chapters. Thus, the interested reader can skip to the next chapter.

Input File Example

```
# This is a comment

DATA: /Users/name/Desktop/mydata.dat;

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

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

BURN: 2000;

THIN: 500;

NIMPS: 10;

MISIND: -99;

SEED: 84398;

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

OPTIONS: separate dat;
```

Explanation of Syntax

Conventions The above input file is presented to BLImP as a plain text (raw ASCII format). Some of the basic conventions for BLImP are that each command (in capital letters in the example) are followed by a colon (:) and a space. Each line is terminated with a semicolon (;). For example, the `VARNAMES` command spans across two lines in the example above. Another convention is the number sign (#) will instruct BLImP to ignore that entire line and treats it as a comment.

Inputting Data The `DATA`, `VARNAMES`, `MISSING` commands deal with the input of data into BLImP. The data set supplied must be presented as a CSV (Comma Separated Values) format or a whitespace delimited file. The missing values must be coded a numeric code which is specified via the `MISSING` command. The `DATA` command specifies the full file path to the data set desired. Take note that the path must not be in quotations. The `VARNAMES` command specifies the variable names given to each column of the data set. These *mist* be alphanumeric in nature with no length requirement.

Model Command

The `MODEL` command specifies the id variable BLImP should use, the variables desired in the imputation, and the type of variable they are. The model statement starts by first specifying the id variable desired. For single level imputation one must specify the keyword `tmp.id`. The id variable is followed by a space, a tilde (~), and another space. Now one can list the variables desired, separating each with a space, in the imputation model. Note that this includes both complete and incomplete variables. In addition, BLImP will automatically determine level-1 and level-2 variables. The example input above, demonstrates all of BLImP's conventions. A `o.` classifies the variable as binary/ordinal and a `n.` classifies the variable as nominal. To specify a random slope between two level-1 variables, one must place a colon (:) in between the two variables instead of a space. Note to specify multiple random slopes with one varaible the name must be listed multiple times.

Algorithmic Options

The `BURN`, `THIN`, `SEED`, `NIMPS`, and `OPTIONS` specify features of the algorithm that BLImP runs. In the above example, the `BURN` command is set to 2000, which is setting the burn-in interval to 2000. This means that the first 2000 iterations will be discarded before the initial imputation is saved. The `THIN` command instructs BLImP that the between imputation thinning interval is equal to 500. Therefore, BLImP will run 500 iterations after an imputation before saving next imputation. The `NIMPS` command specifies the number of imputations desired. The `SEED` command specifies the seeding value for the pseudo random number generator used by BLImP. The `OPTIONS` command specifies various miscellaneous options, such as how the data is saved, priors being used, etc. More information about the specific options and their defaults is presented in Chapter 4. The example above is relying on BLImP to specify most options by default, only providing that separated data files (i.e., each imputation is saved in a separate file) and a space delimited file type is desired. Finally, the `OUTFILE` command specifies the complete file path to where the data should be saved. In the case of separate data files, an asterisk (*) must be provided to where a number will be placed for the imputation number.

Running BLImP

Currently BLImP is ran via a command-line. In order to run BLImP, one must specify open a terminal program (e.g., Terminal.app on Mac OS X). Once open, place the full file path to the BLImP executable followed by the full file path to the input file (labeled as InputExample.imp below).

```
$ /path/to/blimp/BLImp /path/to/input/InputExample.imp
```

Most terminal applications will allow you to drag the file into the open window and paste the file path for that file. In that face, drag the BLImP executable into the window and then drag the input file. More detail about running BLImP is given in Chapter 5.

2 Input File

BLImP uses a plain text (raw ASCII format) document as an input file in order to specify the options. In future releases the file extension '.imp' will be associated with a BLImP graphical user interface. Therefore, it is recommended to save files as a '.imp' extension for future compatibility. There are 10 main commands used in a standard BLImP input and are listed in Subsection 2.2.

2.1 General Conventions

There are several general conventions that are part of a BLImP input file. BLImP is *case-sensitive*, that is, X1 and x1 are interpreted the same. All commands (listed in Subsection 2.2) are followed by a colon (:) and a space. Each line is terminated with a semicolon (;). A commented line is denoted with a number sign (#). This will signal BLImP to ignore all text following the number sign *entire* until a new line.

2.2 Main Commands

`DATA:` The `DATA` command gives the absolute file path to the data set (no quotations). Currently BLImP is set up to attempt to auto-detect the data structure given. Data must be provided in plain text, with no header, and either white space or a comma separating the columns. It is important to note that BLImP currently only accepts numerical values in the data sets. All missing values must be coded as a numeric value (e.g., -9999) that is impossible to obtain in the data set. Furthermore, any nominal or ordinal variables must be coded as numeric values (e.g., 0, 1, 2, etc.).

`VARNAMES:` The `VARNAMES` command lists the variable names in the data set. The names must be alphanumeric with a space separating them in order from left to right in the data set. The variables in this list will not necessarily be used in the imputation process. In contrast,

the variables in the list will appear in the imputed data sets returned by BLImP with no changes made.

`MODEL:` The `MODEL` command is used to specify the underlying model in BLImP . Importantly, this is where one states the variable type (continuous, binary/ordinal, or nominal) of the variables used. The general structure of a model is the id variable followed by a tilde (~) and the desired variables to be used in the imputation. This includes incomplete variables and complete auxiliary variables at both levels. Importantly, BLImP determines whether a variable is incomplete or complete and what level the variable is observed at automatically. More details of the `MODEL` command are given in Section 3.

`BURN:` The `BURN` command is used to specify the burn-in iterations for the algorithm to discard before saving an imputation. This should be determined by diagnostics which are addressed in Section 7.

`THIN:` The `THIN` command is used to specify the within imputation thinning interval used by BLImP.

`NIMPS:` The `NIMPS` command is used to specify the number of imputation to be saved by BLImP.

`MISSING:` The `MISSING` command is used to specify the numeric value that represents a missing value.

`SEED:` The `SEED` command is used to specify the pseudorandom number generator seeding value. It is required to be a number. Note, unlike many software packages, BLImP requires a seeding value to run. This is to force replicability of results and BLImP will not use a timestamp as a seed.

`CHAINS:` The `CHAINS` command is used to specify the number of simultaneous chains that BLImP will run. The main advantage of this is to speed up the imputation process. BLImP uses multithreading and will run each chain on a separate thread. By default this is set to one and not specifying the chains command will use one. The max amount of chains allowed is 8.

`OPTIONS:` The `OPTIONS` command is used to set algorithmic options. If this command is excluded, then BLImP will rely on default settings. Specific details on the options that can be used are given in Chapter 4.

2.3 Example Input File

This section gives a template for a standard BLImP input file.

```
# This is a comment

DATA: /Users/name/Desktop/mydata.dat;

VARNAMES: idvar x y z;

MODEL: idvar ~ x y z;

BURN: 500;

THIN: 100;

NIMPS: 10;

MISSING: -9999;

SEED: 38203;

CHAINS: 1;

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

OPTIONS: csv separated;
```

3 The MODEL Command

The MODEL command is used to specify the variables, the types of the variables, and the underlying model BLImP uses. This is done through a BLImP model statement, which is made up of two parts, the id call and the model variables.

3.1 Specifying the Model Statement

A model statement contains two parts: (1) *id* part and (2) *model* part. Both are required in order to specify enough information so BLImP can form the appropriate model. A complete model statement looks as follows:

$$\text{id} \sim \text{model}$$

id The id call is the name given to the id-variable listed in the VARNAMES command. For two-level imputation, this would be the level-2 cluster id-variable. For single-level imputation, this would be a subject number, however, if single-level imputation is desired, it is suggested to use `tmp.id` instead.

`tmp.id` The `tmp.id` keyword signals to BLImP that the intent is single-level imputation. Using this keyword in the id call will treat each row in the data set as a single subject (i.e., single-level data structure) and will optimize BLImP for this type of data.

model The model call is a list of *all* variables that are used in the imputation model separated by a space. This includes incomplete and complete auxiliary variables at both level-1 and level-2. BLImP does not require any syntax to specify what variables are level-1 and what variables are level-2. This is also true for incomplete and complete variables. The names used are those given in the VARNAMES command. For example, to specify X, Y, and Z in an imputation model with the id-variable, idvar, the model statement is as follows:

```
idvar ~ X Y Z
```

Thus the MODEL command is as follows:

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

3.2 Specifying Random Slopes

When using two-level models, BLImP by default specifies a random intercept model for all incomplete variables. In order to specify a random slope between two level-2 variables a colon (:) is placed in between the two variable names. For example, to specify a random slope between X and Y in an imputation model with the id-variable, idvar, the model statement follows:

```
idvar ~ X:Y
```

If Y is to have a random slope with another variable, Z, then Y is repeated twice in the model statement:

```
idvar ~ X:Y Y:Z
```

Finally, if a random slope is to be specified between Y and X, Y and Z, and X and Z, then one can chain the statements in a compact form:

```
idvar ~ X:Y:Z
```

3.3 Specifying Variable Types

BLImP has the capabilities to handle three types of variables: (1) normal continuous, (2) binary/ordinal, and (3) nominal. These variable types are only specified for *missing* variables and are not needed for complete auxiliary variables. Note, if a nominal auxiliary variable is desired, it currently must be dummy coded outside of BLImP. By default BLImP treats every variable as continuous, thus, there is no syntax required to specify a variable as such. However, for binary/ordinal and nominal variables, one must specify this in the MODEL command using a prefix with a period (.) followed by the variable name. The prefix are as follows:

- `o.` Using an `o.` prefix signals to BLImP that the variable is binary/ordinal (where binary is a special case of ordinal) and imputations will be handled using probit model.

`n.` Using an `n.` prefix signals to BLImP that the variable is nominal and imputations will be handled using multinomial probit model. 212 213

4 The OPTIONS Command

The OPTIONS command is used to specify specific algorithmic options that are desired and other options (e.g., how the data will be saved). Each keyword is placed with a space separating it from the previous one. Section 4.1 defines the keywords and explains what they do.

4.1 Option Keywords

Note: **Bolded** options are the defaults used by BLImP .

`csv` The `csv` keyword will write out the imputations generated by BLImP as CSV (Comma Separated Values) format.

`dat` The `dat` keyword will write out the imputations generated by BLImP as a space delimited file.

`stacked` The `stacked` (`stack`) keyword will write out the imputations generated into one file specified in the OUTPUT command. Each imputation is concatenated to the previous imputation and the first variable denotes which imputation the observation belongs to.

`separate` The `separate` (`sep`) keyword will write out the imputations generated into multiple files specified in the OUTPUT command. A * must be specified in the OUTPUT command filepath in order to denote where a number will be placed with the imputation number.

`heterogeneous` The `heterogenous` (`hev`) keyword specifies a heterogenous residual variance model in the imputation process for all two-level variables.

`homogeneous` The `homogenous` (`hov`) keyword specifies a homogenous residual variance model in the imputation process for all two-level variables.

`clmean` The `clmean` keyword specifies that cluster means will be computed and added for all incomplete and complete variables when imputing level-1 variables.

`noclmean` The `noclmean` keyword specifies to not compute cluster means when imputing level-1 variables.

`iw0` The `iw0` keyword specifies the prior that will be used when drawing the level-2 covariance matrix and the residual variances (both level-1 and level-2) in the imputation model. Specifically, `iw0` gives the prior of $W^{-1}(0, -p-1)$, where p is the number of random effects in the model.

`iw1` The `iw1` keyword specifies the prior that will be used when drawing the level-2 covariance matrix and the residual variances (both level-1 and level-2) in the imputation model. Specifically, `iw0` gives the prior of $W^{-1}(I, -p-1)$, where I is a $p \times p$ identity matrix and p is the number of random effects in the model.

`nopsr` The `nopsr` keyword will turn off the convergence diagnostic. This is the recommended setting by default. The diagnostic routine is memory intensive, thus it is recommended to run a separate input to just check convergence. See Chapter 7 for more information.

`psr` The `psr` keyword specifies will turn on the convergence diagnostic. The diagnostic routine is memory intensive, thus it is recommended to run a separate input to check convergence. See Chapter 7 for more information.

5 Running BLImP

Currently BLImP is distributed as a Unix executable for Mac OS X and must be ran via the Terminal.app or other command-line interface (e.g., R's `system()` command). The command-line arguments BLImP accepts provides several advance uses. However, such uses are covered in Chapter 8 and this chapter will focus on running BLImP for standard purposes.

5.1 Opening BLImP with Terminal.app

One method of running an input file (covered in Chapter 2) is to open BLImP with the Terminal.app, either by secondary clicking and choosing "Open With ▸ Terminal" or by double-clicking the Unix executables (if the file type is associated with a command-line interface). This is analogous to running the Terminal.app and typing BLImP's path with no inputs. Upon running BLImP using this method, BLImP's header will be given between two lines followed by a request for the input file.

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


                              BLImP
                           vBETA 8.62
        Created by Brian T. Keller. Email: bkeller2@g.ucla.edu
         Copyright (c) 2015 Brian Keller. All rights reserved.
         This is Beta software with no expressed license given.
               DO NOT DISTRIBUTE WITHOUT WRITTEN CONSENT

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

Please provide a path to an input file:
>>> 
```

Note that the ▮ denotes the cursor location in the command-line interface. One must provide BLImP with the *full* path to the input file in order for BLImP to find it. If

BLImP cannot find the file, an error message is given and BLImP exits.

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

5.2 Running BLImP via Command-line

BLImP is currently designed to run via a command-line shell. Several arguments can be supplied to BLImP in order to run input files or to facilitate simulations. In order to run an input, specify the path to the BLImP executable, a space, and then the path to the input file. These are the *full* path names.

```
$ /path/to/blimp/BLImp /path/to/input/INPUT.imp
```

BLImP currently does not export output as a file, thus no output filename should be given. In addition to this argument, three other arguments have been added in order to facilitate simulations.

`--data` `-d` The `data` argument will override the DATA command in the input file. A warning is displayed when this argument is used.

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

`--outfile` `-o` The `outfile` argument will override the OUTFILE command in the input file. A warning is displayed when this argument is used.

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

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

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

6 BLImP's Ouput

289

7 Convergence Diagnostics with BLImP

BLImP currently only implements a numeric summary for convergence diagnostics. BLImP uses a modified Potential Scale Reduce (PSR) to assess the convergence of the Markov chain Monte Carlo algorithm. By specifying the `psr` keyword in the OPTIONS command, the PSR will be computed during the burn-in iterations of the chain. This is memory intensive and it is suggested to run a separate input just to assess the convergence. Currently, the PSR is only implemented for a single chain.

7.1 Computation of PSR

7.2 Output from `psr`

Upon using the `psr` keyword, BLImP output will first note that diagnostic mode was enabled.

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


                          BLImP
                       vBETA 8.62
    Created by Brian T. Keller. Email: bkeller2@g.ucla.edu
     Copyright (c) 2015 Brian Keller. All rights reserved.
     This is Beta software with no expressed license given.
          DO NOT DISTRIBUTE WITHOUT WRITTEN CONSENT

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


Entering diagnostics mode.


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

Algorithmic Options Specified: hev, clmean, Prior IW0.
-------------------------------------------------------------
```

After the Gibbs sampler has started, the PSR will be calculated at every 100 iterations. The maximum PSR is displayed only and divided intow four categories: Fixed effects (`Fix eff`), random variance and covariance parameters (`Ran Var`), residual variances (`Err var`), and thresholds (`Threshold`). Below the values, the `Variable` row, BLImP lists the missing variable with the parameter.

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

PSR: Comparing iterations 51 to 75
                     with 76 to 100.

            |   Fix eff|   Ran Var|   Err Var| Threshold|
-----------------------------------------------------------
    Max PSR:|     1.370|     1.115|     1.124|       nan|
   Variable:|        ym|       x1m|       x1m|          |

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

Note that the output gives the specific iteration numbers that are being compared. In this example, BLImP is computing the PSR by comparing iterations 51 through 75 with iterations 76 to 100. In addition, note the use of `nan` to signal that the model used does not have those parameters. This can occur only for the `Ran Var` and `Threshold` columns. The PSR will continue to be computed every 100 iterations displaying similar output. Once the burn-in iterations have finished, an imputation is saved.

```
PSR: Comparing iterations 251 to 375
                     with 376 to 500.

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

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

   Chain 1 saving imputation 1 on Fri Oct  2 13:22:44 2015
-----------------------------------------------------------------

Writing out imputations.
-----------------------------------------------------------------

Writing completed.
Variable Order: imp# id ym x1m
-----------------------------------------------------------------
```

314

8 Running a Simulation

BLImP has utilities to run simulations with two different methods: (1) Within BLImP and (2) Outside of BLImP. Within BLImP simulations use BLImP to automate the analysis of multiple data sets with a common input. Outside of BLImP methods allow for users to analyze different data sets, with BLImP using additional scripting from outside programs. The goal of both methods are to easily automate the processes of BLImP that are required in methodological simulations.

8.1 Within BLImP Simulations

Within BLImP simulations offer users an easy interface to quickly run BLImP on simulated data sets. The only requirement is to have the data sets saved with a replication number, ranging from 1 to the number of replications desired, in the filename. The `SIMULATE` command is then added at the *top* of the input file. One must specify a * in both `DATA` and `OUTFILE` commands. The * will be replaced with the replication number. Note that one cannot specify separate files. Imputations will come stacked in a single file for each replication. In addition, diagnostics is not available with the `SIMULATE` command.

`SIMULATE:` The `SIMULATE` command must be placed above the other commands. A proper `SIMULATE` command consists of two keywords and arguments for those keywords.

`replications` The `replications` (`reps`) keyword must be followed by a space and the number of total replications that are desired. BLImP will start from 1 and increment that number by 1 in the path statement until reaching the number supplied.

`processors` The `processors` (`process`) keyword must be followed by a space and the number of total processors used. A maximum number of 8 is allowed. Specifying multiple processors will run each replication on a separate process. For example, specifying: `processors 4` will run 4 replications at a time until all are finished.

```
SIMULATE: replications 1000 processors 4;

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

VARNAMES: idvar x y z;

MODEL: idvar ~ x y z;

BURN: 500;

THIN: 100;

NIMPS: 10;

MISSING: -9999;

SEED: 38203;

CHAINS: 1;

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

OPTIONS: csv;
```

8.2 Output from **SIMULATE** command

The `SIMULATE` command gives the same output if it is a single processor simulation or a multiprocessor simulation. Adjusting the input from the previous section to only run five replication would produce the following output after BLImP's standard header. Note that the replications will not necessarily come in numerical order when the `processors` subcommand is used. Also note that if any errors occur and force a replication to exit, it will not produce imputations. BLImP will produce an error message that will state what replication failed. The number displayed corresponds to the number in the file name. It is suggested to note these replications and run them separately to see if it is a seeding issue (e.g., sometimes some seeds result in convergence issues).

```
WARNING: Entering simulation mode.



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

Algorithmic Options Specified: hev, clmean, Prior IW0.
-----------------------------------------------------------------


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

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

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

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

8.3 Outside of BLImP Simulations

BLImP offers the capabilities to run simulations outside of the program without rewriting the entire input file for each replication. This allows for a common input file and only changing specific parameters in the input file via the command-line. These commands are presented below, but are described in more detail in Section 5.2.

`-d` The `data` argument will override the DATA command in the input file.

`-o` The `outfile` argument will override the OUTFILE command in the input file.

`-s` The `seed` argument will override the SEED command in the input file.

In order to run the simulation outside of BLImP, an input file is required and a scripting or programming language must be employed to make the calls to the computer's command-line. For the purposes of an example, R statistical programming language will be used.

```
#########################################################
## Running a simulation via outside of 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
#########################################################
## INPUTS BEGIN HERE

# Set a Seed
seed       <- 37298372

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

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

9 Error and Warning Message Reference

Error messages (denoted with `ERROR:`) will cause BLImP to stop running and exit. In contrast, warning messages (denoted with `WARNING:`) will only warn the user, but BLImP will continue to run.

9.1 Error Messages

```
ERROR: Must specify simulation mode before specifying data
   command.
```

The `SIMULATE` command must be specified before the `DATA` command in the input file. For more information on how to properly set up a simulation input see Section 8.1.

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

BLImP read the number of replications specified in simulation mode as 0. Check input to verify that the number is greater than 0. For more information on how to properly set up a simulation input see Section 8.1.

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

BLImP read the number of processors to use in simulation mode as 0. Check input to verify that the number is greater than 0. For more information on how to properly set up a simulation input see Section 8.1.

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

The `CHAINS` command was used with simulation mode. Only single chains can be ran in within BLImP simulations. To specify multiple chains in a simulation you must run

an outside BLImP simulation. For more information on outside BLImP simulations see Section 8.3 in Chapter 8.

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

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

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

Currently the `psr` subcommand is not available in simulation mode. To obtain PSR statistics in simulation you must run an outside BLImP simulation. For more information on outside BLImP simulations see Section 8.3.

```
ERROR: Missing a data, variable name, or model statement.
```

Either the `DATA`, `VARNAMES`, and/or `MODEL` 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 input file see Chapter 2.

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

Either the `BURN`, `THIN`, `MISSING`, `SEED`, and/or 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 input file see Chapter 2.

```
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 input file see Chapter 2.

```
ERROR: Please place ONE asterisk (*) in outfile name.
```

BLImP did not detect an asterisk (*) in the OUTFILE command. This is required when the SIMULATE command or separate subcommand are specified. For more information on specifying a BLImP input file see Chapter 2.

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

BLImP read in more chains than the number of imputations that were requested in the input file. For more information on specifying a BLImP input file see Chapter 2.

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

BLImP read in more processors requested than the number of replications desired in the SIMULATION command. For more information on outside BLImP simulations see Section 8.3.

```
ERROR: PSR only available for 1 chain.
```

The current implementation of PSR is only available with one chain requested.

```
ERROR: More than one id variable given.
```

BLImP read in more than one id variable in the MODEL command. For more information on the MODEL command see Chapter 3.

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

The variable "X" (where "X" could be any variable name) was listed in the MODEL command and not in the VARNAMES command.

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

The variable that is to be used as an id variable is not in the VARNAMES command.

```
ERROR: Please place ONE asterisk (*) in input file name.
```

More than one asterisk (*) was placed in the file path given in the DATA command.

```
ERROR: The variable names and columns of data do not match.
```

The number of variables listed in the VARBANES command does not equal the number of columns the data matrix in BLImP has.

```
ERROR: No missing variables.
```

There were no missing variables listed in the MODEL command.

```
ERROR: Random effects variable not found.
```

A variable specified to have a random effect with another variable was not found.

```
ERROR: Random effects can only be specified with level 1
   variables.
```

A variable was specified to have a random effect with a level-2 variable.

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

BLImP was unable to open the input file.

```
ERROR: A covariance matrix became singular.
       Most likely a random effect is near zero.
       Either:
              (1) Try another seed.
              (2) Specify fewer random effects.
              (3) Specify fewer variables in model.
```

During the imputation process, a matrix became singular and BLImP was unable to continue. It is recommended to first try another seed. If this continues, then try to specify fewer random effects. Finally, try specifying fewer variables.

```
   Replication X had an error on date/time.
```

During the simulation command, the replication "X", where "X" is a number, had an error. It is recommended to run this replication over with another seed for no imputations were generated.

9.2 Warning Messages

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

An argument from the command-line is overriding the `OUTFILE` command in the input file supplied to BLImP . If this is not desired, see Section 5.2.

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

An argument from the command-line is overriding the `SEED` command in the input file supplied to BLImP . If this is not desired, see Section 5.2.

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

An argument from the command-line is overriding the `DATA` command in the input file supplied to BLImP . If this is not desired, see Section 5.2 .

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

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

```
WARNING: Maximum number of chains allowed is 8.
```

More than 8 chains were specified. BLImP will automatically change to 8 chains.

```
WARNING: Maximum number of threads allowed is 8.
```

More than 8 threads were specified. BLImP will automatically change to 8 threads.

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

Simulation mode was enabled with a `SIMULATE` command.